HP QuickTest Professional

Software Version: 10.00

User Guide

Manufacturing Part Number: T6513-90039
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

PART IV: ENHANCING TESTS

PART VII: MAINTAINING AND DEBUGGING TESTS

PART VIII: WORKING WITH THE QUICKTEST IDE

PART X: WORKING WITH ADVANCED TESTING FEATURES

PART XI: WORKING WITH QUALITY CENTER

PART XII: WORKING WITH OTHER HP PRODUCTS

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:

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.

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 I

Introducing QuickTest Professional

1

Introduction

Welcome to HP QuickTest Professional, the advanced solution for functional test and regression test automation. This next-generation automated testing solution deploys the concept of keyword-driven testing to enhance test creation and maintenance. Keyword-driven testing is a technique that separates much of the programming work from the actual test steps so that the test steps can be developed earlier and can often be maintained with only minor updates, even when there are significant changes in your application or your testing needs.

Using the keyword-driven approach, test automation experts have full access to the underlying test and object properties, via an integrated scripting and debugging environment that is round-trip synchronized with the Keyword View.

QuickTest Professional meets the needs of both technical and non-technical users. It works hand-in-hand with HP Business Process Testing to bring non-technical subject matter experts into the quality process in a meaningful way. Plus, it empowers the entire testing team to create sophisticated test suites.

QuickTest Professional provides add-ins that enable you to test objects (controls) created in commonly used development environments.

QuickTest Professional is Unicode compliant according to the requirements of the Unicode standard (http://www.unicode.org/standard/standard.html), enabling you to test applications in many international languages. Unicode represents the required characters using 8-bit or 16-bit code values. This allows processing and display of many diverse languages and character sets. You can test non-English language applications, as long as the relevant Windows language support is installed on the computer on which QuickTest Professional is installed (**Start** > **Settings** > **Control Panel** > **Regional Options** or similar). For additional information on Unicode and multi-lingual support issues, see the *HP QuickTest Professional Readme*.

This chapter includes:

Testing with QuickTest

When you open QuickTest, you can load environment-specific QuickTest add-ins, such as Java, .NET, and Web.

Note: You load add-ins using the Add-in Manager dialog box described in "Starting QuickTest" on page 20. You can find more information on the Add-in Manager dialog box and all QuickTest add-in environments in the *HP QuickTest Professional Add-ins Guide*.

Loading the relevant add-in enables QuickTest Professional to recognize and learn the objects in your application so that you can design automated tests that perform the same types of operations and business processes that your customers do. You can then run these tests to check that your application works as expected.

A test comprises calls to actions. Actions help divide your test into logical units, such as the main sections of a Web site, or specific activities that you perform in your application. By creating tests that call multiple actions, you can design tests that are more modular and efficient.

Each action comprises steps. As you add steps to an actions, they are displayed in the table-based Keyword View, or in the VBScript-based Expert View. Every step includes automatically generated documentation that provides a plain language textual description of what the step does.

While editing your test, you can instruct QuickTest to check the properties of specific objects in your application. For example, you can instruct QuickTest to check that a specific text string is displayed in a particular location in a dialog box, or you can check that a hypertext link on your Web page goes to the correct URL address.

You can further enhance your test by adding and modifying steps. You can also create function libraries and call their functions from your test. For example, you can define functions and use them as keywords in your test.

When you perform a run session, QuickTest performs each step in your test. After the run session ends, you can view a report detailing which steps were performed, and which ones succeeded or failed.

Note: Many QuickTest operations are performed using the mouse. In accordance with Section 508 of the W3C accessibility standards, QuickTest also recognizes operations performed using the **MouseKeys** option in the Windows Accessibility Options utility. Additionally, you can perform many QuickTest operations using shortcut keys. For a list of shortcut keys, see "Performing QuickTest Commands" on page 46.

You can use QuickTest process guidance to guide you through the process of creating a test. For more information, see "Working with Process Guidance" on page 1221.

Understanding the Testing Process

Testing with QuickTest involves the following main stages:

Stage 1: Analyzing Your Application

Before you begin creating a test, you need to analyze your application and determine your testing needs.

First, determine the development environments in which your application controls were developed, such as Web, Java, or .NET, so that you can load the required QuickTest add-ins.

Then determine the functionality that you want to test. To do this, consider the various activities that customers perform in your application to accomplish specific tasks. Which objects and operations are relevant for the set of business processes that need to be tested? Which operations require customized keywords to provide additional functionality?

While you are thinking about the business processes you want to test, consider how you can divide these processes into smaller units, which will be represented by your test's actions. Each action should emulate an activity that a customer might perform when using your application.

As you plan, try to keep the amount of steps you plan to include in each action to a minimum. Creating small, modular actions helps make your tests easier to read, follow, and maintain.

Stage 2: Preparing the Testing Infrastructure

To complete the infrastructure that is part of the planning process, you need to build the set of resources to be used by your tests, including shared object repositories containing test objects (which are representations of the objects in your application), function libraries containing functions that enhance QuickTest functionality, and so on. For more information, see Chapter 5, "Managing Test Objects in Object Repositories" and Chapter 31, "Working with User-Defined Functions and Function Libraries."

At this stage you also need to configure QuickTest according to your testing needs. This can include setting up your global testing preferences, your run session preferences, any test-specific preferences, and recovery scenarios. You can also create automation scripts that automatically set the required configurations (such as the add-ins to load) on the QuickTest client at the beginning of a run session. For more information, see Chapter 50, "Automating QuickTest Operations."

Lastly, you create one or more tests that serve as action repositories in which you can store the actions to be used in your tests. Generally, you create an action repository test for each area of your application to be tested. Storing all of your actions in specific tests enables you to maintain your actions in a central location. When you update an action in the action repository, the update is reflected in all tests that contain a call to that action. When you run a test, only the relevant action repository tests are loaded.

You then associate the shared object repositories with the relevant actions. This enables you to later insert steps using the objects stored in the object repositories.

When you create your tests, you insert calls to one or more of the actions stored in this repository.

Stage 3: Adding Steps to Your Actions

In this stage, you add steps to the actions in your test action repository.

Before you begin adding steps, make sure that you associate your function libraries and recovery scenarios with the relevant tests, so that you can insert steps using keywords.

You can create steps using the keyword-driven functionality available in the table-like, graphical Keyword View—or you can use the Expert View, if you prefer to program steps directly in VBScript. You can add steps to your test in one or both of the following ways:

- Drag objects from your object repository or from the Available Keywords pane to add keyword-driven steps in the Keyword View or Expert View. The object repository and Available Keywords pane contain all of the objects that you want to test in your application. (You create one or more object repositories when you prepare the testing infrastructure, as described in "Stage 2: Preparing the Testing Infrastructure" on page 8.)

  When you drag an object into the Keyword View, a step is created in the action with the default operation for that object. For example, if you drag a button object into the Keyword View, the click operation is automatically defined for the step. You can then modify the step as needed. For more information, see Chapter 14, "Working with the Keyword View" and Chapter 39, "Adding Keywords to Your Test." Advanced users can also add steps using the Expert View. For more information, see Chapter 29, "Working in the Expert View and Function Library Windows."

- Record on your application.

  As you navigate through your application during a recording session, QuickTest graphically displays each step you perform as a row in the Keyword View. A step is something that causes or makes a change in your application, such as clicking a link or image, or submitting a data form. In the Expert View, these steps are displayed as lines in a test script (VBScript). The **Documentation** column of the Keyword View also displays a description of each step in easy-to-understand sentences. For more information, see Chapter 14, "Working with the Keyword View."

Stage 4: Enhancing Your Test

You can enhance the testing process by modifying your test with special testing options and/or with programming statements, such as:

➤ Insert checkpoints and output values into your test.

A **checkpoint** checks specific properties or other characteristics of an object and enables you to identify whether or not your application is functioning correctly. For more information, see Chapter 17, "Understanding Checkpoints."

You can also use output values to extract data from your test. An **output value** is a value retrieved during the run session and entered into your Data Table or stored in a variable or a parameter. You can subsequently use this output value as input data in your test. This enables you to use data retrieved during a run session in other parts of the test. For more information, see Chapter 25, "Outputting Values."

➤ Broaden the scope of your test by replacing fixed values with parameters.

When you test your application, you can parameterize your steps to check how your application performs the same operations with different data. You may supply data in the Data Table, define environment variables and values, define test or action parameters and values, or instruct QuickTest to generate random numbers for current user and test data.

When you parameterize your test, QuickTest substitutes the fixed values in your test with the values stored in the relevant parameters. When you use Data Table parameters, QuickTest uses the values from a different row in the Data Table for each iteration of the test or action. (Each run session that uses a different set of parameterized data is called an iteration.) For more information, see Chapter 24, "Parameterizing Values."

➤ Add user-defined functions by creating function libraries and calling their functions from your test. For more information, see Chapter 31, "Working with User-Defined Functions and Function Libraries."

➤ Use the many functional testing features included in QuickTest to enhance your test and/or add programming statements to achieve more complex testing goals. For more information, see Chapter 28, "Adding Steps Containing Programming Logic."

Stage 5: Running and Debugging Your Test

After you create your test, you can perform different types of runs to achieve different goals.

- **Run your test to debug it.** You can control your run session to help you identify and eliminate defects in your test. You can use the **Step Into**, **Step Over**, and **Step Out** commands to run your test step by step. You can begin your run session from a specific step in your test, or run the test until a specific step is reached. You can also set breakpoints to pause your test at predetermined points. You can view or change the value of variables in your test each time it stops at a breakpoint in the Debug Viewer. You can also manually run VBScript commands in the Debug Viewer. For more information, see Chapter 35, "Debugging Tests and Function Libraries."
- **Run your test to check your application.** The test starts running from the first line in your test and stops at the end of the test. While running, QuickTest connects to your application and performs each operation in your test, including any checkpoints, such as checking any text strings, objects, tables, and so forth. If you parameterized your test with Data Table parameters, QuickTest repeats the test (or specific actions in your test) for each set of data values in the Data Table. For more information, see Chapter 32, "Running Tests."
- **Run your test to update it.**
  - You can run your test using **Maintenance Run Mode** when you know that your application has changed, and you therefore expect that QuickTest will not be able to identify the objects in your test. When you run a test in Maintenance Run Mode, a wizard opens for steps that fail because an object could not be found in the application. The wizard then guides you through the steps of resolving the issue, and, after you resolve the issue, the run continues. For more information, see Chapter 36, "Maintaining Tests."
  - You can run your test using **Update Run Mode** to update the property sets used for test object descriptions, the expected checkpoint values, the data available to retrieve in output values, and/or the Active Screen images and values.

Stage 6: Analyzing Test Results and Reporting Defects

After you run your test, you can view the results of the run in the Test Results window. You can view a summary of your results as well as a detailed report. If you captured still images or movies of your application during the run, you can view these from the Screen Recorder tab of the Test Results window. For more information, see Chapter 33, "Viewing Run Session Results." If you enabled local system monitoring for your test, you can view the results in the System Monitor tab of the Test Results window. For more information, see "Viewing System Monitor Results" on page 1063.

Finally, you can report defects detected during a run session. If you have access to Quality Center, the HP centralized quality solution, you can report the defects you discover to the project database. You can instruct QuickTest to automatically report each failed step in your test, or you can report them manually from the Test Results window. For more information, see Chapter 51, "Integrating with Quality Center."

Programming in the Expert View

You can use the Expert View tab to view a text-based version of your test. The test is composed of statements written in VBScript (Microsoft Visual Basic Scripting Edition) that correspond to the steps and checks displayed in the Keyword View. For more information, see Chapter 29, "Working in the Expert View and Function Library Windows."

For more information on the test objects and methods available for use in your test and how to program using VBScript, see the *HP QuickTest Professional Object Model Reference* and the *VBScript Reference* (select **Help** > **QuickTest Professional Help**).

Understanding Functions and Function Libraries

If you have sets of steps that are repeated in several actions or tests, you may want to consider creating and using user-defined functions. 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 in your tests, your tests are shorter, and easier to design, read, and maintain.

You can use the QuickTest function library editor to create and edit user-defined functions during your QuickTest session. A function library is a Visual Basic script containing VBscript functions, subroutines, modules, and so forth. You can also use the Function Definition Generator to assist you in defining new functions.

When you create a function, you can insert it directly in an action to make it available only within that action, or you can insert it in a function library to make it available to any test that is associated with that function library. For more information, see Chapter 31, "Working with User-Defined Functions and Function Libraries."

Managing the Testing Process Using Quality Center

You can use QuickTest together with Quality Center to manage the entire testing process. For example, you can use Quality Center to create a project (central repository) of manual and automated tests, build test cycles, run tests, and report and track defects. You can also create reports and graphs to help you review the progress of test planning, runs, and defect tracking before a software release.

In QuickTest, you can create tests and components and then save them directly to your Quality Center project. For more information, see Chapter 51, "Integrating with Quality Center." You can also run QuickTest tests from Quality Center and then use Quality Center to review and manage the results. For more information, see the *HP Quality Center User Guide*.

Finally, you can use Quality Center with Business Process Testing support to create business process tests, which are comprised of the business components you create either in QuickTest or Quality Center (with Business Process Testing support). For more information, see Chapter 56, "Working with Business Process Testing."

Understanding Business Process Testing

Business Process Testing is a role-based testing model that enables Subject Matter Experts—who understand the various parts of the application being tested—to create business process tests in Quality Center. Automation Engineers—who are experts in QuickTest and automated testing—use QuickTest to define all of the resources and settings required to create business process tests. Integration between QuickTest and Quality Center enables the Automation Engineer to effectively maintain the resources and settings, while enabling Subject Matter Experts to implement business process tests.

Business Process Testing uses a keyword-driven methodology for testing, based on the creation and implementation of business components and business process tests. A business component is an easily-maintained, reusable unit comprising one or more steps that perform a specific task within an application. A business process test comprises a series of business components, which together test a specific scenario or business process. For example, for a Web-based application, a business process test might contain five components—one for logging on to the application, another for navigating to specific pages, a third for entering data and selecting options in each of these pages, a fourth for submitting a form, and a fifth component for logging off of the application. Business components and business process tests are generally created in Quality Center by Subject Matter Experts, although Automation Engineers can also create business components in QuickTest.

In QuickTest, Automation Engineers define the resources and settings needed to create and run business components and business process tests. For example, the Automation Engineer can create function libraries to define various keywords (operations) and populate shared object repositories with test objects for the specific part of the application being tested. All resources and settings are saved in an application area, which is stored in a Quality Center project. By associating a business component with an application area, the component can access specific settings and resource files, such as function libraries, shared object repositories that contain the test objects used by the application, associated QuickTest add-ins, recovery scenario files, and so forth.

The Automation Engineer can create multiple application areas—each one focusing on a particular part (area) of the application being tested. For example, for a flight reservation application, one application area could be created for the login module, another application area for the flight search module, another for the flight reservation module, and still another for the billing module.

For more information on using QuickTest with Business Process Testing, see the *HP QuickTest Professional for Business Process Testing User Guide*.

Setting Required Access Permissions

You must make sure the following access permissions are set to run QuickTest Professional or to work with Quality Center.

Permissions Required to Run QuickTest Professional

You must have the following file system permissions:

- Full read and write permissions for all the files and folders under the folder in which QuickTest is installed
- Full read and write permissions to the Temp folder
- Read permissions to the Windows folder and to the System folder

You must have the following registry key permissions:

- Full read and write permissions to all the keys under **HKEY_CURRENT_USER\Software\Mercury Interactive**
- Read and Query Value permissions to all the **HKEY_LOCAL_MACHINE** and **HKEY_CLASSES_ROOT** keys

Permissions Required When Working with Quality Center

You must have the following permissions to use QuickTest with Quality Center:

- Full read and write permissions to the Quality Center cache folder
- Full read and write permissions to the QuickTest Add-in for Quality Center installation folder

Using the Sample Site

Many examples in this guide use the Mercury Tours sample Web site. The URL for this Web site is: http://newtours.demoaut.com.

Note that you must register a user name and password to use this site.

A sample Flight Windows-based application is also provided with the QuickTest Professional installation. You can access it from **Start** > **Programs** > **QuickTest Professional** > **Sample Applications** > **Flight**.

Modifying License Information

Working with QuickTest requires a license. When you install QuickTest, you select one of the following license types:

- a permanent **seat** license that is specific to the computer on which it is installed
- a network-based **concurrent** license that can be used by multiple QuickTest users

You can change your license type at any time (as long as you are logged in with administrator permissions on your computer). For example, if you are currently working with a seat license, you can choose to connect to a concurrent license server, if one is available on your network.

For information on modifying your license information, see the *HP QuickTest Professional Installation Guide*.

Updating QuickTest Software

By default, QuickTest automatically checks for online software updates once every seven days. When you start QuickTest, it checks to see if the last automatic Check for Updates took place more than seven days ago, and if so, it checks for updates.

You can also manually check for updates at any time by choosing **Help** > **Check for Updates** from within QuickTest, or by choosing **Start** > **Programs** > **QuickTest Professional** > **Check for Updates**.

If updates are available, you can choose which ones you want to download and (optionally) install. Follow the on-screen instructions for more information.

Tip: You can disable automatic checking for updates by clearing the **Check for software updates automatically** check box in the General pane of the Options dialog box. To open the Options dialog box, select **Tools** > **Options**.

2

QuickTest at a Glance

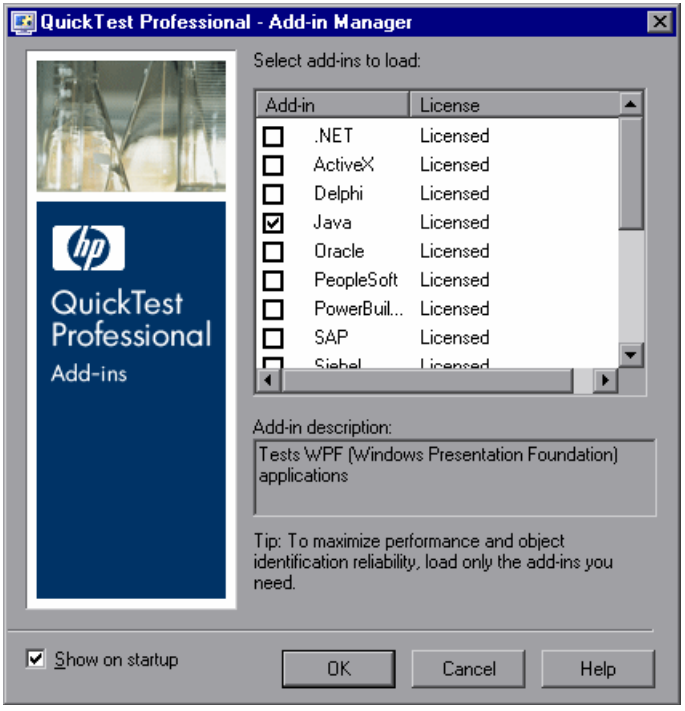
This chapter explains how to start QuickTest and introduces the QuickTest window.

This chapter includes:

Starting QuickTest

To start QuickTest, select **Programs** > **QuickTest Professional** > **QuickTest Professional** in the **Start** menu, or double-click the **QuickTest Professional** shortcut on your desktop.

The first time you start QuickTest, the Add-in Manager dialog box opens, displaying the currently installed add-ins. Select the add-ins you want to load.

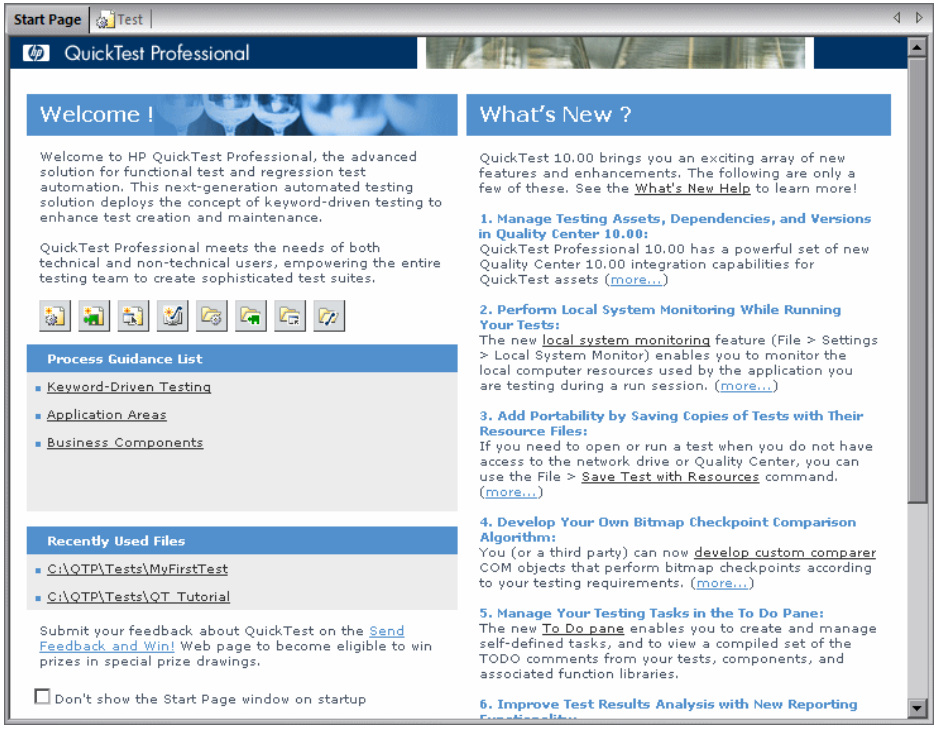


QuickTest remembers the add-ins you load so that the next time you open QuickTest, the add-ins you selected in the previous session are selected by default. For best performance, it is recommended to clear any add-ins that are not needed for a particular session.

Tip: If you do not want this dialog box to open the next time you start QuickTest, clear the **Show on startup** check box.

For more information on installing, loading, and working with add-ins, see the *HP QuickTest Professional Installation Guide* and the *HP QuickTest Professional Add-ins Guide*.

Click **OK**. The QuickTest Professional window opens displaying the Start Page and a blank test. To access a blank test, click the **Test** tab.



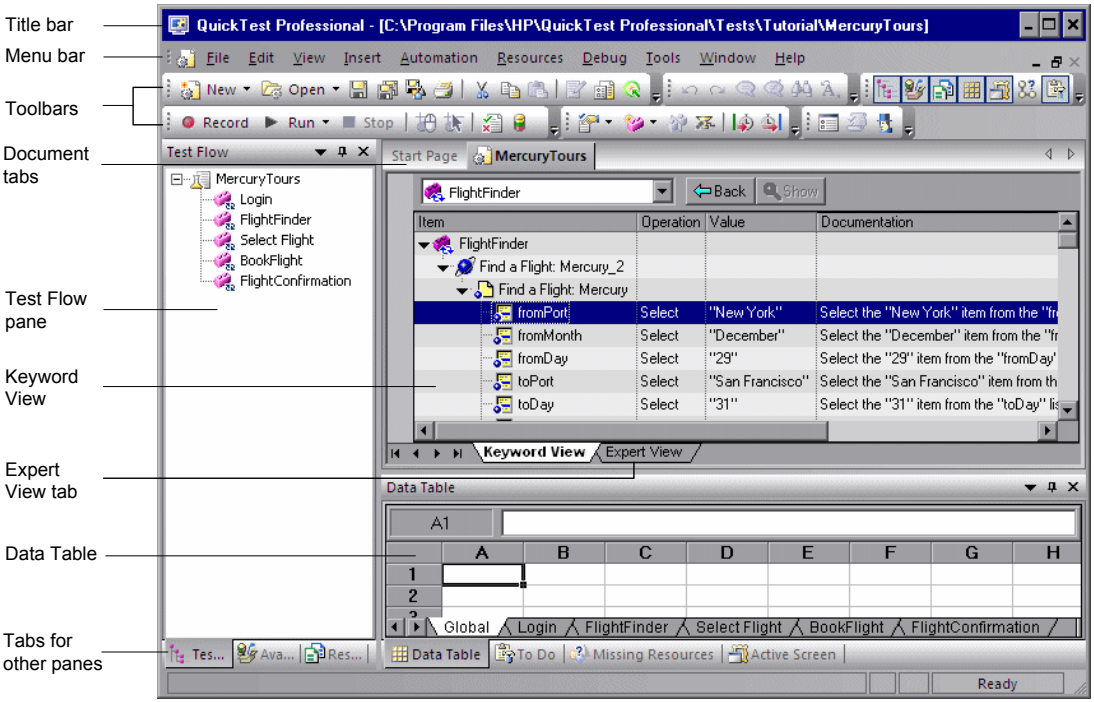
In the Start page, you can:

- Click a QuickTest process guidance link for best practices on working with QuickTest. If your organization has its own custom process guidance, you may be able to click the link for it in the **Process Guidance List**.
- Click a shortcut button to open a new or existing test or function library. If business process testing is enabled, you can also open a new or existing business component or application area.
- Click the links in the **What's New** section to learn more about the new features provided with this version of QuickTest.

For more information on the Start Page, see "Start Page" on page 31.

The QuickTest Window

The QuickTest window displays your testing documents in the document area.



You can work on one test and one or more function libraries simultaneously. (For your convenience, you can display one active document in the document area, or you can cascade or tile your open documents.) For more information, see “Working with Multiple Documents” on page 1159.

Document Area

The document area of the QuickTest window can display the following:

- **Test.** Enables you to create, view, and modify your test in Keyword View or Expert View (described below).
- **Function Library.** Enables you to create, view, and modify functions and subroutines for use with your test. For more information, see Chapter 31, "Working with User-Defined Functions and Function Libraries."
- **Start Page.** Welcomes you to QuickTest and provides links to Process Guidance. You can use the shortcut buttons to open new and existing documents. For more information, see "Start Page" on page 31.

Note: The document area of the QuickTest window also enables you to create, view, and modify business components, scripted components, and application areas. For more information, see the *HP QuickTest Professional for Business Process Testing User Guide*.

Key Elements in the QuickTest Window

In addition to the document area, the QuickTest window contains the following key elements:

- **QuickTest title bar.** Displays the name of the active document. If changes have been made since it was last saved, an asterisk (*) is displayed next to the document name in the title bar.
- **Menu bar.** Displays menus of QuickTest commands.
- **Standard toolbar.** Contains buttons to assist you in managing your document.
- **Automation toolbar.** Contains buttons to assist you in the testing process.
- **Debug toolbar.** Contains buttons to assist you in debugging your document. (Not displayed by default)
- **Edit toolbar.** Contains buttons to assist you in editing your test or function library.

- **Insert toolbar.** Contains buttons to assist you when working with steps and statements in your test or function library.
- **Tools toolbar.** Contains buttons with tools to assist you in the testing process.
- **View toolbar.** Contains buttons to assist you in viewing your document.
- **Action toolbar.** Contains buttons and a list of actions, enabling you to view the details of an individual action or the entire test flow. (Not displayed by default)
- **Document tabs and scroll arrows.** Enables you to navigate open documents in the document area by selecting the tab of the document you want to activate (bring into focus). When there is not enough space in the document area to display all of the tabs simultaneously, you can use the left and right arrows to scroll between your open documents.
- **Keyword View.** Contains each step, and displays the object hierarchy, in a modular, icon-based table. For more information, see Chapter 14, "Working with the Keyword View."
- **Expert View.** Contains each step as a VBScript line. In object-based steps, the VBScript line defines the object hierarchy. For more information, see Chapter 29, "Working in the Expert View and Function Library Windows."
- **Status bar.** Displays the status of the QuickTest application and other relevant information.

You can show or hide the following panes from the **View** menu:

- **Active Screen.** Provides a snapshot of your application as it appeared when you performed a certain step during the recording session.
- **Available Keywords.** Displays all the keywords available to your test. Enables you to drag and drop objects or calls to functions into your test.
- **Data Table.** Assists you in parameterizing your test. The Data Table contains the **Global** tab and a tab for each action.
- **Debug Viewer.** Assists you in debugging your document. The Debug Viewer pane contains the **Watch**, **Variables**, and **Command** tabs.
- **Information.** Displays a list of syntax errors found in your test and function library scripts.

- **Missing Resources.** Provides a list of the resources that are specified in your test but cannot be found, such as missing calls to actions, unmapped shared object repositories, and parameters that are connected to shared object repositories. The Missing Resources pane then enables you to locate or remove them from your test.
- **Process Guidance.** Displays two panes that provide procedures and descriptions on how to best perform specific processes, such as creating a test in QuickTest. The Process Guidance Activities pane lists the activities that you can perform, such as adding steps to a test. The Process Guidance Description pane describes the tasks that you need to perform for a selected activity. Your organization may also provide you with process guidance that is accessible from these panes.
- **Resources.** Displays all the resources associated with your current test and enables you to manage these resources.
- **Test Flow.** Displays the hierarchy of actions and action calls in the current test, and shows the order in which they are run.
- **To Do.** Displays and enables you to manage the tasks defined for the current test. The To Do pane also displays the TODO comment steps of the test's actions or currently open function libraries.

You can modify the QuickTest window to create user-defined menus and to customize the appearance of existing menus and toolbars. For more information, see "Customizing Toolbars and Menus" on page 1146.

You can also customize the layout of the QuickTest window by moving, resizing, displaying, or hiding most of the elements. QuickTest remembers your preferred layout settings and opens subsequent sessions with your customized layout. For more information, see "Modifying the QuickTest Window Layout" on page 1135.

Changing the Appearance of the QuickTest Window

By default, the QuickTest window uses the Microsoft Office 2003 theme. You can change the look and feel of the main QuickTest window, as required.

To change the appearance of the main QuickTest window:

In the QuickTest 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 QuickTest window only if your computer is set to use a Windows XP theme.

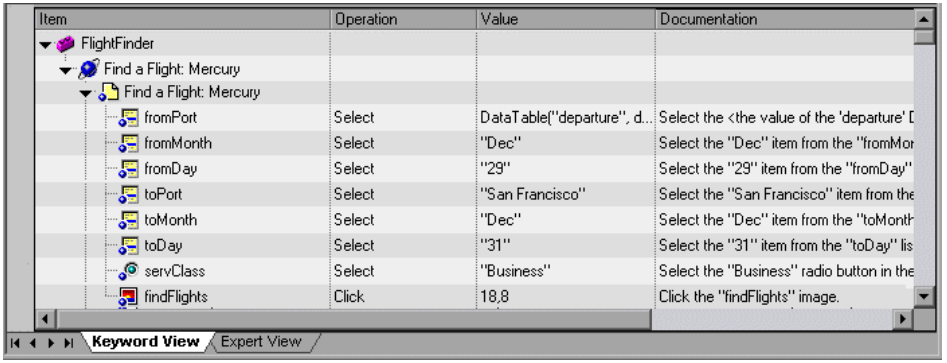
Tip: You can also change the theme used for the Test Results window. For more information, see "Changing the Appearance of the Test Results Window" on page 979.

Keyword View

The Keyword View enables you to create and view the steps of your test in a keyword-driven, modular, table format. The Keyword View is comprised of a table-like view, in which each step is a separate row in the table, and each column represents different parts of the steps. You can modify the columns displayed to suit your requirements.

You create and modify tests by selecting items and operations in the Keyword View and entering information as required. Each step is automatically documented as you complete it, enabling you to view a description of your test steps in understandable English.

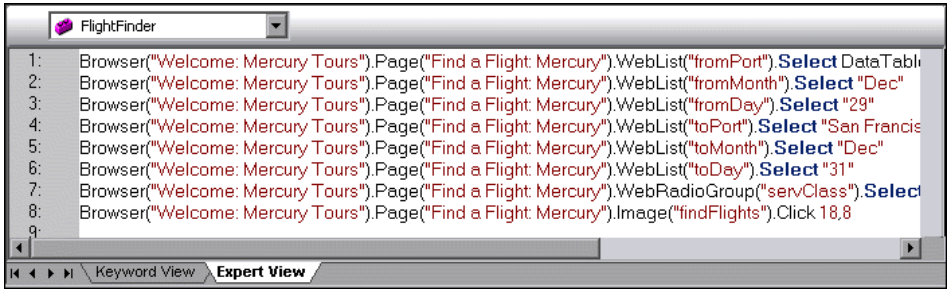
Each operation performed on your application during a recording session is recorded as a row in the Keyword View.



For each row in the Keyword View, QuickTest displays a corresponding line of script in the Expert View. If you focus on a specific step in the Keyword View and switch to the Expert View, the cursor is located in that corresponding line of the test. For more information on using the Keyword View, see Chapter 14, “Working with the Keyword View.”

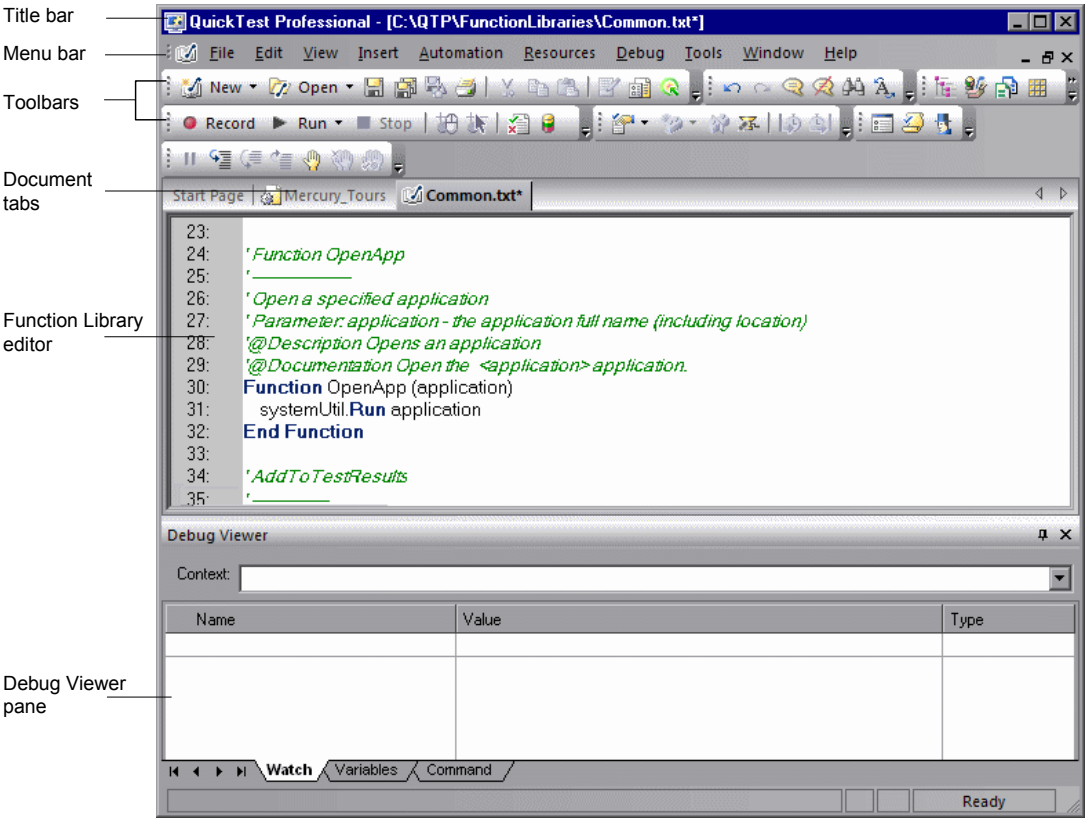
Expert View

In the Expert View, QuickTest displays each operation performed on your application in the form of a script, comprised of VBScript statements. The Expert View is a script editor with many script editing capabilities. For each object and method in an Expert View statement, a corresponding row exists in the Keyword View. The action list above the Expert View window lists the actions that are called from the test. For more information on using the Expert View, see Chapter 29, "Working in the Expert View and Function Library Windows."



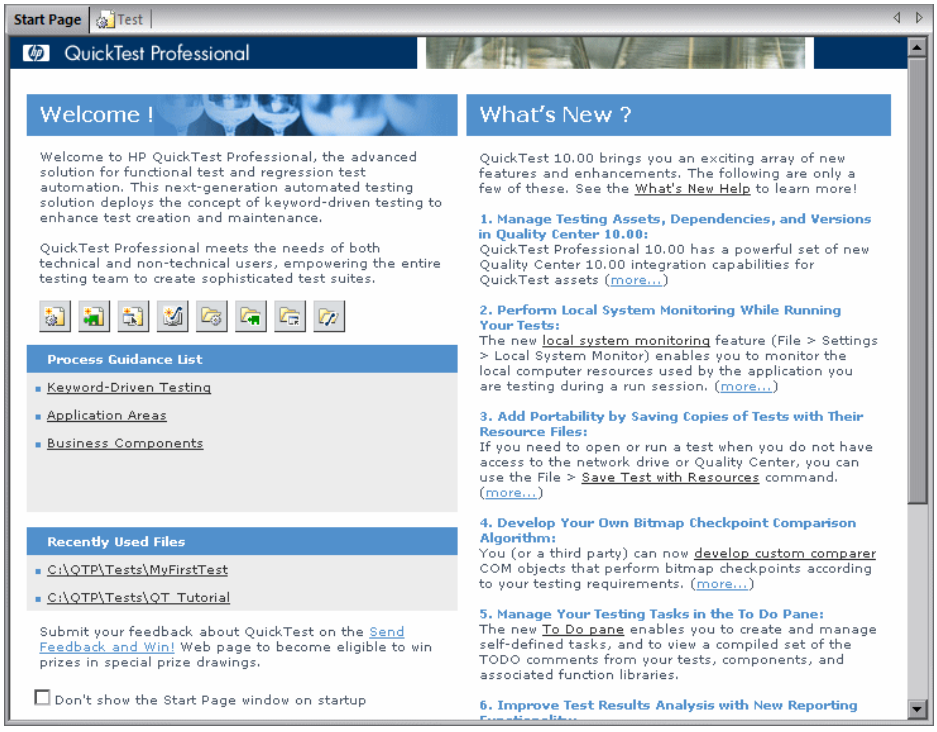
Function Library

QuickTest provides a built-in editor that enables you to create and debug function libraries using the same editing features that are available in the Expert View. Each function library is a separate QuickTest document containing VBscript functions, subroutines, classes, modules, and so forth. Each function library opens in its own window, in addition to the test that is already open. You can 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. For more information, see Chapter 31, "Working with User-Defined Functions and Function Libraries."



Start Page

The Start Page welcomes you to QuickTest and describes the new features in this release—including links to more information about these features. It also provides links to Process Guidance, a tool that offers best practices for working with QuickTest. If your organization has descriptions for its own custom processes, these processes may also be available from the **Process Guidance List**. For more information, see "Working with Process Guidance" on page 1221.



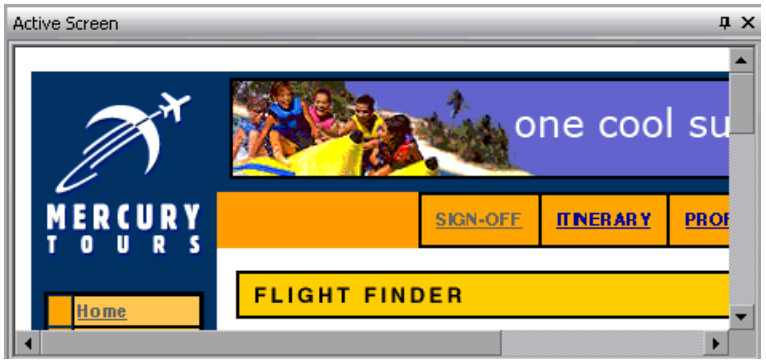
You can open a document from the list of **Recently Used Files**, or you can click the buttons in the **Welcome!** area to open new or existing documents:

Click	to...
	Open a new test.
	Open a new business component.
	Open a new application area.
	Open a new function library.
	Open an existing test.
	Open an existing business component.
	Open an existing application area.
	Open an existing function library.

Tip: If you do not want QuickTest to display the Start Page when you next open QuickTest, select the **Don't show the Start Page window on startup** check box. When you select this option, the Start Page is also automatically hidden for the current QuickTest session as soon as you open another QuickTest document. To display the Start Page again, select **View > Start Page**.

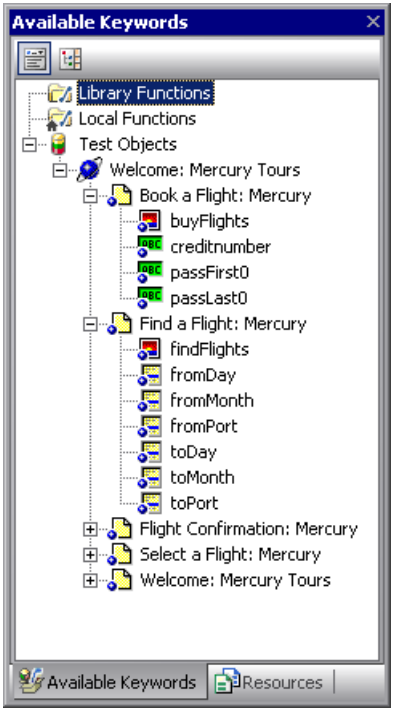
Active Screen

The Active Screen provides a snapshot of your application as it appeared when you performed a certain step during a recording session. Additionally, depending on the Active Screen capture options that you used while recording, the page displayed in the Active Screen can contain detailed property information about each object displayed on the page. To view the Active Screen, click the **Active Screen** button or select **View** > **Active Screen**. For more information, see "Working with the Active Screen" on page 376.



Available Keywords Pane

The Available Keywords pane enables you to drag and drop objects or calls to functions into your test. When you drag and drop an object into your test, 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. To view the Available Keywords pane, click the **Available Keywords Pane** button or select **View** > **Available Keywords**.



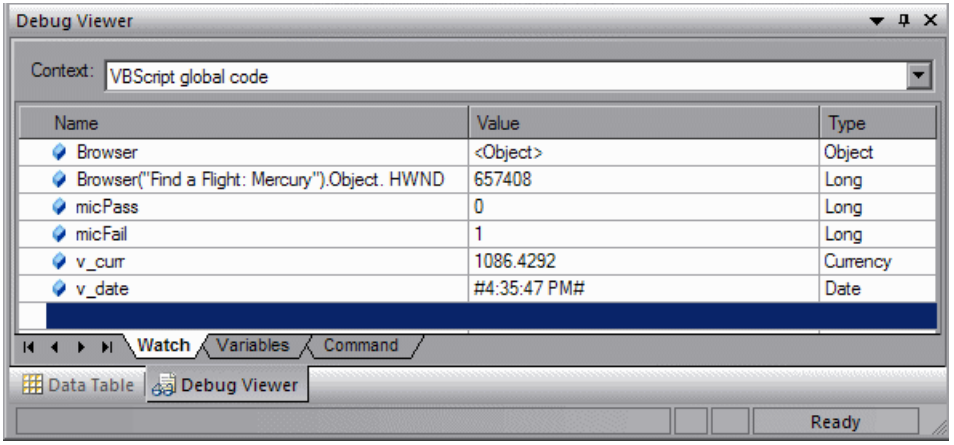
For more information, see “Understanding the Available Keywords Pane” on page 1165.

Data Table

The Data Table contains one Global tab plus an additional tab for each action in your test. The Data Table assists you in parameterizing your test. To view the Data Table, click the **Data Table** toolbar button or select **View** > **Data Table**. The Data Table is a spreadsheet-like sheet with columns and rows representing the data applicable to your test. For more information, see Chapter 42, "Working with Data Tables."

Debug Viewer Pane

The Debug Viewer pane assists you in debugging your tests or function libraries. To view the Debug Viewer pane, select **View** > **Debug Viewer**.



This pane contains three tabs—Watch, Variables, and Command.

Watch

The Watch tab displays the current value and type of any variable or VBScript expression that you added to the Watch tab. You can also set or modify the values of the variables and properties that are displayed.

Variables

The Variables tab displays the current value and type of all variables that were recognized up to the last step performed during the run session that you are debugging. You can also set or modify the values of the variables that are displayed.

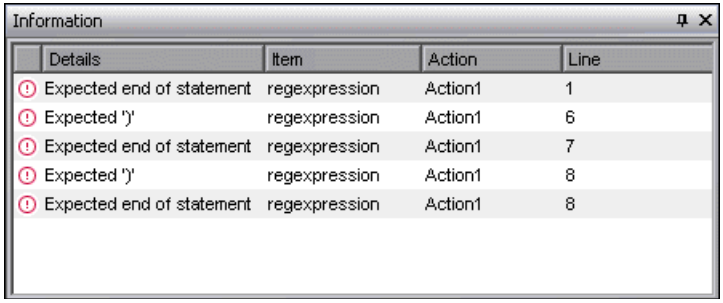
Command

The Command tab enables you to run lines of script to set or modify the current value of a variable or VBScript object in your test or function library.

For more information, see “The Debug Viewer Pane” on page 1082.

Information Pane

The Information pane provides a list of syntax errors in your test or function library scripts. To show or hide the Information pane, select **View** > **Information**.



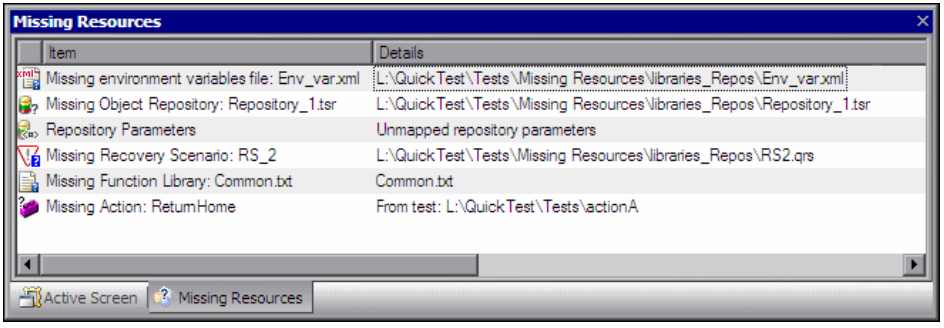
When you switch from the Expert View to the Keyword View, QuickTest automatically checks for syntax errors in your script, and shows them in the Information pane. If the Information pane is not currently displayed, QuickTest automatically opens it when a syntax error is detected.

You can double-click a syntax error to locate the error in the script or function library, and then correct it. For more information, see “Handling VBScript Syntax Errors” on page 860.

Missing Resources Pane

The Missing Resources pane provides a list of the resources that are specified in your test but cannot be found. Missing resources can include calls to missing actions, missing function libraries, missing recovery scenarios, missing environment variable XML files, unmapped shared object repositories, and parameters that are connected to shared object repositories.

Each time you open your test, QuickTest automatically checks that all specified resources are accessible. If it finds any resources that are not accessible, QuickTest lists them in the Missing Resources pane. If the Missing Resources pane is not currently displayed, QuickTest automatically opens it when a missing resource is detected. To show or hide the Missing Resources pane, select **View** > **Missing Resources** or click the **Missing Resource** button.



The Missing Resources pane contains the following columns.

- The **Item** column lists the missing resources.
- The **Details** column provides information about each missing resource, such as the location in which QuickTest expects to find the resource.

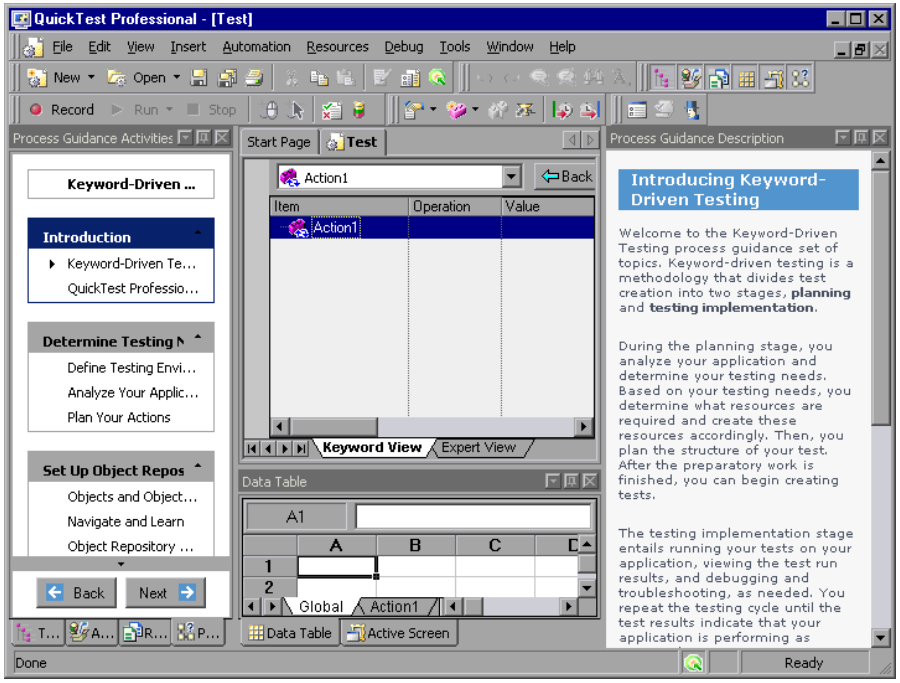
You can double-click a missing resource to remap it or remove it. You can also filter the pane to display a specific type of missing resource, such as Missing Object Repository and hide the other types.

For more information, see “Handling Missing Resources” on page 1179.

Process Guidance Panes

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.

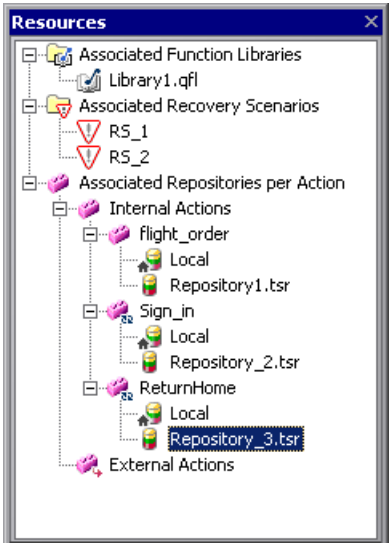
Process guidance is displayed in two panes: the **Process Guidance Activities** pane and the **Process Guidance Description** pane. You display or hide these panes by choosing **View** > **Process Guidance** or clicking the **Process Guidance panes** toggle button.



The **Process Guidance Activities** pane (shown on the left) lists the activities that are part of the selected process. The **Process Guidance Description** pane (shown on the right) displays the topic (description), for the selected activity. For more information, see Chapter 43, “Working with Process Guidance.”

Resources Pane

Tests and actions are associated with resources such as function libraries, recovery scenarios, and object repositories. QuickTest displays all the resources associated with a test in the Resources pane. The Resources pane enables you to add, remove, and manage all of the resources in your test. To view the Resources pane, click the **Resources Pane** button or select **View** > **Resources**.

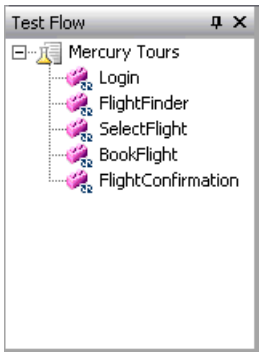


For more information, see “The Resources Pane” on page 1161.

Test Flow Pane

The Test Flow pane is comprised of a hierarchy of actions and action calls in the current test, and shows the order in which they are run. Each action is displayed as a node in a tree, and includes calls to all of a test's actions. The steps of the action that you double-click in the Test Flow pane are displayed in the Keyword View and Expert View.

The Test Flow pane is displayed by default when you start QuickTest Professional. To view the Test Flow pane, click the **Test Flow Pane** button or select **View** > **Test Flow**.

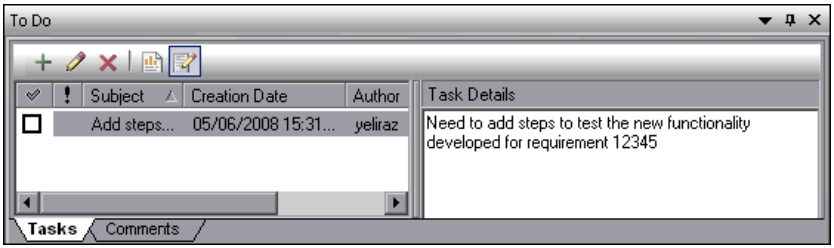


For more information, see "Using the Test Flow Pane" on page 431.

To Do Pane

The To Do pane enables you to create, view, and manage your TODO tasks. A TODO task is anything that needs to be done in a test, such as providing information relevant for handing over a testing document, or adding a reminder to yourself to add steps that test a new page in your application. You can assign tasks to others, and you can mark a task as complete when it is done. Your TODO tasks are saved with the test.

The To Do pane also enables you to view the TODO comments that exist in the action or an open function library (for example, instructions or notes adjacent to a step). To show or hide the To Do pane, select **View** > **To Do** or click the **To Do Pane** toolbar button.



For more information, see “Managing QuickTest Tasks and Comments” on page 1169.

Using QuickTest Commands

You can select QuickTest commands from the menu bar or from a toolbar. QuickTest displays a different set of commands and toolbar buttons for tests. Each set is customized for the type of document you are creating or modifying. You can also perform some QuickTest commands by pressing shortcut keys or selecting commands from context (right-click) menus. The menus and toolbars are enabled according to the active document type.

Most commands are available from the menu bar or by pressing shortcut keys. You can perform frequently used QuickTest commands by clicking buttons in the toolbars. For more information, see:

- "QuickTest Toolbars" on page 44
- "File Menu Commands" on page 47
- "Edit Menu Commands" on page 50
- "View Menu Commands" on page 54
- "Insert Menu Commands" on page 55
- "Automation Menu Commands" on page 58
- "Resources Menu Commands" on page 60
- "Debug Menu Commands" on page 61
- "Tools Menu Commands" on page 62
- "Window Menu Commands" on page 64
- "Help Menu Commands" on page 64
- "Data Table Menu Commands" on page 66
- "Other QuickTest Commands" on page 68

QuickTest Toolbars

This section describes the QuickTest built-in toolbars.

You can add, remove, reorder, or change the appearance of the QuickTest toolbars using the Customize Toolbars and Menus dialog box and the Button Appearance Dialog Box. For more information, see "Customizing Toolbars and Menus" on page 1146.

Standard Toolbar

The **Standard** toolbar contains buttons for managing a test or function library.

For information on the **Standard** toolbar buttons, see "File Menu Commands" on page 47.

Note: The icons for the **New** and **Open** buttons change depending on the type of active document, such as test or function library.

For more information on managing your test, see Chapter 12, "Creating Tests Using the Keyword-Driven Methodology." For more information on managing business process tests, see Chapter 56, "Working with Business Process Testing." For more information on working with function libraries, see Chapter 31, "Working with User-Defined Functions and Function Libraries."

Automation Toolbar

The **Automation** toolbar contains buttons for recording and running your test.

For information on the **Automation** toolbar buttons, see "Automation Menu Commands" on page 58.

Debug Toolbar

The **Debug** toolbar contains buttons for the commands used when debugging the steps in your test and any associated function library.

For information on the **Debug** toolbar buttons, see "Debug Menu Commands" on page 61.

Edit Toolbar

The **Edit** toolbar contains buttons for the commands used when editing your test or function library.

For information on the **Edit** toolbar buttons, see "Edit Menu Commands" on page 50.

Insert Toolbar

The **Insert** toolbar contains buttons for the commands used when creating and modifying your test steps and when working with function libraries.

For information on the **Insert** toolbar buttons, see "Insert Menu Commands" on page 55.

Tools Toolbar

The **Tools** toolbar contains buttons for the commands used to access tools that assist you when working with your test.

For information on the **Tools** toolbar buttons, see "Tools Menu Commands" on page 62.

View Toolbar

The **View** toolbar contains buttons for viewing different elements of the QuickTest window.

For information on the **View** toolbar buttons, see "View Menu Commands" on page 54.

Action Toolbar

The **Action** toolbar is available in the Keyword View and contains options that enable you to view all actions in the test flow or to view the details of a selected action. The following options are displayed on the **Action** toolbar:

When your test contains reusable or external actions, the Action toolbar is always visible. If there are no reusable or external actions in your test, you can select **View** > **Toolbars** > **Action** to show the Action toolbar.

When you have reusable or external actions in your test, only the action icon is visible when viewing the entire Test Flow in the Keyword View. You can view the details of the reusable or external actions by double-clicking on the action, selecting the action name from the list in the Action toolbar, or selecting the action in the Keyword View and clicking the **Show** button. You can return to the Test Flow by clicking the **Back** button.

For more information on actions, see Chapter 15, "Working with Actions" and Chapter 16, "Working with Advanced Action Features."

Performing QuickTest Commands

In addition to performing frequently-used commands by clicking toolbar buttons, you can perform most QuickTest commands by choosing the relevant menu option. You can also perform some QuickTest commands by pressing the relevant shortcut keys.

File Menu Commands

You can manage your test or function library using the following **File** menu commands:

	Command	Shortcut Key	Function
	**New > Test**	CTRL+N	Creates a new test.
	**New > Business Component**	CTRL+SHIFT+N	Creates a new business component.
	**New > Scripted Component**		Creates a new scripted component.
	**New > Application Area**	CTRL+ALT+N	Creates a new application area.
	**New > Function Library**	SHIFT+ALT+N	Creates a new function library.
	**Open > Test**	CTRL+O	Opens an existing test.
	**Open > Business/ Scripted Component**	CTRL+SHIFT+O	Opens an existing business or scripted component.
	**Open > Application Area**	CTRL+ALT+O	Opens an existing application area.
	**Open > Function Library**	SHIFT+ALT+O	Opens an existing function library.
	Close		Closes the active function library.
	Close All Function Libraries		Closes all open function libraries.

	Command	Shortcut Key	Function
	**Quality Center Connection**		Opens the Quality Center Connection dialog box, enabling you to connect to a Quality Center project. **Tip:** Double-click the Quality Center icon on the status bar to manage your connection. Point to the Quality Center icon on the status bar to view connection information. 
	Quality Center Version Control		Provides a sub-menu of options for managing versions of QuickTest assets and baselines in Quality Center. The version-related sub-menu is available only when you are connected to version-control enabled Quality Center project.
	**Save**	CTRL+S	Saves the active document.
	Save As		Opens the relevant Save dialog box so you can save the open document.
	**Save Test with Resources**		Saves a standalone copy of the current test together with its resource files.
	**Save All**		Saves all open documents.
	**Enable Editing**		Makes read-only function libraries editable.
	Export Test to Zip File	CTRL+ALT+S	Creates a zip file of the active test.

	Command	Shortcut Key	Function
	Import Test from Zip File	CTRL+ALT+I	Imports a test from a zip file.
	Convert to Scripted Component	CTRL+ALT+C	Converts a business component to a scripted component.
	**Print**	CTRL+P	Prints the active document.
	Print Preview		Displays the Keyword View as it will look when printed and enables you to modify the page setup.
	**Settings**		Opens the Settings dialog box, enabling you to define settings for the open document. (Not relevant for function libraries)
	Process Guidance Management		Opens the Process Guidance Management dialog box, enabling you to manage the list of processes that are available in QuickTest.
	Associate Library '<Function Library Name>' with '<Document Name>'		Associates the active function library with the open document. (Available only from function libraries)
	Recent Files		Lists the recently viewed files.
	Exit		Closes the QuickTest session.

Many of the **File** menu commands are also available from the **Standard** toolbar (described on page 44).

Edit Menu Commands

You can manage your test actions and your test or function library steps using the following **Edit** menu commands:

	Command	Shortcut Key	Function
	**Undo**	CTRL+Z	Reverses the last command or deletes the last entry you typed.
	**Redo**	CTRL+Y	Reverses the most recent operation of the Undo command.
	**Cut**	CTRL+X	Removes the selection from your document.
	**Copy**	CTRL+C	Copies the selection from your document.
	**Paste**	CTRL+V	Pastes the selection to your document.
	**Delete**	DELETE	Deletes the selection from your document.
	Copy Documentation to Clipboard		Copies the content of the **Documentation** column of the Keyword View, enabling you to paste it in an external application.
	**Action > Split Action**		Separates an action into two sibling actions or into parent-child nested actions.
	Action > Rename Action	SHIFT+F2	Changes the name of an action.
	**Action > Delete Action**		Enables you to remove the selected call to the action, or delete the action and its calls from the active test.

	Command	Shortcut Key	Function
	Action > **Action Properties**		Enables you to specify options, parameters, and associated object repositories for a stored action.
	Action > **Action Call Properties**		Enables you to specify the number of run iterations according to the number of rows in the Data Table, and to define the values of input parameters and the storage location of output parameters.
	Step Properties > **Comment Properties**	CTRL+ENTER; ALT+ENTER	Opens the Comment Properties dialog box for a comment step. Available only when the selected step is a comment.
	Step Properties > **Object Properties**	CTRL+ENTER; ALT+ENTER	Opens the Object Properties dialog box for a selected object. Available only when the selected step contains a test object.
	Step Properties > **Checkpoint Properties**		Opens the relevant Checkpoint Properties dialog box for a selected object. Available only when the selected step is a checkpoint step.
	Step Properties > **Output Value Properties**		Opens the relevant Output Value Properties dialog box for a selected object. Available only when the selected step is an output value step.
	Step Properties > **Report Properties**	CTRL+ENTER; ALT+ENTER	Displays the Report Properties dialog box for a report step. Available only when the selected step is a **Reporter.ReportEvent** step.

	Command	Shortcut Key	Function
	**Find**	CTRL+F	Searches for a specified string.
	**Replace**	CTRL+H	Searches and replaces a specified string.
	**Go To**	CTRL+G	Moves the cursor to a particular line in the test or function library.
	Bookmarks	CTRL+B	Creates bookmarks in your test or function library for easy navigation.
	**Advanced > Comment Block**	CTRL+M	Comments out the current row, or selected rows.
	**Advanced > Uncomment Block**	CTRL+SHIFT+M	Removes the comment formatting from the current or selected rows.
	**Advanced > Indent**	TAB	Indents the step according to the tab spacing defined in the Editor Options dialog box.
	**Advanced > Outdent**	BACKSPACE	Outdents the step (reduces the indentation) according to the tab spacing defined in the Editor Options dialog box.
	Advanced > Go to Function Definition	ALT+G	Navigates to the definition of the selected function.
	Advanced > Complete Word	CTRL+SPACE	Completes the word when you type the beginning of a VBScript method or object.
	Advanced > Argument Info	CTRL+SHIFT+ SPACE	Displays the syntax of a method.
	Advanced > Apply "With" to Script	CTRL+W	Generates **With** statements for the action displayed in the Expert View, and enables IntelliSense within **With** statements.

	Command	Shortcut Key	Function
	Advanced > **Remove "With" Statements**	CTRL+SHIFT+W	Converts any **With** statements in the action displayed in the Expert View to regular (single-line) VBScript statements.
	Optional Step		Inserts an optional step (a step that is not required to successfully complete a run session).

Many of the **Edit** menu commands are also available from the **Edit** toolbar (described on page 45).

View Menu Commands

You can manage the way that QuickTest is displayed on your screen using the following **View** menu commands:

	Command	Function
	Start Page	Opens the Start Page. (Enabled only when the Start Page is closed)
	**Active Screen**	Displays the Active Screen.
	**Available Keywords**	Shows and hides the Available Keywords Pane.
	**Data Table**	Displays the Data Table.
	Debug Viewer	Shows and hides the Debug Viewer Pane.
	Information	Shows and hides the Information Pane.
	Missing Resources	Shows and hides the Missing Resources Pane.
	**Process Guidance**	Shows and hides the Process Guidance Panes.
	**Resources**	Shows and hides the Resources Pane.
	**Test Flow**	Shows and hides the Test Flow Pane. (Relevant only for tests)
	**To Do**	Shows and hides the To Do Pane.
	Expand All	Expands all steps in the Keyword View.
	Collapse All	Collapses all steps in the Keyword View.
	**Keyword View**	Displays the Keyword View if the Expert View is displayed.
	**Expert View**	Displays the Expert View if the Keyword View is displayed.
	Toolbars	Enables you to show and hide QuickTest toolbars.
	Window Theme	Enables you to select a theme to apply to your QuickTest window.

Some of the **View** menu commands are also available from the **View** toolbar (described on page 46).

Insert Menu Commands

You can insert various types of test and function library steps using the following **Insert** menu commands:

	Command	Shortcut Key	Function
	Checkpoint > Existing Checkpoint	ALT +F12	Opens the Add Existing Checkpoint dialog box, enabling you to insert an existing checkpoint for an object or a table. **Note:** From the menu option, context-menu option, or toolbar button, you can also insert other types of checkpoints, if available.
	**Checkpoint > Standard Checkpoint**	F12	Opens the Checkpoint Properties dialog box, enabling you to create a standard checkpoint for an object or a table. **Note:** From the menu option, context-menu option, or toolbar button, you can also insert other types of checkpoints, if available.
	**Output Value > Existing Output Value**	SHIFT+CTRL+ F12	Opens the Add Existing Output Value dialog box, enabling you to create a standard output value for an object or a table. **Note:** From the menu option, context-menu option, or toolbar button, you can also insert other types of output values, if available.

	Command	Shortcut Key	Function
	**Output Value > Standard Output Value**	CTRL+F12	Opens the Output Value Properties dialog box, enabling you to create a standard output value for an object or a table. **Note:** From the menu option, context-menu option, or toolbar button, you can also insert other types of output values, if available.
	**Step Generator**	F7	Opens the Step Generator.
	**Function Definition Generator**		Opens the Function Definition Generator.
	**Synchronization Point**		Inserts a synchronization point in the test, instructing QuickTest to pause the test until the object property value is achieved (or times out).
	New Step	F8; INSERT	Inserts a new step in the Keyword View.
	New Step After Block	SHIFT+F8	Inserts a new step after a conditional or loop block in the Keyword View.
	**Operation**		Inserts an operation (function) step in a component.
	**Comment**		Inserts a Comment step in the Keyword View.
	**Report**		Inserts a Reporter step in the Keyword View, instructing QuickTest to report an event to the Test Results.
	Conditional Statement		Inserts an **If...Then**, **ElseIf...Then**, or **Else** statement according to your selection.

	Command	Shortcut Key	Function
	Loop Statement		Inserts a **While...Wend**, **For...Next**, **Do...While**, or **Do...Until** statement according to your selection.
	Call to New Action		Creates a new action and inserts it in the specified location.
	Call to Copy of Action		Inserts a call to an editable copy of an existing action.
	Call to Existing Action		Inserts a call to an existing reusable action.
	Call to WinRunner		Inserts a call to a WinRunner test or user-defined function. (Available only if WinRunner is installed on the QuickTest computer)
	**Start Transaction**		Inserts a StartTransaction step in the test, marking the beginning of the transaction to be timed. (Relevant only if the test includes transactions to be used by LoadRunner or Business Availability Center)
	**End Transaction**		Inserts an EndTransaction step in the test, marking the end of the transaction to be timed. (Relevant only if the test includes transactions to be used by LoadRunner or Business Availability Center)

Some of the **Insert** menu commands are also available from the **Insert** toolbar (described on page 45).

Automation Menu Commands

You can manage your record and run sessions using the following **Automation** menu commands:

	Command	Shortcut Key	Function
	**Record**	F3	Starts a recording session.
	**Run**	F5	Starts a run session from the beginning or from the line at which the session was paused.
	**Stop**	F4 (You can also define a shortcut key or key combination. See "Setting Run Testing Options" on page 1253.)	Stops the recording or run session.
	Run Current Action		Runs only the active action.
	**Run from Step**	CTRL+F5	Starts a run session from the selected step.
	**Maintenance Run Mode**		Starts a run session during which the Maintenance Run Mode wizard opens for steps that failed because an object was not found in the application (if applicable).
	**Update Run Mode**		Starts a run session to update test object descriptions and other options (if applicable).
	**Analog Recording**	SHIFT+ALT+F3	Starts recording in **Analog Recording** mode.
	**Low Level Recording**	CTRL+SHIFT +F3	Starts recording in **Low Level Recording** mode.

	Command	Shortcut Key	Function
	Record and Run Settings		Opens the Record and Run Settings dialog box, enabling you to define browser preferences for recording and running your test.
	Process Guidance List		Lists the processes that are available for the current document type and for the currently loaded QuickTest add-ins, enabling you to open them.
	**Results**		Opens the Test Results viewer, enabling you to view results for a test run session.

Some of the **Automation** menu commands are also available from the **Automation** toolbar (described on page 44).

Resources Menu Commands

You can manage your object repositories and other resources using the following **Resources** menu commands:

	Command	Shortcut Key	Function
	**Object Repository**	CTRL+R	Opens the Object Repository window, which displays a tree containing all objects in the current test or component.
	**Object Repository Manager**		Opens the Object Repository Manager dialog box, enabling you to open and modify multiple shared object repositories.
	**Associate Repositories**		Opens the Associate Repositories dialog box, enabling you to manage the object repository associations for the test.
	**Map Repository Parameters**		Opens the Map Repository Parameters dialog box, enabling you to map repository parameters, as needed.
	Recovery Scenario Manager		Opens the Recovery Scenario Manager dialog box.
	Associated Function Libraries		Lists the function libraries associated with the active document, enabling you to open them.

The **Object Repository** menu command is also available from the **Automation** toolbar (described on page 44).

Debug Menu Commands

You can debug the steps in your test and any associated function library using the following **Debug** menu commands:

	Command	Shortcut Key	Function
	**Pause**		Pauses the debug session.
	**Step Into**	F11	Runs only the current line of the script. If the current line calls a method, the method is displayed in the view but is not performed.
	**Step Over**	F10	Runs only the current line of the script. When the current line calls a method, the method is performed in its entirety, but is not displayed in the view.
	**Step Out**	SHIFT+F11	Runs to the end of the method then pauses the run session. (Available only after running a method using **Step Into**)
	Run to Step	CTRL+F10	Runs until the current step.
	Debug from Step		Runs from the selected step instead of the start of the test.
	Add to Watch	CTRL+T	Adds the selected item to the Watch tab.
	**Insert/Remove Breakpoint**	F9	Sets or clears a breakpoint in the test.
	**Enable/Disable Breakpoint**	CTRL+F9	Enables or disables a breakpoint in the test.
	**Clear All Breakpoints**	CTRL+ SHIFT+F9	Deletes all breakpoints in the test.
	**Enable/Disable All Breakpoints**		Enables or disables all breakpoints in the test.

Some of the **Debug** commands are also available from the **Debug** toolbar (described on page 45).

Tools Menu Commands

You can perform the following **Tools** menu commands:

	Command	Shortcut Key	Function
	**Options**		Opens the Options dialog box, enabling you to modify global testing options.
	View Options		Opens the Editor Options dialog box, enabling you to customize how tests and function libraries are displayed in the Expert View and function library windows.
	**Check Syntax**	CTRL+F7	Checks the syntax of the active document.
	Object Identification		Opens the Object Identification dialog box, enabling you to specify how QuickTest identifies a particular test object.
	**Object Spy**		Opens the Object Spy dialog box, enabling you to view the native properties and operations of any object in an open application, as well as the test object hierarchy, identification properties, and test object operations that QuickTest uses to represent that object.
	Web Event Recording Configuration		Opens the Web Event Recording Configuration dialog box, enabling you to specify a recording configuration level.

	Command	Shortcut Key	Function
	Data Driver		Opens the Data Driver dialog box, which displays the default Constants list for the action.
	Change Active Screen		Replaces the previously recorded Active Screen with the selected Active Screen.
	Virtual Objects > New Virtual Object		Opens the Virtual Object Wizard, enabling you to teach QuickTest to recognize an area of your application as a standard test object.
	Virtual Objects > Virtual Object Manager		Opens the Virtual object Manager, enabling you to manage all of the virtual object collections defined on your computer.
	Customize		Opens the Customize dialog box, which enables you to customize toolbars and menus, and create new menus.

Some of the **Tools** menu commands are also available from the **Tools** toolbar (described on page 45).

Window Menu Commands

You can perform the following **Window** menu commands:

Command	Function
Cascade	Displays the open documents cascaded.
Tile Horizontally	Displays the open documents one above the other.
Tile Vertically	Displays the open documents side-by-side.
Close All Function Libraries	Closes all open function libraries.
open files section	Lists the documents that are currently open in the QuickTest session.
Windows	Opens the Windows dialog box, enabling you to manage your open document windows.

Help Menu Commands

You can perform the following **Help** menu commands:

Command	Shortcut Key	Function
QuickTest Professional Help	F1	Opens the QuickTest Professional Help.
Printer-Friendly Documentation		Opens a page that provides links to printer-friendly versions of all QuickTest documentation, in Adobe Acrobat Reader (PDF) format.
QuickTest Professional Tutorial		Opens the QuickTest Professional tutorial, which teaches you basic QuickTest skills and shows you how to start testing your applications.
What's New		Opens the What's New in QuickTest Professional Help.

Command	Shortcut Key	Function
Product Feature Movies		Enables you to view movies illustrating various QuickTest features.
Troubleshooting & Knowledge Base		Opens the Troubleshooting area of the HP Software Support Web site, enabling you to select from several self-help troubleshooting options, including a product-specific knowledge base articles. (Requires login.) The URL is: http://h20230.www2.hp.com/troubleshooting.jsp
HP Software Support		Opens the HP Software Support Web site. This site enables you to browse the HP Support Knowledge Base and add your own articles. You can also post to and search user discussion forums, submit support requests, download patches and updated documentation, and more. The URL is: www.hp.com/go/hpsoftwaresupport
Send Feedback and Win!		Opens the HP QuickTest Professional Send Feedback and Win Web site, where you can answer surveys about QuickTest and become eligible to win prizes in special prize drawings. The URL is: http://www.hpqtp.com
Check for Updates		Checks online for any available updates to QuickTest Professional. You can choose which updates you want to download and (optionally) install.

Command	Shortcut Key	Function
HP QuickTest Professional Software Web Page		Uses your default Web browser to access the HP QuickTest Professional software Web page within the HP corporate Web site. This site provides you with overview information, data sheets, demos and white papers about QuickTest as well as access to other technical resources. The URL is: https://h10078.www1.hp.com/cda/hpms/display/main/hpms_content.jsp?zn=bto&cp=1-11-127-24^1352_4000_100__
About QuickTest Professional		Displays information about the installed version of QuickTest Professional.

Data Table Menu Commands

You can perform the following **Data Table** menu commands by right-clicking in a Data Table cell or pressing the corresponding shortcut keys when one or more cells are selected in the Data Table.

Command	Shortcut Key	Function
Edit > **Cut**	CTRL+X	Cuts the table selection and puts it on the Clipboard.
Edit > **Copy**	CTRL+C	Copies the table selection and puts it on the Clipboard.
Edit > **Paste**	CTRL+V	Pastes the contents of the Clipboard to the current table selection.
Edit > **Clear** > **Contents**	CTRL+DEL	Clears the contents from the current selection.

Command	Shortcut Key	Function
Edit > **Insert**	CTRL+I	Inserts empty cells at the location of the current selection. Cells adjacent to the insertion are shifted to make room for the new cells.
Edit > **Delete**	CTRL+K	Deletes the current selection. Cells adjacent to the deleted cells are shifted to fill the space left by the vacated cells.
Edit > **Fill Right**	CTRL+R	Copies data in the left-most cell of the selected range to all cells to the right of it, within the selected range.
Edit > **Fill Down**	CTRL+D	Copies data in the top cell of the selected range to all cells below it within the selected range.
Edit > **Find**	CTRL+F	Finds a cell containing specified text. You can search the table by row or column and specify to match case or find entire cells only.
Edit > **Replace**	CTRL+H	Finds a cell containing specified text and replaces it with different text. You can search the table by row or column and specify to match case and/or to find entire cells only. You can also replace all.
Data > **Recalc**	F9	Recalculates the selected data in the Data Table.
Switch between Data Table sheets	CTRL+PAGE UP/PAGE DOWN	Switches through the Data Table sheets when the Data Table is in focus.

Other QuickTest Commands

You can perform the following special options using shortcut keys:

Option	Shortcut Key	Function
Switch between Keyword View and Expert View	CTRL+PAGE UP/PAGE DOWN	Toggles between the Keyword View and Expert View.
Switch between open documents	CTRL+TAB	Changes the display to another open document type.
Open context menu	SHIFT+F10, or press the Application Key () *[Microsoft Natural Keyboard only]*	Opens the context menu for the selected step data cell in the Data Table.
Expand all branches	* *[on the numeric keypad]*	Expands all branches in the Keyword View.
Expand branch	+ *[on the numeric keypad]*	Expands the selected item branch and all branches below it in the Keyword View.
Collapse branch	- *[on the numeric keypad]*	Collapses the selected item branch and all branches below it in the Keyword View.
Open the **Item** or **Operation** list	SHIFT+F4 or SPACE, when the **Item** or **Operation** column is selected in the Keyword View.	Opens the **Item** or **Operation** list in the Keyword View, when the **Item** or **Operation** column is selected.

Browsing the QuickTest Professional Program Folder

After the QuickTest Professional setup process is complete, the following items are added to your QuickTest Professional program folder (**Start** > **Programs** > **QuickTest Professional**).

Note: If you uninstalled a previous version of QuickTest Professional before installing this version, you may have additional (outdated) items in your QuickTest Professional program folder. In addition, if you have QuickTest Professional add-ins or extensibility SDKs installed, you may have items in your program folder that relate specifically to these items.

- **Documentation.** Provides the following links to commonly used documentation files:
  - **Printer-Friendly Documentation.** Opens a page that provides links to printer-friendly versions of all QuickTest documentation, in Adobe Acrobat Reader (PDF) format.
  - **QuickTest Automation Reference.** Opens the QuickTest Professional Automation Object Model Reference. The automation object model assists you in automating test management, by providing objects, methods and properties that enable you to control QuickTest features and configurations. The Automation Object Model Reference provides syntax, descriptive information, and examples for the objects, methods, and properties. It also contains a detailed overview to help you get started writing QuickTest automation scripts.
  - **QuickTest Professional Code Samples Plus.** Opens the QuickTest Professional Code Samples Plus Help, which provides sample function libraries, code, and SDK samples with accompanying explanations.

- **QuickTest Professional Help.** Opens a comprehensive help file containing the *HP QuickTest Professional User Guide*, the *HP QuickTest Professional for Business Process Testing User Guide*, the *HP QuickTest Professional Add-ins Guide*, the *HP QuickTest Professional Object Model Reference* (including the relevant sections for any installed add-ins), *QuickTest Advanced References* (Automation API and XML Schema references), and the *Microsoft VBScript Reference*.
- **Tutorial.** Opens the QuickTest Professional tutorial, which teaches you basic QuickTest skills and shows you how to start testing your applications.

- **Extensibility**. Provides links to the Help for the add-in Extensibility SDKs available with QuickTest Professional 10.00. If you install an extensibility SDK, this program folder may also contain additional items.
- **Sample Applications.** Contains the following links to sample applications that you can use to practice testing with QuickTest:
  - **Flight.** Opens a sample flight reservation Windows application. To access the application, enter any username and the password **mercury**.
  - **Mercury Tours Web site.** Opens a sample flight reservation Web application. This Web application is used as a basis for the QuickTest tutorial. For more information, see the *HP QuickTest Professional Tutorial*.
- **Tools.** Contains the following utilities and tools that assist you with the testing process:

Note: There may be additional tools depending on the installed QuickTest add-ins.

  - **Additional Installation Requirements.** Opens the Additional Installation Requirements dialog box, which displays any prerequisite software that you must install or configure to work with QuickTest.
  - **HP Micro Player.** Opens the HP Micro Player, which enables you to view captured movies of a run session without opening QuickTest. For more information, click the **Help** button in the HP Micro Player window.

- **License Validation Utility.** Opens the License Validation utility, which enables you to retrieve and validate license information. For more information, click the **Help** button in the License Validation Utility window.
- **Password Encoder.** Opens the Password Encoder dialog box, which enables you to encode passwords. You can use the resulting strings as method arguments or Data Table parameter values (tests only). For more information, see "Inserting Encoded Passwords into Method Arguments and Data Table Cells" on page 406.
- **QuickTest Script Editor.** Opens the QuickTest Script Editor, which enables you to open and modify the scripts of multiple tests and function libraries, simultaneously. For more information, see "Working with the QuickTest Script Editor" on page 1381.
- **Register New Browser Control.** Opens the Register Browser Control Utility, which enables you to register your browser control application so that QuickTest Professional recognizes your Web object when recording or running tests. For more information, see the section on registering browser controls in the *HP QuickTest Professional Add-ins Guide*.
- **Remote Agent.** Activates the QuickTest Remote Agent, which enables you to configure how QuickTest behaves when a test is run by a remote application such as Quality Center. For more information, see "Enabling Quality Center to Run Tests on a QuickTest Computer" on page 1440.
- **Save and Restore Settings**. Opens the Save and Restore Settings dialog box, which enables you to save certain existing configurations before uninstalling a QuickTest 9.2 or older version, and then restore them after installing a new version. For more information, see "Save and Restore Settings" on page 1619.
- **Silent Test Runner.** (Relevant only for tests) Opens the Silent Test Runner dialog box, which enables you to run a QuickTest test the way it is run from LoadRunner and Business Availability Center. For more information, see "Using Silent Test Runner" on page 1538.
- **Test Batch Runner.** (Relevant only for tests) Opens the Test Batch Runner dialog box, which enables you to set up QuickTest to run several tests in succession. For more information, see "Running a Test Batch" on page 966.

- **Test Results Deletion Tool.** Opens the Test Results Deletion Tool dialog box, which enables you to delete unwanted or obsolete results from your system according to specific criteria that you define. For more information, see "Deleting Results Using the Test Results Deletion Tool" on page 1004.

- **Check for Updates.** Checks online for any available updates to QuickTest Professional. You can choose which updates you want to download and (optionally) install.

- **QuickTest Professional.** Opens the QuickTest Professional application.

- **Readme.** Opens the *HP QuickTest Professional Readme*, which provides the latest news and information on QuickTest Professional and the QuickTest Professional add-ins.

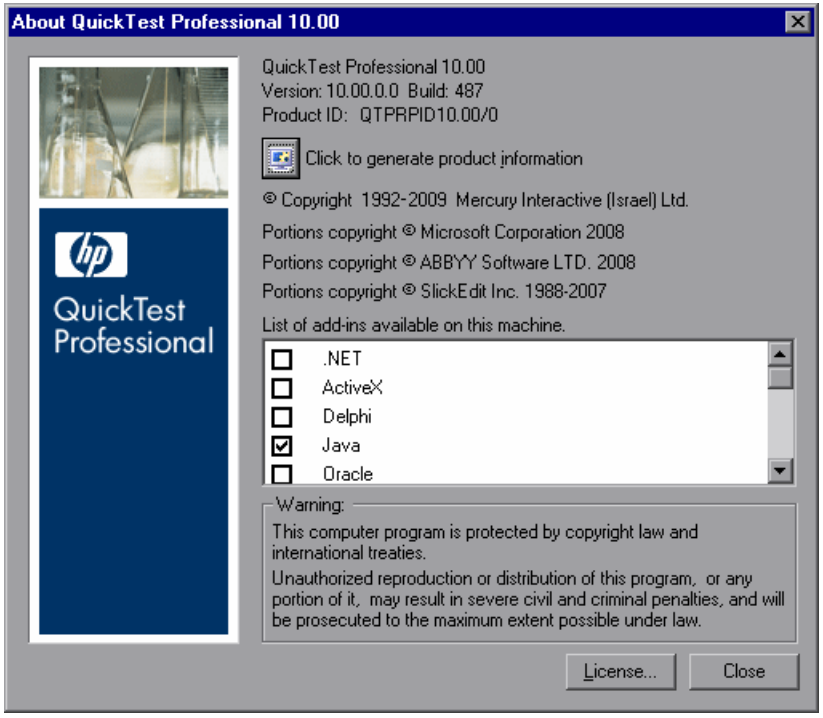
- **Test Results Viewer.** Opens the Test Results window, which enables you to select a test and view information about the steps performed during the run session. For more information, see "The Test Results Window" on page 971.

Viewing Product Information

You can view information regarding the QuickTest add-ins, hotfixes, and patches installed on your computer, as well as other basic information about your computer. This information is useful for troubleshooting and when working with HP Software Support.

To view the product information:

1 In QuickTest, select **Help** > **About QuickTest Professional**. The About QuickTest Professional dialog box opens.



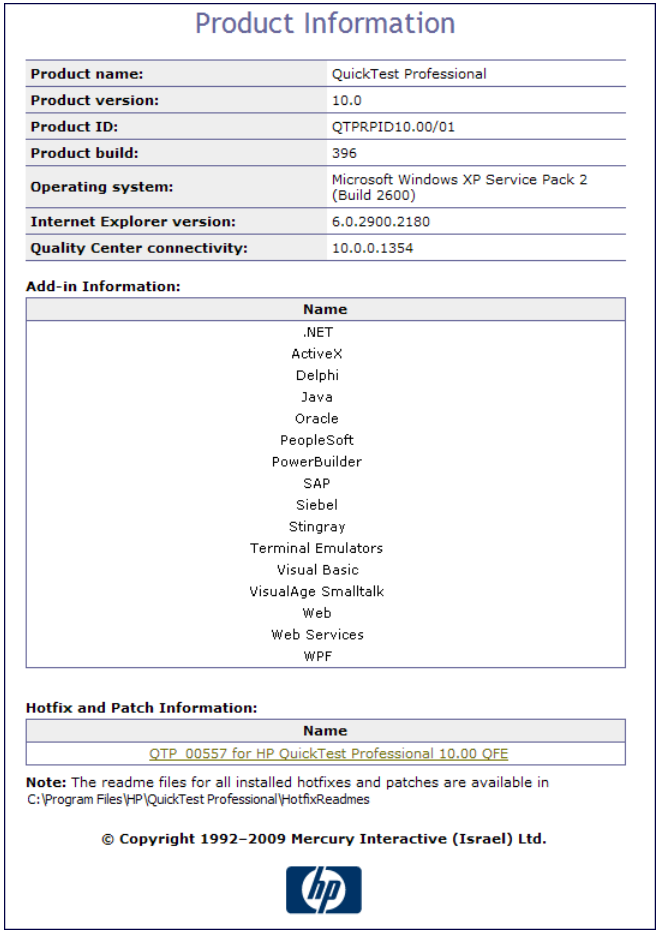
The About QuickTest Professional window displays the following information:

- The version of QuickTest that is installed on your computer, its build number, and Product ID number.
- The list of QuickTest add-ins that are installed on your computer. A check mark next to the add-in name indicates that the add-in is currently loaded. For more information on QuickTest add-ins, see the *HP QuickTest Professional Add-ins Guide*.

Tip: To view details for, or modify, the QuickTest Professional licenses installed on your computer, click the **License** button. For more information, see the *HP QuickTest Professional Installation Guide*.

---

2 To view more detailed information on the QuickTest Professional products installed on your computer, click the **Product Information** button. The Product Information window opens.



Product Information

Product name:	QuickTest Professional
Product version:	10.0
Product ID:	QTPRPID10.00/01
Product build:	396
Operating system:	Microsoft Windows XP Service Pack 2 (Build 2600)
Internet Explorer version:	6.0.2900.2180
Quality Center connectivity:	10.0.0.1354

Add-in Information:

Name
.NET
ActiveX
Delphi
Java
Oracle
PeopleSoft
PowerBuilder
SAP
Siebel
Stingray
Terminal Emulators
Visual Basic
VisualAge Smalltalk
Web
Web Services
WPF

Hotfix and Patch Information:

Name
QTP_00557 for HP QuickTest Professional 10.00 QFE

Note: The readme files for all installed hotfixes and patches are available in C:\Program Files\HP\QuickTest Professional\HotfixReadmes

The Product Information window displays the following information:

- The QuickTest Professional version, product ID, and build numbers installed on your computer.
- **Operating system.** The operating system version installed on your computer.
- **Internet Explorer version.** The version of Microsoft Internet Explorer installed on your computer.
- **Quality Center connectivity.** The version of the Quality Center connectivity add-in installed on your computer.
- **Add-in Information.** The QuickTest add-ins installed on your computer.
- **Hotfix and Patch Information.** The names of any QuickTest hotfixes or patches installed on your computer, and links to their readme files.

Part II

Working with Test Objects

3

Understanding the Test Object Model

This chapter describes how QuickTest learns and identifies objects in your application, explains the concepts of **test object** and **run-time object**, object repositories types, and explains how to view the available methods for an object and the corresponding syntax. With the help of this information, you can add statements to your script in the Expert View or use test objects and methods in your functions.

This chapter includes:

About Understanding the Test Object Model

QuickTest tests your dynamically changing application by learning and identifying test objects and their expected properties and values. To do this, QuickTest analyzes each object in your application in much the same way that a person would look at a photograph and remember its details.

The following sections introduce the concepts related to the test object model and describe how QuickTest uses the information it gathers to test your application.

Understanding How QuickTest Learns Objects

QuickTest learns objects just as you would. For example, suppose as part of an experiment, Alex is told that he will be shown a photograph of a picnic scene for a few seconds during which someone will point out one item in the picture. Alex is told that he will be expected to identify that item again in identical or similar pictures one week from today.

Before he is shown the photograph, Alex begins preparing himself for the test by thinking about which characteristics he wants to learn about the item that the tester indicates. Obviously, he will automatically note whether it is a person, inanimate object, animal, or plant. Then, if it is a person, he will try to commit to memory the gender, skin color, and age. If it is an animal, he will try to remember the type of animal, its color, and so forth.

The tester shows the scene to Alex and points out one of three children sitting on a picnic blanket. Alex notes that it is a Caucasian girl about 8 years old. In looking at the rest of the picture, however, he realizes that one of the other children in the picture could also fit that description. In addition to learning his planned list of characteristics, he also notes that the girl he is supposed to identify has long, brown hair.

Now that only one person in the picture fits the characteristics he learned, he is fairly sure that he will be able to identify the girl again, even if the scene the tester shows him next week is slightly different.

Since he still has a few moments left to look at the picture, he attempts to notice other, more subtle differences between the child he is supposed to remember and the others in the picture—just in case.

If the two similar children in the picture appeared to be identical twins, Alex might also take note of some less permanent feature of the child, such as the child's position on the picnic blanket. That would enable him to identify the child if he were shown another picture in which the children were sitting on the blanket in the same order.

QuickTest uses a very similar method when it learns objects.

First, it "looks" at the object being learned and stores it as a **test object**, determining in which test object class it fits. In the same way, Alex immediately checked whether the item was a person, animal, plant, or inanimate object. QuickTest might classify the test object as a standard Windows dialog box (Dialog), a Web button (WebButton), or a Visual Basic scroll bar object (VbScrollBar), for example.

Then, QuickTest "considers" the **identification properties** for the test object. For each test object class, QuickTest has a list of **mandatory** properties that it always learns; similar to the list of characteristics that Alex planned to learn before seeing the picture. When QuickTest learns an object, it always learns these default property values, and then "looks" at the rest of the objects on the page, dialog box, or other parent object to check whether this **description** is enough to uniquely identify the object. If not, QuickTest adds **assistive** properties, one by one, to the description, until it has compiled a unique description; similar to when Alex added the hair length and color characteristics to his list. If no assistive properties are available, or if those available are not sufficient to create a unique description, QuickTest adds a special **ordinal identifier**, such as the object's location on the page or in the source code, to create a unique description, just as Alex would have remembered the child's position on the picnic blanket if two of the children in the picture had been identical twins.

Understanding How QuickTest Identifies Objects During the Run Session

QuickTest also uses a very human-like technique for identifying objects during the run session.

Suppose as a continuation to the experiment, Alex is now asked to identify the same "item" he initially identified but in a new, yet similar environment.

The first photograph he is shown is the original photograph. He searches for the same Caucasian girl, about eight years old, with long, brown hair that he was asked to remember and immediately picks her out. In the second photograph, the children are playing on the playground equipment, but Alex is still able to easily identify the girl using the same criteria.

Similarly, during a run session, QuickTest searches for a **run-time object** that exactly matches the description of the test object it learned previously. It expects to find a perfect match for both the mandatory and any assistive properties it used to create a unique description while learning the object. As long as the object in the application does not change significantly, the description learned is almost always sufficient for QuickTest to uniquely identify the object. This is true for most objects, but your application could include objects that are more difficult to identify during subsequent run sessions.

Consider the final phase of Alex's experiment. In this phase, the tester shows Alex another photograph of the same family at the same location, but the children are older and there are also more children playing on the playground. Alex first searches for a girl with the same characteristics he used to identify the girl in the other pictures (the test object), but none of the Caucasian girls in the picture have long, brown hair. Luckily, Alex was smart enough to remember some additional information about the girl's appearance when he first saw the picture the previous week. He is able to pick her out (the run-time object), even though her hair is now short and dyed blond.

How is he able to do this? First, he considers which features he knows he must find. Alex knows that he is still looking for a Caucasian female, and if he were not able to find anyone that matched this description, he would assume she is not in the photograph.

After he has limited the possibilities to the four Caucasian females in this new photograph, he thinks about the other characteristics he has been using to identify the girl—her age, hair color, and hair length. He knows that some time has passed and some of the other characteristics he remembers may have changed, even though she is still the same person.

Thus, since none of the Caucasian girls have long, dark hair, he ignores these characteristics and searches for someone with the eyes and nose he remembers. He finds two girls with similar eyes, but only one of these has the petite nose he remembers from the original picture. Even though these are less prominent features, he is able to use them to identify the girl.

QuickTest uses a very similar process of elimination with its **Smart Identification** mechanism to identify an object, even when the learned description is no longer accurate. Even if the values of your identification properties change, QuickTest maintains your test's reusability by identifying the object using Smart Identification. For more information on Smart Identification, see Chapter 4, "Configuring Object Identification."

The remainder of this guide assumes familiarity with the concepts presented here, including test objects, run-time objects, object properties, mandatory and assistive properties, and Smart Identification. An understanding of these concepts will enable you to create well-designed, functional tests for your application.

Applying the Test Object Model Concept

The test object model is a large set of object types or classes that QuickTest uses to represent the objects in your application. Each test object class has a list of identification properties that QuickTest can learn about the object, a sub-set of these properties that can uniquely identify objects of that class, and a set of relevant operations that QuickTest can perform on the object.

A **test object** is an object that QuickTest creates in the test to represent the actual object in your application. QuickTest stores information on the object that will help it identify and check the object during the run session.

A **run-time object** is the actual object in your application on which methods are performed during the run session.

When QuickTest learns an object in your application, it adds the corresponding test object to an **object repository**, which is a storehouse for objects. You can add test objects to an object repository in several ways. For example, you can use the QuickTest Navigate and Learn option, add test objects manually, or perform an operation on your application while recording. For more information on object repositories, see Chapter 5, "Managing Test Objects in Object Repositories", Chapter 7, "Managing Object Repositories" and Chapter 12, "Creating Tests Using the Keyword-Driven Methodology".

When you add an object to an object repository, QuickTest:

- Identifies the QuickTest test object class that represents the learned object and creates the appropriate test object.
- Reads the current value of the object's properties in your application and stores the list of **identification properties** and values with the test object.
- Chooses a unique name for the test object, generally using the value of one of its prominent properties.

For example, suppose you add a **Search** button with the following HTML source code:

```
<INPUT TYPE="submit" NAME="Search" VALUE="Search">
```

QuickTest identifies the object as a **WebButton** test object. In the object repository, QuickTest creates a WebButton object with the name Search, learns a set of identification properties for the object, and decides to use the following properties and values to uniquely identify the **Search** WebButton:

Name	Value
Description properties	
type	submit
name	Search
html tag	INPUT

If you add an object to an object repository by recording on your application, QuickTest records the operation that you performed on the object using the appropriate QuickTest test object method. For example, QuickTest records that you performed a **Click** method on the WebButton.

QuickTest displays your step in the Keyword View like this:

Item	Operation	Documentation
Action1		
Search Results: Search		
Search Results: Search		
Search	Click	Click the "Search" button.

QuickTest displays your step in the Expert View as follows:

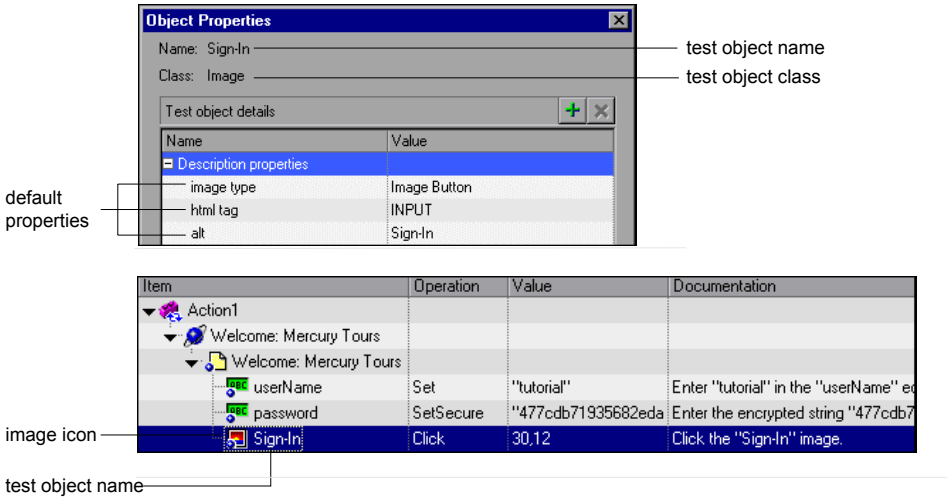
```
Browser("Search Results: Search").Page("Search Results:
    Search").WebButton("Search").Click
```

When you run a test, QuickTest identifies each object in your application by its test object class and its **description** (the set of identification properties and values used to uniquely identify the object). The list of test objects and their properties and values are stored in the object repository. In the above example, QuickTest would search in the object repository during the run session for the WebButton object with the name Search to look up its description. Based on the description it finds, QuickTest would then look for a WebButton object in the application with the HTML tag INPUT, of type submit, with the value Search. When it finds the object, it performs the **Click** method on it.

Understanding Test Object Descriptions

For each test object class, QuickTest learns a set of identification properties when it learns an object, and selects a sub-set of these properties to serve as a unique object description. QuickTest then uses this description to identify the object when it runs the test.

For example, by default, QuickTest learns the image type (such as plain image or image button), the **html tag**, and the **Alt** text of each Web image it learns.



If these three mandatory property values are not sufficient to uniquely identify the object within its parent object, QuickTest adds some assistive properties and/or an ordinal identifier to create a unique description.

When the test runs, QuickTest searches for the object that matches the description it learned. If it cannot find any object that matches the description, or if it finds more than one object that matches, QuickTest may use the Smart Identification mechanism to identify the object.

You can configure the mandatory, assistive, and ordinal identifier properties that QuickTest uses to learn the descriptions of the objects in your application, and you can enable and configure the Smart Identification mechanism. For more information, see Chapter 4, "Configuring Object Identification."

Understanding Test Object and Native Properties and Operations

The identification property set for each test object is created and maintained by QuickTest. The native property set for each run-time object is created and maintained by the object creator (for example, Microsoft for Microsoft Internet Explorer objects, Netscape for Netscape Browser objects, the product developer for ActiveX objects, and so on).

Similarly, a test object operation is a method or property that QuickTest recognizes as applicable to a particular test object class. For example, the **Click** method is applicable to a WebButton test object. As you add steps to your test, you specify which operation to perform on each test object. If you record steps, QuickTest records the relevant operation as it is performed on an object.

During a run session, QuickTest performs the specified test object operation on the run-time object. Native operations are the methods of the object in your application as defined by the object creator.

Property values of objects in your application may change dynamically each time your application opens, or based on certain conditions. You may need to modify the identification property values to match the native property values. You can modify identification properties manually while designing your test, or use **SetTOProperty** statements during a run session. You can also use regular expressions to identify property values based on conditions or patterns you define, or you can parameterize property values with Data Table parameters so that a different value is used during each iteration of the test. For more information on modifying object properties, see Chapter 5, "Managing Test Objects in Object Repositories." For more information on parameterization, see Chapter 24, "Parameterizing Values." For more information on regular expressions, see "Understanding and Using Regular Expressions" on page 762.

You can view or modify the identification property values that are stored with your test in the Object Properties or Object Repository dialog box. For more information, see "Specifying or Modifying Property Values" on page 163.

You can view the current identification property values of any object on your desktop using the Properties tab of the Object Spy. For more information, see "Viewing Object Properties and Operations Using the Object Spy" on page 97.

You can view the syntax of the test object operations as well as the native operations of any object on your desktop using the Operations tab of the Object Spy. For more information, see "Viewing Object Properties and Operations Using the Object Spy" on page 97.

You can retrieve or modify property values of the test object during the run session by adding **GetTOProperty** and **SetTOProperty** statements in the Keyword View or Expert View. You can retrieve property values from the run-time object during the run session by adding **GetROProperty** statements. For more information, see "Retrieving and Setting Identification Property Values" on page 886.

If the available test object operations and identification properties for a test object do not provide the functionality you need, you can access the internal operations and properties of the run-time object using the **Object** property. You can also use the **attribute** object property to identify Web objects in your application according to user-defined properties. For information, see "Accessing Native Properties and Operations" on page 887.

For more information on test object operations and identification properties, see the *HP QuickTest Professional Object Model Reference*.

Understanding Object Repository Types

Objects can be stored in two types of object repositories—a shared object repository and a local object repository. A **shared object repository** stores objects in a file that can be accessed by multiple tests (in read-only mode). A **local object repository** stores objects in a file that is associated with one specific action, so that only that action can access the stored objects.

When you plan and create tests, you must consider how you want to store the objects in your tests. You can store the objects for each action in its corresponding local object repository, or you can store the objects in your tests in one or more shared object repositories. By storing objects in shared object repositories and associating these repositories with your actions, you enable multiple actions to use the objects. For each action, you can use a combination of objects from your local and shared object repositories, according to your needs. You can also transfer local objects to a shared object repository, if required. This reduces maintenance and enhances the reusability of your tests because it enables you to maintain the objects in a single, shared location instead of multiple locations. For more information, see "Deciding Whether to Use Local or Shared Object Repositories" on page 92.

If you are new to using QuickTest, you may want to use local object repositories. In this way, you can record and run tests without creating, choosing, or modifying shared object repositories because all objects are automatically saved in a local object repository that can be accessed by its corresponding action. If you modify an object in the local object repository, your changes do not have any effect on any other action or any other test (except tests that call the action, as described in "Inserting Calls to Existing Actions" on page 464).

If you are familiar with testing, it is probably most efficient to save objects in a shared object repository. In this way, you can use the same shared object repository for multiple actions—if the actions include the same objects. Object information that applies to many actions is kept in one central location. When the objects in your application change, you can update them in one location for all the actions that use this shared object repository.

If an object with the same name is located in both the local object repository and in a shared object repository associated with the same action, the action uses the local object definition. If an object with the same name is located in more than one shared object repository associated with the same action, the object definition is used from the first occurrence of the object, according to the order in which the shared object repositories are associated with the action. For more information on associating shared object repositories, see "Associating Object Repositories with Actions" on page 446.

Local objects are saved locally with the action, and can be accessed only from that action. When using a shared object repository, you can use the same object repository for multiple actions. You can also use multiple object repositories for each action.

When you open and work with an existing test, it always uses the object repositories that are specified in the Associated Repositories tab of the Action Properties dialog box or in the Associate Repositories dialog box. Shared object repositories are read-only when accessed from tests; you edit them using the Object Repository Manager.

Note: If you want to use a shared object repository from Quality Center, you must save the shared object repository in the Test Resources module in your Quality Center project before you associate the object repository using the Associated Repositories tab of the Action Properties dialog box or the Associate Repositories dialog box. (You can save the shared object repository to your Quality Center project using the Object Repository Manager.)

Note for users of previous QuickTest versions:

If you open a test stored in the file system that was created using QuickTest 9.0 or earlier, the object repository associations are changed as follows:

- If the test previously used per-action repositories, the objects in each **per-action repository** are transferred to the **local object repository** of each action in the test.
- If the whole test previously used one shared object repository, the same shared object repository is associated with each of the actions in the test, and the actions' local object repositories are empty.

If the test is opened in read-only mode, these changes are not saved.

Deciding Whether to Use Local or Shared Object Repositories

To choose where to save objects, you need to understand the differences between local and shared object repositories.

In general, the local object repository is easiest to use when you are creating simple tests, especially under the following conditions:

- You have only one, or very few, tests that correspond to a given application, interface, or set of objects.
- You do not expect to frequently modify object properties.
- You generally create single-action tests.

Conversely, the shared object repository is generally the preferred option when:

- You are creating tests using keyword-driven methodologies (not by recording).
- You have several tests that test elements of the same application, interface, or set of objects.
- You expect the object properties in your application to change from time to time and/or you regularly need to update or modify object properties.
- You often work with multi-action tests and regularly use the **Insert Copy of Action** and **Insert Call to Action** options.

Understanding the Local Object Repository

When you use a local object repository, QuickTest uses a separate object repository for each action. (You can also use one or more shared object repositories if needed. For more information, see "Understanding the Shared Object Repository" on page 94.) The local object repository is fully editable from within its action.

When working with a local object repository:

- QuickTest creates a new (empty) object repository for each action.
- When QuickTest learns new objects (either because you add them to the local object repository, or you record operations on objects in your application), it automatically stores the information about those objects in the corresponding local object repository (if the test objects do not already exist in an associated shared object repository).

  QuickTest adds all new objects to the local object repository even if one or more shared object repositories are already associated with the action. (This assumes that a object with the same description does not already exist in one of the associated shared object repositories).
- If a child object is added to a local object repository, and its parents are in a shared object repository, its parents are automatically added to the local object repository.
- Every time you create a new action, QuickTest creates a new, corresponding local object repository and adds test objects to the repository as it learn them.
- If QuickTest learns the same object in your application in two different actions, the test object is stored as a separate test object in each of the local object repositories.
- When you save your test, all of the local object repositories are automatically saved with the test (as part of each action within the test). The local object repository is not accessible as a separate file (unlike the shared object repository).

Understanding the Shared Object Repository

When you use shared object repositories, QuickTest uses the shared object repositories you specify for the selected action. You can use one or more shared object repositories. (You can also save some objects in a local object repository for each action if you need to access them only from the specific action. For more information, see "Understanding the Local Object Repository" on page 92.)

After you begin creating your test, you can specify additional shared object repositories. You can also create new ones and associate them with your action. Before running the test, you must ensure that the object repositories being used by the test contain all of the objects in your test. Otherwise, the test may fail. For more information, see "Adding Test Objects to a Local or Shared Object Repository" on page 136.

You modify a shared object repository using the Object Repository Manager. For more information, see Chapter 7, "Managing Object Repositories."

When working with a shared object repository:

- If QuickTest Professional learns a test object that already exists in either the shared or local object repository, QuickTest uses the existing information and does not add the object to that object repository.
- If a child object is added to a local object repository, and its parents are in a shared object repository, its parents are automatically moved to the local object repository.
- When QuickTest learns a test object, it adds it to the local object repository (not the shared object repository)—unless the same test object already exists in an associated shared object repository. (In this case, QuickTest uses the existing information in the shared object repository.)

You can export objects from the local object repository to a shared object repository. You can also export the local object repository and replace it with a shared object repository. This enables you to make the local objects accessible to other actions. For more information, see "Exporting Local Objects to a Shared Object Repository" on page 193.

You can also merge objects from the local object repository directly to a shared object repository that is associated with the same action. This can help reduce maintenance since you can maintain the objects in a single shared location, instead of multiple locations. For more information, see "Updating a Shared Object Repository from Local Object Repositories" on page 269.

The following table lists features and functionality, indicating if they are available in the Object Repository window or the Object Repository Manager:

Functionality	Object Repository window	Object Repository Manager
"Adding Test Objects to a Local or Shared Object Repository" on page 136	✓	✓
"Copying, Pasting, and Moving Objects in the Object Repository" on page 150	✓	✓
"Deleting Objects from the Object Repository" on page 153	✓	✓
"Highlighting an Object in Your Application" on page 157	✓	✓
"Locating a Test Object in the Object Repository" on page 159	✓	✓
"Specifying or Modifying Property Values" on page 163	✓	✓
"Updating Identification Properties from an Object in Your Application" on page 165	✓	✓
"Restoring Default Mandatory Properties for a Test Object" on page 168	✓	✓

Functionality	Object Repository window	Object Repository Manager
"Renaming Test Objects" on page 169	✓	✓
"Adding Properties to a Test Object Description" on page 171	✓	✓
"Defining New Identification Properties" on page 174	✓	✓
"Removing Properties from a Test Object Description" on page 177	✓	✓
"Exporting Local Objects to a Shared Object Repository" on page 193	✓	✗
"Copying an Object to the Local Object Repository" on page 195	✓	✗
"Creating New Object Repositories" on page 217	✗	✓
"Opening Object Repositories" on page 217	✗	✓
"Saving Object Repositories" on page 219	✗	✓
"Closing Object Repositories" on page 221	✗	✓
"Editing Object Repositories" on page 224	✗	✓
"Adding Test Objects to Your Test Using the Object Repository Manager" on page 225	✓	✓
"Adding Test Objects Using the Navigate and Learn Option" on page 225	✗	✓
"Managing Repository Parameters" on page 229	✗	✓
"Adding Repository Parameters" on page 230	✗	✓
"Modifying Repository Parameters" on page 232	✗	✓
"Deleting Repository Parameters" on page 233	✗	✓
"Specifying a Property Value" on page 235	✗	✓
"Locating Test Objects" on page 239	✓	✓
"Performing Merge Operations" on page 240	✗	✓

Functionality	Object Repository window	Object Repository Manager
"Importing from XML" on page 242	✗	✓
"Exporting to XML" on page 243	✗	✓

Viewing Object Properties and Operations Using the Object Spy

Using the Object Spy pointing hand mechanism, you can view the supported properties and operations of any object in an open application. As you move the pointing hand over the objects in the application, their details are displayed in the Object Spy. These details may include the test object's hierarchy tree, its identification properties and values, and the operations associated with the object. For operations, the syntax is also displayed. For information about using the run-time object's operations or retrieving the values of its properties, see "Retrieving and Setting Identification Property Values" on page 886 and "Accessing Native Properties and Operations" on page 887.

In most environments, you can choose to view the identification properties, the native properties, the test object operations, or the native operations.

To view identification properties, native properties, test object operations, or native operations:

1 Open your application to the page containing the object on which you want to spy.

2 Select **Tools** > **Object Spy** or click the **Object Spy** toolbar button to open the Object Spy dialog box.

3 Select the details you want to view for the object. For more information, see "The Object Spy Dialog Box" on page 100.

4 If the objects on which you want to spy have a deep hierarchy, or long property names and values, resize the Object Spy dialog box to view all the information without scrolling.

5 In the Object Spy dialog box, click the pointing hand. QuickTest is hidden. As you move the pointing hand over the objects in your application, the objects are highlighted, and you can view their test object hierarchy and properties or operations in the Object Spy dialog box.

Note: Highlighting the object in the application is supported only in some environments.

For more information on using the pointing hand, see "Tips for Working with the Pointing Hand" on page 99.

6 Hover over an object in your application. The object is highlighted in the application, and the Object Spy displays the corresponding test object, its properties or operations, and the test object hierarchy tree. You can move your mouse from one object to another in your application (without clicking) to view information on each object.

To view different details about the test object in the Object Spy dialog box, hold the left CTRL key and click the relevant options in the dialog box.

To view the properties and operations of another test object currently displayed in the test object hierarchy tree, hold the left CTRL key and select the relevant test object.

7 To capture information about a particular object and its parent objects in the Object Spy, click on the object (in your application). The Object Spy displays the test object hierarchy tree and details for the selected object according to the object details tab and object type radio button that are selected.

After clicking on an object, you can change the selected radio button or tab to view additional details.

To view properties, values, or operations of other test objects currently displayed in the test object hierarchy tree, select that test object in the tree.

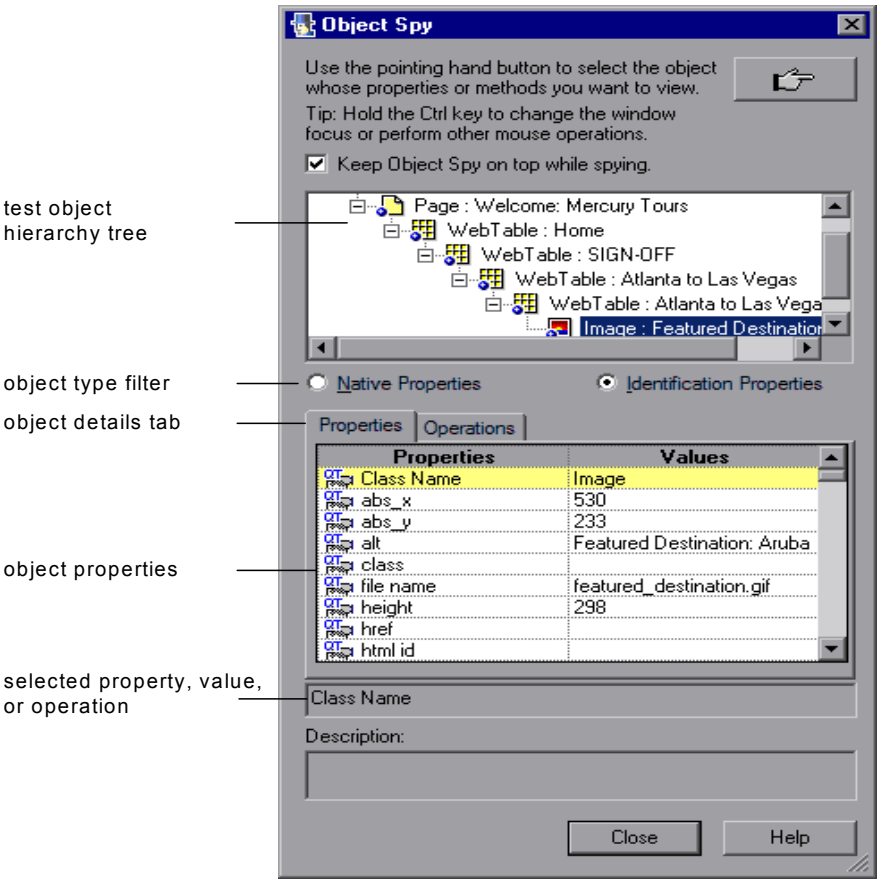
Tips for Working with the Pointing Hand

- 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.

The Object Spy Dialog Box

Description	Enables you to view the native properties and operations of any object in an open application, as well as the test object hierarchy, identification properties, and operations of the test object that QuickTest uses to represent that object.
How to Access	➤ Select the **Tools** > **Object Spy** menu command ➤ Click the **Object Spy** toolbar button  ➤ Press ALT+T+S You can access the Object Spy dialog box using the methods described above, from any of the following locations: ➤ The QuickTest Window (described on page 23) ➤ The Object Repository Window (described on page 183) ➤ The Object Repository Manager (described on page 210)
Learn More	**Conceptual overview**: "Understanding Test Object and Native Properties and Operations" on page 87 **Primary task**: "Viewing Object Properties and Operations Using the Object Spy" on page 97 **Additional related topics**: "Additional References" on page 104

Below is an image of the Object Spy dialog box:



Object Spy Dialog Box Options

Option	Description
	Click the pointing hand button to turn the mouse pointer into a pointing hand. Then use the pointing hand to highlight or click the object whose properties and/or operations you want to view. As you move the pointing hand over the objects in the application, the objects are highlighted in the application (in some environments), and their details are displayed in the Object Spy dialog box. To capture information about a particular object and its parent objects in the Object Spy, click on the object in the application. **See also:** "Tips for Working with the Pointing Hand" on page 99.
Keep Object Spy on top while spying	Select this check box to keep the Object Spy dialog box in view while spying on an object in your application. **Note:** When this check box is cleared, the Object Spy dialog box may potentially be hidden on your screen behind your application. To view the Object Spy dialog box, press the left CTRL key and arrange the windows as needed.
Test object hierarchy tree	Displays the hierarchy of test objects that are related to the object you selected. While an object is highlighted, test object classes are displayed in the tree, but test object names are not. Test object names (such as Atlanta to Las Vegas and Featured Destinations in the image shown above) are displayed only after clicking the object to capture the information in the Object Spy. To view properties, values, or operations for another test object within the displayed tree, select that test object in the tree.

Option	Description
Native Properties / **Native Operations**	Select this option to display the native properties or operations of the run-time object associated with the test object selected in the Object Spy test object hierarchy tree. Note that the label changes depending on whether the **Properties** or **Operations** tab is selected.
Identification Properties / **Test Object Operations**	Select this option to display the identification properties or the test object operations of the test object selected in the Object Spy test object hierarchy tree. Note that the label changes depending on whether the **Properties** or **Operations** tab is selected.
Properties tab	Displays the native properties or the identification properties of the selected object and the values of the properties. ➤ **Properties.** Displays the property names for the test object that is currently selected in the Object Spy test object hierarchy tree, or the run-time object associated with it. ➤ **Values.** Displays the property values for the properties listed in the **Properties** column.
Operations tab	Displays the native operations or test object operations, and their corresponding syntax, for the test object that is currently selected in the Object Spy test object hierarchy tree, or the run-time object associated with it.

Option	Description
Selected property, value, or operation box	**Properties tab:** Displays the property name or value that was most recently clicked. **Operations tab:** Displays the syntax of the most recently clicked operation. **Tip:** To copy the text that is displayed in this box to the Clipboard, highlight the text and press CTRL+C or right-click the highlighted text and select **Copy**.
Description	Provides a description of the most recently clicked property or operation, when available.

Additional References

Related Concepts	➤ "Accessing Native Properties and Operations" on page 887 ➤ "Retrieving and Setting Identification Property Values" on page 886

4

Configuring Object Identification

When QuickTest learns an object, it learns a set of properties and values that uniquely describe the object within the object hierarchy. In most cases, this description is sufficient to enable QuickTest to identify the object during the run session.

If you find that the description QuickTest uses for a certain object class is not the most logical one for the objects in your application, or if you expect that the values of the properties in the object description may change frequently, you can configure the way that QuickTest learns and identifies objects. You can also map user-defined objects to standard test object classes and configure the way QuickTest learns objects from your user-defined object classes.

This chapter includes:

About Configuring Object Identification

QuickTest has a predefined set of properties that it learns for each test object. If these mandatory property values are not sufficient to uniquely identify a learned object, QuickTest can add some assistive properties and/or an ordinal identifier to create a unique description.

Mandatory properties are properties that QuickTest always learns for a particular test object class.

Assistive properties are properties that QuickTest learns only if the mandatory properties that QuickTest learns for a particular object in your application are not sufficient to create a unique description. If several assistive properties are defined for an object class, then QuickTest learns one assistive property at a time, and stops as soon as it creates a unique description for the object. If QuickTest does learn assistive properties, those properties are added to the test object description.

Note: If the combination of all defined mandatory and assistive properties is not sufficient to create a unique test object description, QuickTest also learns the value for the selected ordinal identifier. For more information, see "Selecting an Ordinal Identifier" on page 113.

When you run a test, QuickTest searches for the object that matches the description it learned (without the ordinal identifier). If it cannot find any object that matches the description, or if more than one object matches the description, QuickTest uses the **Smart Identification** mechanism (if enabled) to identify the object. In many cases, a Smart Identification definition can help QuickTest identify an object, if it is present, even when the learned description fails due to changes in one or more property values. The test object description is used together with the ordinal identifier only in cases where the Smart Identification mechanism does not succeed in narrowing down the object candidates to a single object.

You use the Object Identification dialog box (**Tools** > **Object Identification**) to configure the mandatory, assistive, and ordinal identifier properties that QuickTest uses to learn descriptions of the objects in your application, and to enable and configure the Smart Identification mechanism.

The Object Identification dialog box also enables you to configure new user-defined classes and map them to an existing test object class so that QuickTest can recognize objects from your user-defined classes when you run your test.

Understanding the Object Identification Dialog Box

You use the main screen of the Object Identification dialog box to set mandatory and assistive properties, to select the ordinal identifier, and to specify whether you want to enable the Smart Identification mechanism for each test object.

From the Object Identification dialog box, you can also define user-defined object classes and map them to Standard Windows object classes, and you can configure the Smart Identification mechanism for any object displayed in the **Test Object classes** list for a selected environment.

Notes:

- Any changes you make in the Object Identification dialog box have no effect on objects already added to the object repository.
- The learned and Smart Identification properties of certain test objects cannot be configured, for example, the WinMenu, VbLabel, and VbToolbar objects. These objects are therefore not included in the **Test Object classes** list for the selected environment.

For more information, see:

- "Configuring Mandatory and Assistive Properties" on page 108
- "Selecting an Ordinal Identifier" on page 113
- "Enabling and Disabling Smart Identification" on page 118
- "Restoring Default Object Identification Settings for Test Objects" on page 119
- "Generating Automation Scripts for Your Object Identification Settings" on page 120

Configuring Mandatory and Assistive Properties

If you find that the description QuickTest uses for a certain object class is not the most logical one for the objects in your application, or if you expect that the values of the properties currently used in the object description may change, you can modify the mandatory and assistive properties that QuickTest learns when it learns an object of a given class.

During the run session, QuickTest looks for objects that match all properties in the test object description—it does not distinguish between properties that were learned as mandatory properties and those that were learned as assistive properties.

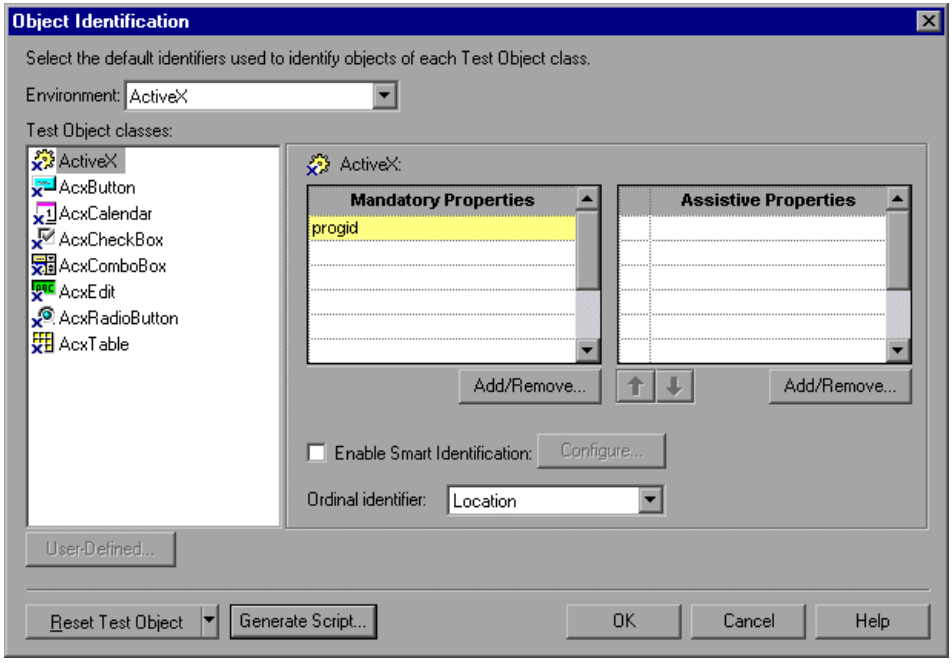
For example, the default mandatory properties for a Web Image object are the **alt**, **html tag**, and **image type** properties. There are no default assistive properties defined. Suppose your Web site contains several space holders for different collections of rotating advertisements. You want to create a test that clicks on the images in each one of these space holders.

However, since each advertisement image has a different **alt** value, one **alt** value would be added when you create the test, and most likely another **alt** value will be captured when you run the test, causing the run to fail. In this case, you could remove the **alt** property from the Web Image mandatory properties list. Instead, since each advertisement image displayed in a certain space holder in your site has the same value for the image **name** property, you could add the **name** property to the mandatory properties to enable QuickTest to uniquely identify the object.

Also, suppose that whenever a Web image is displayed more than once on a page (for example, a logo displayed on the top and bottom of a page), the Web designer adds a special **ID** property to the Image tag. The mandatory properties are sufficient to create a unique description for images that are displayed only once on the page, but you also want QuickTest to learn the **ID** property for images that are displayed more than once on a page. To do this, you add the **ID** property as an assistive property, so that QuickTest learns the **ID** property only when it is necessary for creating a unique test object description.

To configure mandatory and assistive properties for a test object class:

1 Select **Tools** > **Object Identification**. The Object Identification dialog box opens.



2 Select the appropriate environment in the **Environment** list. The test object classes associated with the selected environment are displayed alphabetically in the **Test Object classes** list. (In Standard Windows, the user-defined objects are displayed at the bottom of the list.)

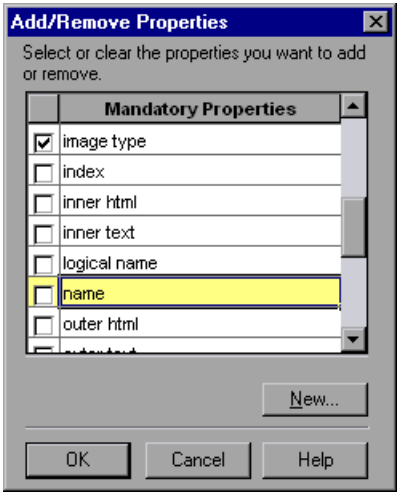
Notes:

- The environments included in the **Environment** list correspond to the loaded add-ins. For more information on loading add-ins, see the section on loading QuickTest add-ins in the *HP QuickTest Professional Add-ins Guide*.
- The **Environment** list might also include additional environments for which you or a third party developed support using add-in extensibility.

---

3 In the **Test Object classes** list, select the test object class you want to configure.

4 In the **Mandatory Properties** list, click **Add/Remove**. The Add/Remove Properties dialog box for mandatory properties opens.



5 Select the properties you want to include in the Mandatory Properties list and/or clear the properties you want to remove from the list.

Note: You cannot include the same property in both the mandatory and assistive property lists.

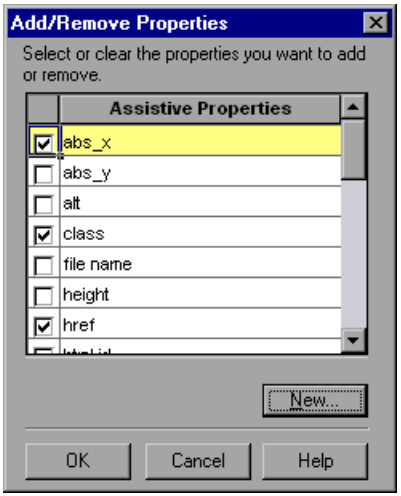
You can specify a new property by clicking **New** and specifying a valid property name in the displayed dialog box.

Tip: You can also add property names to the set of available properties for Web objects using the attribute/<*PropertyName*> notation. To do this, click **New**. The New Property dialog box opens. Enter a valid property using the format attribute/<*PropertyName*> and click **OK**. The new property is added to the **Mandatory Properties** list. For example, to add a property called MyColor, enter attribute/MyColor.

---

6 Click **OK** to close the Add/Remove Properties dialog box. The updated set of mandatory properties is displayed in the **Mandatory Properties** list.

7 In the **Assistive Properties** list, click **Add/Remove**. The Add/Remove Properties dialog box for assistive properties opens.



8 Select the properties you want to include in the assistive properties list and/or clear the properties you want to remove from the list.

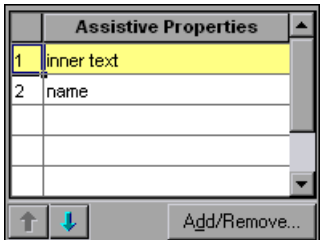
Note: You cannot include the same property in both the mandatory and assistive property lists.

You can specify a new property by clicking **New** and specifying a valid property name in the displayed dialog box.

Tip: You can also add property names to the set of available properties for Web objects using the attribute/*<PropertyName>* notation. To do this, click **New**. The New Property dialog box opens. Enter a valid property in the format attribute/*<PropertyName>* and click **OK**. The new property is added to the **Assistive Properties** list. For example, to add a property called MyColor, enter attribute/MyColor.

---

9 Click **OK** to close the Add/Remove Properties dialog box. The properties are displayed in the Assistive Properties list.



10 Use the up and down arrows to set your preferred order for the assistive properties. When QuickTest learns an object, and assistive properties are necessary to create a unique object description, QuickTest adds the assistive properties to the description one at a time until it has enough information to create a unique description, according to the order you set in the Assistive Properties list.

Selecting an Ordinal Identifier

In addition to learning the mandatory and assistive properties specified in the Object Identification dialog box, QuickTest can also learn a backup ordinal identifier for each test object. The **ordinal identifier** assigns the object a numerical value that indicates its order relative to other objects with an otherwise identical description (objects that have the same values for all properties specified in the mandatory and assistive property lists). This ordered value enables QuickTest to create a unique description when the mandatory and assistive properties are not sufficient to do so.

The assigned ordinal property value is a relative value and is accurate only in relation to the other objects displayed when QuickTest learns an object. Therefore, changes in the layout or composition of your application page or screen can cause this value to change, even though the object itself has not changed in any way. For this reason, QuickTest learns a value for this backup ordinal identifier only when it cannot create a unique description using all available mandatory and assistive properties.

In addition, even if QuickTest learns an ordinal identifier, it will use the identifier during the run session only if the learned description and the Smart Identification mechanism are not sufficient to identify the object in your application. If QuickTest can use other identification properties to identify the object during a run session, the ordinal identifier is ignored.

QuickTest can use the following types of ordinal identifiers to identify an object:

- **Index.** Indicates the order in which the object appears in the application code relative to other objects with an otherwise identical description. For more information, see "Identifying an Object Using the Index Property" on page 115.
- **Location.** Indicates the order in which the object appears within the parent window, frame, or dialog box relative to other objects with an otherwise identical description. For more information, see "Identifying an Object Using the Location Property" on page 115.
- **CreationTime.** (Browser object only.) Indicates the order in which the browser was opened relative to other open browsers with an otherwise identical description. For more information, see "Identifying an Object Using the CreationTime Property" on page 117.

By default, an ordinal identifier type exists for each test object class. To modify the default ordinal identifier, you can select the desired type from the **Ordinal identifier** box.

Tip: While recording, if QuickTest successfully creates a unique test object description using the mandatory and assistive properties, it does not learn an ordinal identifier value. You can add an ordinal identifier to an object's identification properties at a later time using the **Add/Remove** option from the Object Properties or Object Repository dialog box. For more information, see Chapter 5, "Managing Test Objects in Object Repositories."

Identifying an Object Using the Index Property

While learning an object, QuickTest can assign a value to the test object's **Index** property to uniquely identify the object. The value is based on the order in which the object appears within the source code. The first occurrence is 0.

Index property values are object-specific. Therefore, if you use Index:=3 to describe a WebEdit test object, QuickTest searches for the fourth WebEdit object in the page. However, if you use Index:=3 to describe a WebElement object, 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 a page contains 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 following statement refers to the third item in the list, as this is the first WebEdit object on the page with the name UserName:

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

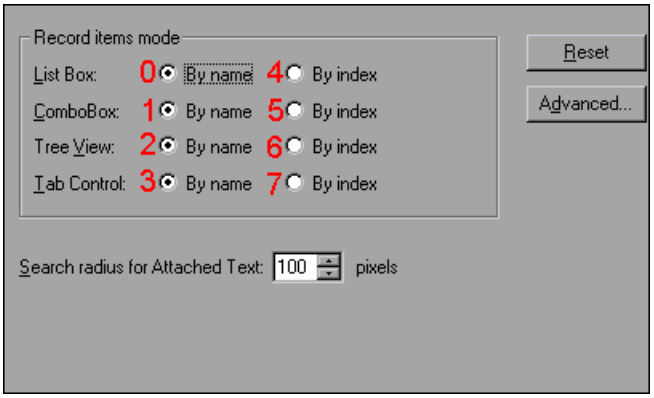
In contrast, the following statement refers to the second item in the list, as that is the first object of any type (WebElement) with the name UserName:

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

Identifying an Object Using the Location Property

While learning an object, QuickTest can assign a value to the test object's **Location** property to uniquely identify the object. The value is based on the order in which the object appears within the window, frame, or dialog box, in relation to other objects with identical properties. The first occurrence of the object is 0. Values are assigned in columns from top to bottom, and left to right.

In the following example, the radio buttons in the dialog box are numbered according to their **Location** property.



Location property values are object-specific. Therefore, if you use Location:=3 to describe a WinButton test object, QuickTest searches from top to bottom, and left to right for the fourth WinButton object in the page. However, if you use Location:=3 to describe a WinObject object, QuickTest searches from top to bottom, and left to right for the fourth standard object on the page—regardless of the type—because the WinObject object applies to all standard objects.

For example, suppose a dialog box contains the following objects:

- A button object with the name OK
- A button object with the name Add/Remove
- A check box object with the name Add/Remove
- A button object with the name Help
- A check box object with the name Check spelling

The following statement refers to the third item in the list, as this is the first check box object on the page with the name Add/Remove.

```
WinCheckBox("Name:=Add/Remove", "Location:=0")
```

In contrast, the following statement, refers to the second item in the list, as that is the first object of any type (WinObject) with the name Add/Remove.

```
WinObject("Name:=Add/Remove", "Location:=0")
```

Identifying an Object Using the CreationTime Property

While learning a browser object, QuickTest assigns a value to the **CreationTime** identification property. This value indicates the order in which the browser was opened relative to other open browsers. The first browser that opens receives the value CreationTime = 0.

During the run session, if QuickTest is unable to identify a browser object based solely on its test object description, it examines the order in which the browsers were opened, and then uses the **CreationTime** property to identify the correct one.

For example, if QuickTest learns three browsers that are opened at 9:01 pm, 9:03 pm, and 9:05 pm, QuickTest assigns the CreationTime values, as follows: CreationTime = 0 to the 9:01 am browser, CreationTime = 1 to the 9:03 am browser, and CreationTime = 2 to the 9:06 am browser.

At 10:30 pm, when you run a test with these browser objects, suppose the browsers are opened at 10:31 pm, 10:33 pm, and 10:34 pm. QuickTest identifies the browsers, as follows: the 10:31 pm browser is identified with the browser test object with CreationTime = 0, 10:33 pm browser is identified with the test object with CreationTime = 1, 10:34 pm browser is identified with the test object with CreationTime = 2.

If there are several open browsers, the one with the lowest CreationTime is the first one that was opened and the one with the highest CreationTime is the last one that was opened. For example, if there are three or more browsers open, the one with CreationTime = 2 is the third browser that was opened. If seven browsers are opened during a recording session, the browser with CreationTime = 6 is the last browser opened.

If a step was created on a Browser object with a specific CreationTime value, but during a run session there is no open browser with that CreationTime value, the step will run on the browser that has the highest CreationTime value. For example, if a step was created on a Browser object with CreationTime = 6, but during the run session there are only two open browsers, with CreationTime = 0 and CreationTime = 1, then the step runs on the last browser opened, which in this example is the browser with CreationTime = 1.

Note: It is possible that at a particular time during a session, the available CreationTime values may not be sequential. For example, if you open six browsers during a record or run session, and then during that session, you close the second and fourth browsers (CreationTime values 1 and 3), then at the end of the session, the open browsers will be those with CreationTime values 0, 2, 4, and 5.

Enabling and Disabling Smart Identification

Selecting the **Enable Smart Identification** check box for a particular test object class instructs QuickTest to learn the property values of all properties specified as the object's base and/or optional filter properties in the Smart Identification Properties dialog box.

By default, some test objects already have Smart Identification configurations and others do not. Those with default configurations also have the **Enable Smart Identification** check box selected by default.

You should enable the Smart Identification mechanism only for test object classes that have defined Smart Identification configuration. However, even if you define a Smart Identification configuration for a test object class, you may not always want to learn the Smart Identification property values. If you do not want to learn the Smart Identification properties, clear the **Enable Smart Identification** check box.

Note: Even if you choose to learn Smart Identification properties for an object, you can disable use of the Smart Identification mechanism for a specific object in the Object Properties or Object Repository dialog box. You can also disable use of the mechanism for an entire test in the Run node of the Test Settings dialog box. For more information, see Chapter 5, "Managing Test Objects in Object Repositories," and "Defining Run Settings for Your Test" on page 1270.

However, if you do not learn Smart Identification properties, you cannot enable the Smart Identification mechanism for an object later.

For more information on the Smart Identification mechanism, see "Configuring Smart Identification" on page 121.

Restoring Default Object Identification Settings for Test Objects

You can restore the default settings for object identification and the Smart Identification property settings for all loaded environments, for the current environment only, or for a selected test object.

Only built-in object properties can be reset. When you reset the settings for the Standard Windows environment, user-defined objects are also deleted. For more information on user-defined objects, see "Mapping User-Defined Test Object Classes" on page 131.

Note: Only currently loaded environments are listed in the Environments box in the Object Identification dialog box.

By default, the **Reset Test Object** button is displayed, but you can click the down arrow to select one of the following options:

- **Reset Test Object.** Resets the settings for the selected test object to the system default.
- **Reset Environment.** Resets the settings for all the test objects in the current environment to the system default.
- **Reset All.** Resets the settings for all currently loaded environments to the system default.

Generating Automation Scripts for Your Object Identification Settings

You can click the **Generate Script** button to generate an automation script containing the current object identification 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**).

Configuring Smart Identification

Configuring Smart Identification properties enables you to help QuickTest identify objects in your application, even if some of the properties in the object's learned description have changed.

When QuickTest uses the learned description to identify an object, it searches for an object that matches all of the property values in the description. In most cases, this description is the simplest way to identify the object, and, unless the main properties of the object change, this method will work.

If QuickTest is unable to find any object that matches the learned object description, or if it finds more than one object that fits the description, then QuickTest ignores the learned description, and uses the Smart Identification mechanism to try to identify the object.

While the Smart Identification mechanism is more complex, it is more flexible. Therefore, if configured logically, a Smart Identification definition can probably help QuickTest identify an object, if it is present, even when the learned description fails.

The Smart Identification mechanism uses two types of properties:

- **Base Filter Properties.** The most fundamental properties of a particular test object class; those whose values cannot be changed without changing the essence of the original object. For example, if a Web link's tag was changed from <A> to any other value, you could no longer call it the same object.
- **Optional Filter Properties.** Other properties that can help identify objects of a particular class. These properties are unlikely to change on a regular basis, but can be ignored if they are no longer applicable.

Understanding the Smart Identification Process

If QuickTest activates the Smart Identification mechanism during a run session (because it was unable to identify an object based on its learned description), it follows the following process to identify the object:

1 QuickTest "forgets" the learned test object description and creates a new **object candidate** list containing the objects (within the object's parent object) that match all of the properties defined in the Base Filter Properties list.

2 QuickTest filters out any object in the object candidate list that does not match the first property listed in the Optional Filter Properties list. The remaining objects become the new object candidate list.

3 QuickTest evaluates the new object candidate list:

- ➤ If the new object candidate list still has more than one object, QuickTest uses the new (smaller) object candidate list to repeat step 2 for the next optional filter property in the list.
- ➤ If the new object candidate list is empty, QuickTest ignores this optional filter property, returns to the previous object candidate list, and repeats step 2 for the next optional filter property in the list.
- ➤ If the object candidate list contains exactly one object, then QuickTest concludes that it has identified the object and performs the statement containing the object.

4 QuickTest continues the process described in steps 2 and 3 until it either identifies one object, or runs out of optional filter properties to use.

If, after completing the Smart Identification elimination process, QuickTest still cannot identify the object, then QuickTest uses the learned description plus the ordinal identifier to identify the object.

If the combined learned description and ordinal identifier are not sufficient to identify the object, then QuickTest stops the run session and displays a Run Error message.

Reviewing Smart Identification Information in the Test Results

If the learned description does not enable QuickTest to identify a 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.

If QuickTest successfully uses Smart Identification to find an object after no object matches the learned description, the step is assigned a **Warning** status in the Test Results, and the result details for the step indicate that the Smart Identification mechanism was used.

If the Smart Identification mechanism cannot successfully identify the object, QuickTest uses the learned description plus the ordinal identifier to identify the object. If the object is still not identified, the test fails and a normal failed step is displayed in the results.

For more information, see "Analyzing Smart Identification Information in the Test Results" on page 1024.

Walking Through a Smart Identification Example

The following example walks you through the object identification process for an object.

Suppose you have the following statement in your test:

```
Browser("Mercury Tours").Page("Mercury Tours").Image("Login").Click 22,17
```

When you created your test, QuickTest learned the following object description for the Login image:

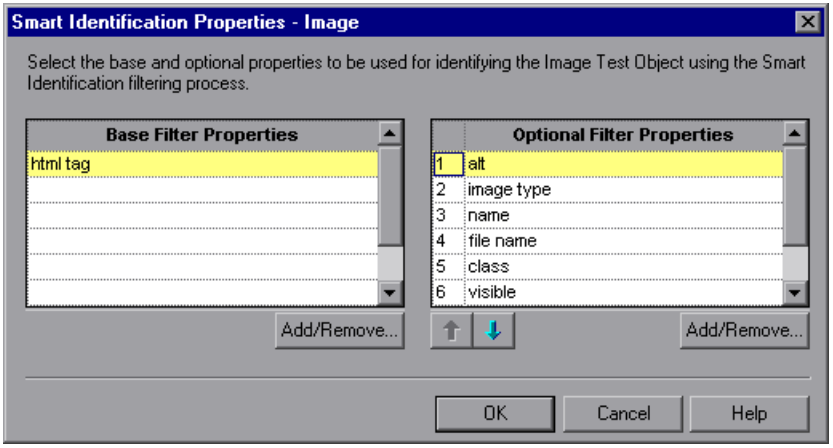
Name	Value
⊟ Description properties	
image type	Image Button
html tag	INPUT
alt	Login

However, at some point after you created your test, a second login button (for logging into the VIP section of the Web site) was added to the page, so the Web designer changed the original Login button's **alt** tag to: basic login.

The default description for Web Image objects (**alt**, **html tag**, **image type**) works for most images in your site, but it no longer works for the Login image, because that image's **alt** property no longer matches the learned description. Therefore, when you run your test, QuickTest is unable to identify the Login button based on the learned description. However, QuickTest succeeds in identifying the Login button using its Smart Identification definition.

The explanation below describes the process that QuickTest uses to find the Login object using Smart Identification:

1 According to the Smart Identification definition for Web image objects, QuickTest learned the values of the following properties it learned the Login image:



The learned values are as follows:

Base Filter Properties:

Property	Value
html tag	INPUT

Optional Filter Properties:

Property	Value
alt	Login
image type	Image Button
name	login
file name	login.gif
class	<null>
visible	1

2 QuickTest begins the Smart Identification process by identifying the five objects on the Mercury Tours page that match the base filter properties definition (**html tag** = INPUT). QuickTest considers these to be the object candidates and begins checking the object candidates against the **Optional Filter Properties** list.

3 QuickTest checks the **alt** property of each of the object candidates, but none have the **alt** value: Login, so QuickTest ignores this property and moves on to the next one.

4 QuickTest checks the **image type** property of the each of the object candidates, but none have the **image type** value: Image Button, so QuickTest ignores this property and moves on to the next one.

5 QuickTest checks the **name** property of each of the object candidates, and finds that two of the objects (both the basic and VIP Login buttons) have the name: login. QuickTest filters out the other three objects from the list, and these two login buttons become the new object candidates.

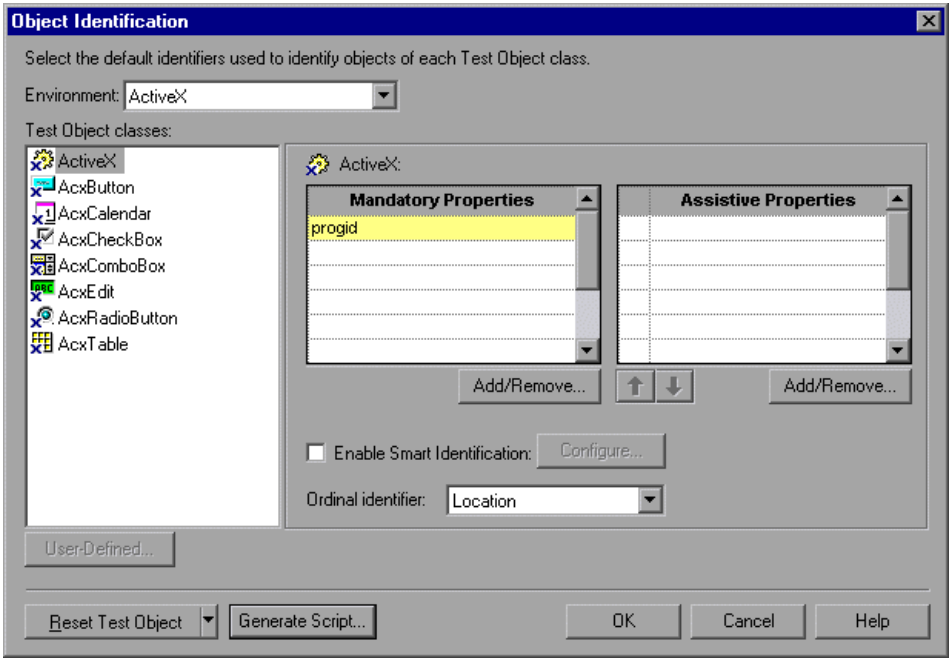
6 QuickTest checks the **file name** property of the two remaining object candidates. Only one of them has the file name login.gif, so QuickTest correctly concludes that it has found the Login button and clicks it.

Step-by-Step Instructions for Configuring a Smart Identification Definition

You use the Smart Identification Properties dialog box, accessible from the Object Identification dialog box, to configure the Smart Identification definition for a test object class.

To configure Smart Identification properties:

1 Select **Tools > Object Identification**. The Object Identification dialog box opens.



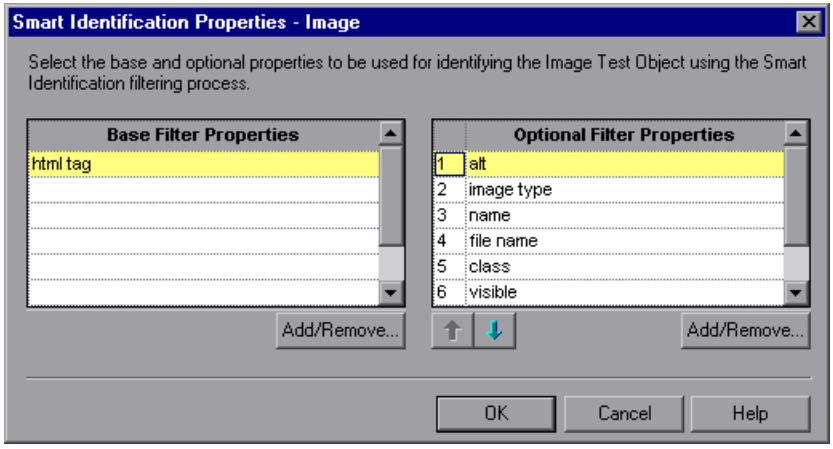
2 Select the appropriate environment in the **Environment** list. The test object classes associated with the selected environment are displayed in the **Test Object classes** list.

Note: The environments included in the Environment list are those that correspond to the loaded add-ins. For more information on loading add-ins, see the section on loading QuickTest add-ins in the *HP QuickTest Professional Add-ins Guide*.

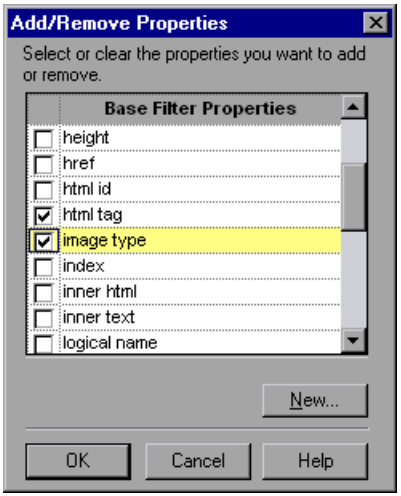
---

3 Select the test object class you want to configure.

4 Click the **Configure** button next to the **Enable Smart Identification** check box. The **Configure** button is enabled only when the **Enable Smart Identification** option is selected. The Smart Identification Properties dialog box opens.



5 In the **Base Filter Properties** list, click **Add/Remove**. The Add/Remove Properties dialog box for base filter properties opens.



6 Select the properties you want to include in the **Base Filter Properties** list and/or clear the properties you want to remove from the list.

Note: You cannot include the same property in both the base and optional property lists.

You can specify a new property by clicking **New** and specifying a valid property name in the displayed dialog box.

Tip: You can also add property names to the set of available properties for Web objects using the attribute/*<PropertyName>* notation. To do this, click **New**. The New Property dialog box opens. Enter a valid property in the format attribute/*<PropertyName>* and click **OK**. The new property is added to the **Base Filter Properties** list. For example, to add a property called MyColor, enter attribute/MyColor.

---

7 Click **OK** to close the Add/Remove Properties dialog box. The updated set of base filter properties is displayed in the **Base Filter Properties** list.

8 In the **Optional Filter Properties** list, click **Add/Remove**. The Add/Remove Properties dialog box for optional filter properties opens.

9 Select the properties you want to include in the **Optional Filter Properties** list and/or clear the properties you want to remove from the list.

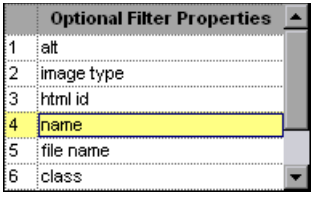
Note: You cannot include the same property in both the base and optional property lists.

You can specify a new property by clicking **New** and specifying a valid property name in the displayed dialog box.

Tip: You can also add property names to the set of available properties for Web objects using the attribute/<*PropertyName*> notation. To do this, click **New**. The New Property dialog box opens. Enter a valid property in the format attribute/<*PropertyName*> and click **OK**. The new property is added to the **Optional Filter Properties** list. For example, to add a property called MyColor, enter attribute/MyColor.

---

10 Click **OK** to close the Add/Remove Properties dialog box. The properties are displayed in the **Optional Filter Properties** list.



11 Use the up and down arrows to set your preferred order for the optional filter properties. When QuickTest uses the Smart Identification mechanism, it checks the remaining object candidates against the optional properties one-by-one according to the order you set in the **Optional Filter Properties** list until it filters the object candidates down to one object.

Mapping User-Defined Test Object Classes

The Object Mapping dialog box enables you to map an object of an unidentified or custom class to a Standard Windows class. For example, if your application has a button that cannot be identified, this button is learned as a generic WinObject. You can teach QuickTest to identify your object as if it belonged to a standard Windows button class. Then, when you click the button while recording, QuickTest records the operation in the same way as a click on a standard Windows button. When you map an unidentified or custom object to a standard object, your object is added to the list of Standard Windows test object classes as a user-defined test object class. You can configure the object identification settings for a user-defined test object class just as you would any other test object class.

You should map an object that cannot be identified only to a Standard Windows class with comparable behavior. For example, do not map an object that behaves like a button to the edit class.

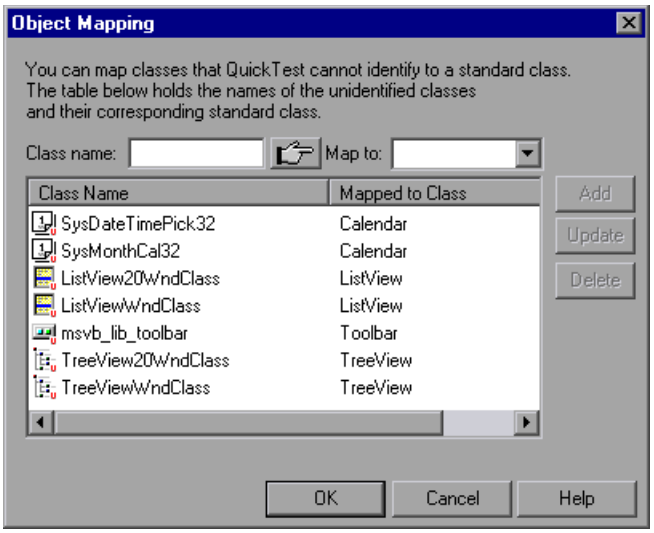
Notes:

- You can define user-defined classes only when **Standard Windows** is selected in the **Environment** box.
- If you click the down arrow on the **Reset Test Object** button and select **Reset Environment**, when **Standard Windows** is selected in the **Environment** box, all of the user-defined test object classes are deleted.

To map an unidentified or custom class to a standard Windows class:

1. Select **Tools** > **Object Identification**. The Object Identification dialog box opens.
2. Select **Standard Windows** in the **Environment** box. The **User-Defined** button becomes enabled.

3 Click **User-Defined**. The Object Mapping dialog box opens.



4 Click the pointing hand and then click the object whose class you want to add as a user-defined class. The name of the user-defined object is displayed in the **Class name** box.

For more information about using the pointing hand feature, see "Tips for Using the Pointing Hand" on page 134.

5 In the **Map to** box, select the standard object class to which you want to map your user-defined object class and click **Add**. The class name and mapping is added to the object mapping list.

6 If you want to map additional objects to standard classes, repeat steps 4 to 5 for each object.

7 Click **OK**. The Object Mapping dialog box closes and your object is added to the list of Standard Windows test object classes as a user-defined test object. Note that your object has an icon with a red U in the lower-right corner, identifying it as a user-defined class.

8 Configure the object identification settings for your user defined object class just as you would any other object class. For more information, see "Configuring Mandatory and Assistive Properties" on page 108, and "Configuring Smart Identification" on page 121.

To modify an existing mapping:

1 In the Object Mapping dialog box, select the class you want to modify from the object mapping list. The class name and current mapping are displayed in the Class name and Map to boxes.

2 Select the standard object class to which you want to map the selected user-defined class and click **Update**. The class name and mapping is updated in the object mapping list.

3 Click **OK** to close the Object Mapping dialog box.

To delete an existing mapping:

1 In the Object Mapping dialog box, select the class you want to delete from the object mapping list.

2 Click **Delete**. The class name and mapping is deleted from the object mapping list in the Object Mapping dialog box.

3 Click **OK**. The Object Mapping dialog box closes and the class name is deleted from the Standard Windows test object classes list in the Object Identification dialog box.

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.

5

Managing Test Objects in Object Repositories

This chapter explains how to manage and maintain the objects in your object repositories. It describes how to modify object properties and how to modify the way QuickTest identifies an object, which is useful when working with objects that change dynamically.

This chapter includes:

Adding Test Objects to a Local or Shared Object Repository

The functionality described in this section is available in the Object Repository window for the local object repository, and the Object Repository Manager for shared object repositories.

When you create a shared object repository for your keyword-driven testing infrastructure, you can add test objects to it in different ways. You can choose to add only a selected test object, or to add all test objects of a certain type, such as all button objects, or to add all test objects of a specific class, such as all WebButton objects.

You can use the Navigate and Learn option, for example, to add objects to the shared object repository according to your defined filter. If you record a test, QuickTest adds each object on which you perform an operation to the local object repository (for objects that do not already exist in an associated shared object repository). You can also add test objects to the local object repository while editing your test.

For example, you may find that users need to perform a step on an object that is not in the object repository. You may also find that an additional object was added to the application you are testing after you built the object repository. You can add the object directly to a shared object repository using the Object Repository Manager, so that it is available in all actions that use this shared object repository. Alternatively, you can add it to the local object repository of the action.

Note: You can add a test object to the local object repository only if that test object does not already exist in a shared object repository that is associated with the action. If a test object already exists in an associated shared object repository, you can add it to the local object repository using the **Copy to Local** option. For more information, see "Copying an Object to the Local Object Repository" on page 195.

If needed, you can merge test objects from the local object repository into a shared object repository. For more information, see Chapter 8, "Merging Shared Object Repositories."

You can also add test objects to a shared object repository while navigating through your application. For more information, see "Adding Test Objects Using the Navigate and Learn Option" on page 225.

Tips:

- You can also add a test object to the local object repository by choosing it from your application in the Select Object for Step dialog box (from a new step in the Keyword View or from the Step Generator).
- You can add new test objects to your object repository that do not yet exist in your application. For more information, see "Defining New Test Objects" on page 147.

Adding a Test Object Using the Add Objects to Local or Add Objects Option

You can add test objects to a local or shared object repository directly from your application. You can choose to add a specific test object either with or without its descendants. You can also control which descendants to add, according to their object and class types, based on selections that you define in the object filter.

Note: You cannot add WinMenu objects directly to an object repository using the **Add Objects to Local** button in the Object Repository window or the **Add Objects** button in the Object Repository Manager. If you want to add a WinMenu object to the object repository, you can use the **Add Objects** or **Add Objects to Local** button to add its parent object and then select to add the parent object together with its descendants, or you can record a step on a WinMenu object and then delete the recorded step.

To add test objects to the object repository using the Add Objects to Local or Add Objects option:

1 Perform one of the following:

➤ In the Object Repository window, Select **Object** > **Add Objects to Local** or click the **Add Objects to Local** toolbar button. If you select this option, the test object is added to the local object repository and can only be used by the current action.

➤ In the Object Repository Manager, select **Object** > **Add Objects** or click the **Add Objects** toolbar button. If you select this option, the test object is added to a shared object repository and can be used in multiple actions.

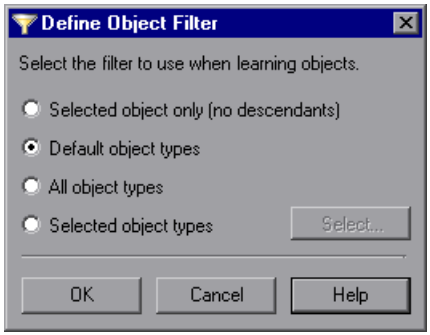
QuickTest and the Object Repository window or Object Repository Manager are hidden, and the pointer changes into a pointing hand. For more information on using the pointing hand, see "Tips for Using the Pointing Hand" on page 141.

2 Click the object you want to add to your object repository.

3 If the location you click is associated with more than one object, the Object Selection dialog box opens. Select the object you want to add to the repository and click **OK**.

If the object you select in the Object Selection dialog box is a bottom-level object in the test object hierarchy, for example, a WebButton object, it is added directly to the object repository.

If the object you select in the Object Selection dialog box is a parent (container) object, such as a browser or page in a Web environment, or a dialog box in a standard Windows application, the Define Object Filter dialog box opens. The Define Object Filter dialog box retains the settings that you defined in the previous add object session.



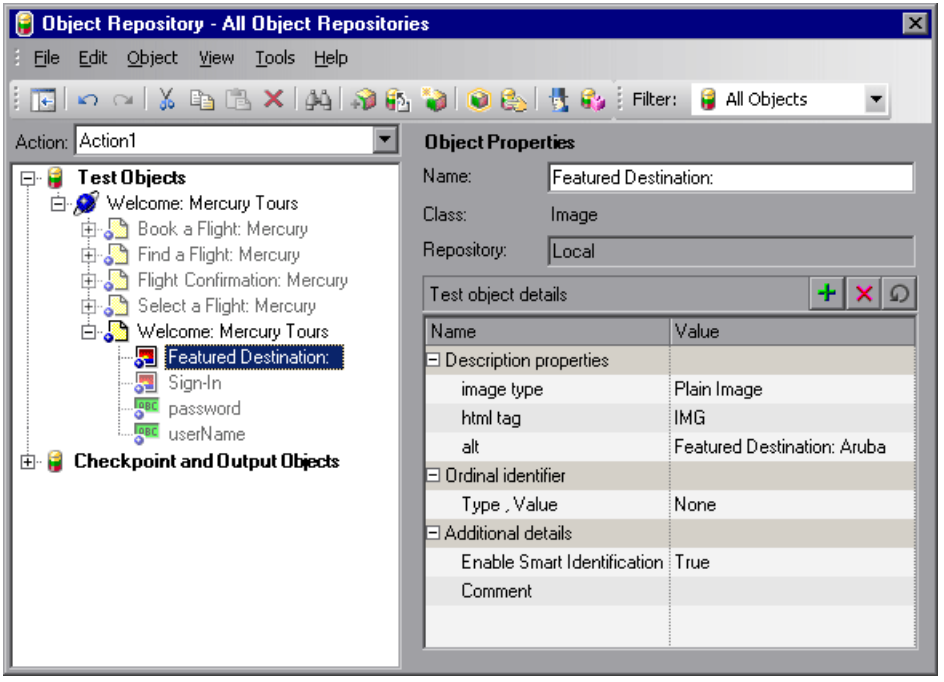
You can choose from the following options:

- **Selected object only (no descendants).** Adds to the object repository the previously selected object's properties and values, without its descendant objects.
- **Default object types.** Adds to the object repository the previously selected object's properties and values, with the properties and values of its descendant objects according to the object types specified by the default filter. You can see which objects are in the default filter by clicking the **Select** button and then clicking the **Default** button.
- **All object types.** Adds to the object repository the previously selected object's properties and values, together with the properties and values of all of its descendant objects.
- **Selected object types.** Adds to the object repository the previously selected object's properties and values, as well as the properties and values of its descendant objects according to the object types and classes you specify in the object filter. You specify the objects and classes in the filter by clicking the **Select** button and selecting the required items in the Select Object Types dialog box. For more information on the Select Object Types dialog box, see "Understanding the Select Object Types Dialog Box" on page 146.

4 Select the required option and click **OK** to close the Define Object Filter dialog box and add the specified objects to the object repository according to the selected object filter.

5 The Object Repository window is redisplayed, showing the new local objects and their properties and values in the object repository. If you chose to add the objects from the Object Repository Manager, the objects are added to the active shared object repository.

QuickTest also adds the new object's parent objects if they do not already exist in the object repository. Local objects are shown in black in the object repository tree to indicate they are editable; shared objects are shown in gray and can only be edited in the Object Repository Manager.



You can edit the new test object's details just as you would edit any other object in a local or shared object repository. For more information, see "Maintaining Identification Properties" on page 162.

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.

Adding Test Objects to the Local Object Repository from the Active Screen

You can add test objects to the local object repository of the current action by selecting the required object in the Active Screen.

To add test objects to the object repository using the Active Screen, the Active Screen must contain information for the object you want to add. You can control how much information is captured in the Active Screen in the Active Screen node of the Options dialog box. For more information, see "Setting Active Screen Options" on page 1240.

When you add a test object to the object repository in one of the ways described in this section, the test object is added to the local object repository and can only be used by the current action. If you want to add the test object to the shared object repository, so that it can be used in multiple actions, add it using the Object Repository Manager (not from the Active Screen).

To add a test object to the object repository using the View/Add Object option from the Active Screen:

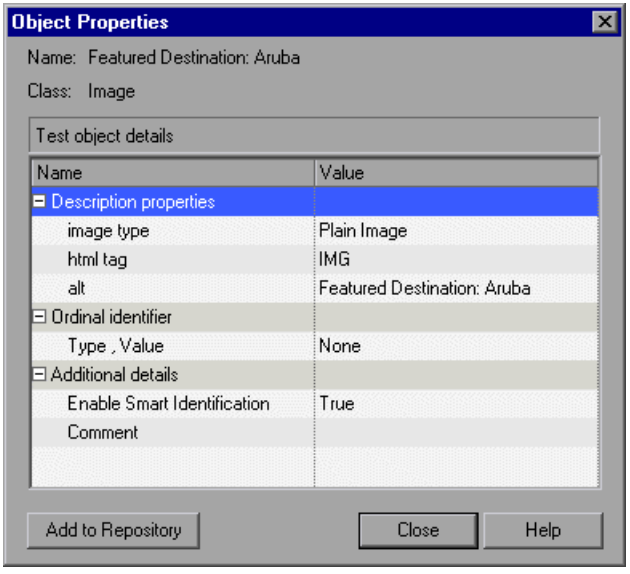
1 If the Active Screen is not displayed, select **View** > **Active Screen** or click the Active Screen toolbar button to display it.

2 Select a step in your test whose Active Screen contains the object that you want to add to the object repository.

3 In the Active Screen, right-click the object you want to add and select **View/Add Object**.

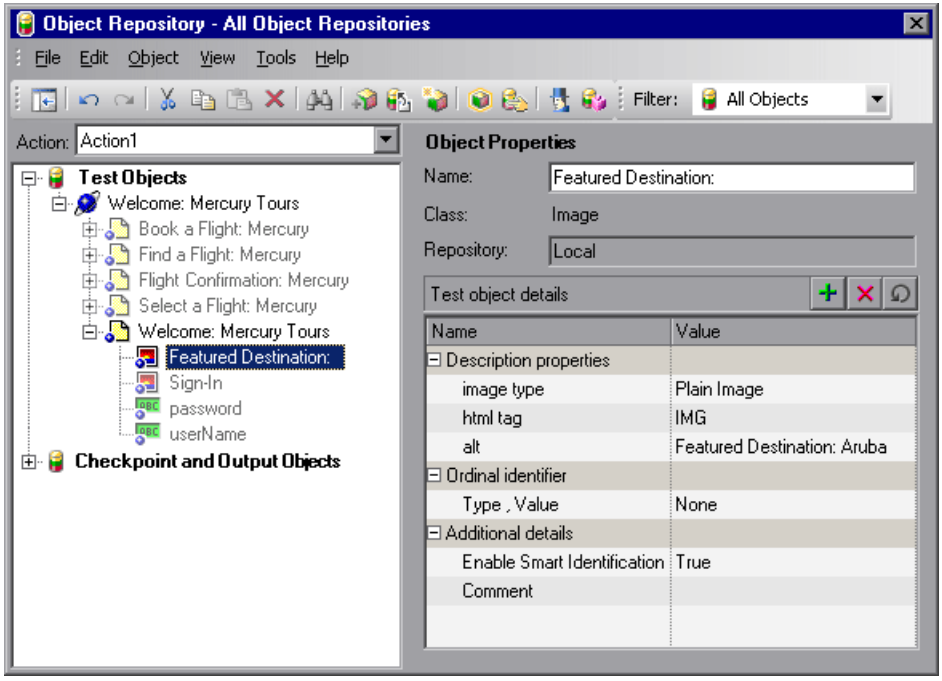
4 If the location you clicked is associated with more than one object, the Object Selection dialog box opens. Select the object you want to add to the object repository, and click **OK** to close the Object Selection dialog box.

5 The Object Properties dialog box opens and displays the default identification properties for the object.



6 Click **Add to Repository**. The selected object is added to the local object repository for the current action with the default identification properties and values. The **Add to Repository** button changes to **View in Repository**.

7 Click **View in Repository**. The Object Repository window opens and displays the object properties for the selected test object.



You can edit your new test object's properties in the Object Repository window just as you would any other test object in your local object repository.

To add a test object to the object repository by inserting a step from the Active Screen:

1 If the Active Screen is not displayed, select **View** > **Active Screen** or click the Active Screen toolbar button to display it.

2 Select a step in your test whose Active Screen contains the object for which you want to add a step.

3 In the Active Screen, right-click the object for which you want to add a step and select the type of step you want to insert (checkpoint, output value, Step Generator, and so forth).

4 If the location you clicked is associated with more than one object, the Object Selection dialog box opens. Select the object for which you want to add a step, and click **OK**.

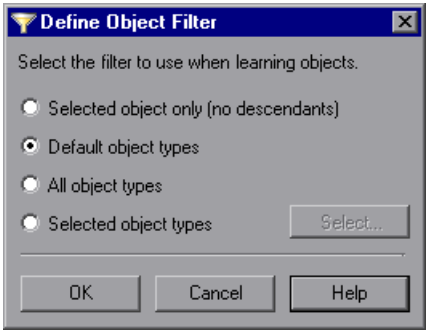
The appropriate dialog box opens, enabling you to configure your preferences for the step you want to insert.

5 Set your preferences and select whether to insert the step before or after the step currently selected in the Keyword View or in the Expert View. Click **OK** to close the dialog box. A new step is inserted in your test, and the object is added to the local object repository for the current action (if it was not yet included).

Understanding the Define Object Filter Dialog Box

When adding a test object to the object repository, if the object you select to add is typically a parent object, such as a browser or page in a Web environment or a dialog box in a standard Windows application, the Define Object Filter dialog box opens.

The object filter contains predefined settings that decide which objects should be learned (while using the **Navigate and Learn** option or the **Add Objects** option). The option you select in the Define Object Filter dialog box is saved and used for each subsequent learn session.



You can choose from the following options:

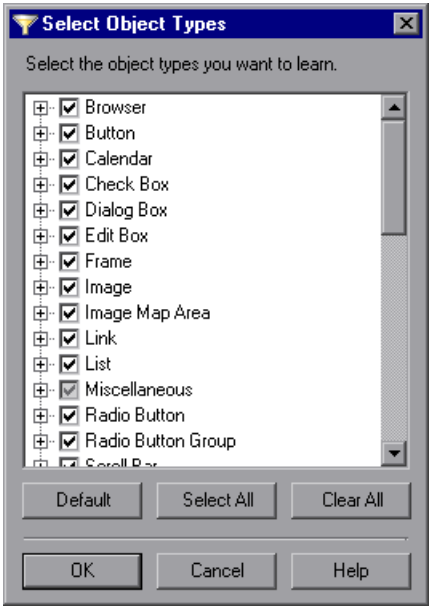
- **Selected object only (no descendants)**. Adds to the object repository the previously selected object's properties and values, without its descendant objects.
- **Default object types**. Adds to the object repository the previously selected object's properties and values, with the properties and values of its descendant objects according to the object types specified by the default filter. You can see which objects are in the default filter by selecting **Selected object types**, clicking the **Select** button, and then clicking the **Default** button.
- **All object types**. Adds to the object repository the previously selected object's properties and values, together with the properties and values of all of its descendant objects.
- **Selected object types**. Adds to the object repository the previously selected object's properties and values, as well as the properties and values of its descendant objects according to the object types and classes you specify in the object filter. You specify the objects and classes in the filter by clicking the **Select** button and selecting the required items in the Select Object Types dialog box. For more information on the Select Object Types dialog box, see "Understanding the Select Object Types Dialog Box" on page 146.

Understanding the Select Object Types Dialog Box

The Select Object Types dialog box enables you to specify a custom object filter for adding test objects to the object repository (while using the **Navigate and Learn** option or the **Add Objects** option).

When you define an object filter, it is automatically saved for future add object operations (performed from both the **Navigate and Learn** option and the **Add Objects** option).

You open the Select Object Types dialog box by clicking the **Select** button in the Define Object Filter dialog box.



The object types in this list are a generic grouping of objects according to the general object characteristics. For example, the **List** type contains list and list view objects, as well as combo boxes; the **Table** type contains both tables and grids.

The list shows all objects supported by the installed add-ins and is not specific to the object you selected. For some add-ins, certain child objects may be automatically filtered out and not added to the object repository when you choose to add all descendants of a specific object, even if those object types are selected in the list. If you want to add an object that is automatically filtered out, you can add it by selecting it in the Object Selection dialog box. To check whether your add-in automatically filters out certain objects, see the *HP QuickTest Professional Add-ins Guide*.

Tip: Click **Select All** or **Clear All** to select or clear all the check boxes in the Select Object Types dialog box. Click **Default** to restore the check box selections to their preset defaults. The preset defaults are equivalent to choosing the **Default object types** option in the Define Object Filter dialog box.

Make your selections and click **OK** to define your custom object filter and close the Select Object Types dialog box.

Defining New Test Objects

You can define test objects in your object repository that do not yet exist in your application. This enables you to prepare an object repository and build tests for your application before the application is ready for testing.

For example, you may already know the names, types, and descriptive properties of some of the objects in your application, and know only the types of other objects in your application. Before your application is ready, you can create WebEdit objects for UserName and Password fields in your Login page (plus the relevant parent Page and Browser objects). If you know the property values for these objects, you can also add them. If not, you can add them when your application is ready for testing.

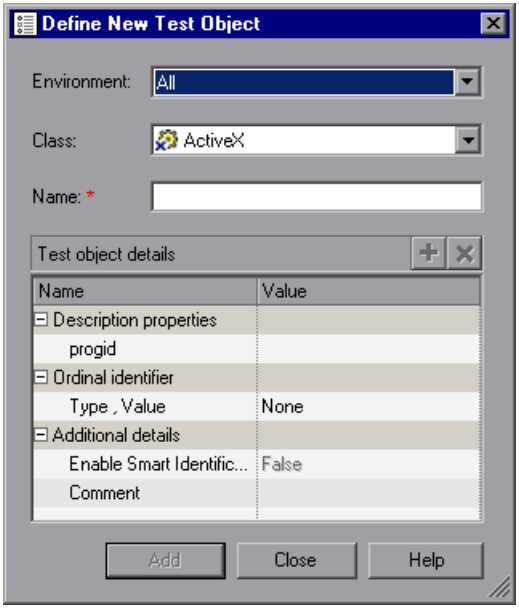
When you define a new object in the object repository as described in this section, the object is added to the local object repository and can only be used by the current action. If you want to add the object to the shared object repository so that it can be used in multiple actions, you must add it using the Object Repository Manager. For more information, see Chapter 7, "Managing Object Repositories."

After you have defined the new test object, if the properties of the object in your application do not match the test object description that you defined, or if an object has been updated in your application, you can update the object description at any time. For more information, see "Updating Identification Properties from an Object in Your Application" on page 165.

To define a new test object:

1 Select the object under which you want to define the new object, according to the correct object hierarchy.

2 Click the **Define New Test Object** button or select **Object** > **Define New Test Object**. The Define New Test Object dialog box opens.



3 In the **Environment** box, select the appropriate environment. The test object classes associated with the selected environment are displayed in the **Class** box.

Notes:

- The environments included in the **Environment** list correspond to the loaded add-ins. For more information on loading add-ins, see the section on loading QuickTest add-ins in the *HP QuickTest Professional Add-ins Guide*.
- The **Environment** list might also include additional environments for which you or a third party developed support using add-in extensibility.

---

4 In the **Class** box, select the class of the test object you want to define.

5 In the **Name** box, enter a name for the new test object. After you enter a name, the **Test object details** area is enabled.

6 In the **Test object details** area, define the properties and values for your test object. The **Test object details** area automatically contains the mandatory properties defined for the object class in the Object Identification dialog box. You can add or remove properties as required, and define values for the properties. For more information, see "Maintaining Identification Properties" on page 162.

7 Click **Add**. The new test object is added to the local object repository in the selected location.

8 Repeat step 3 to step 7 to define additional test objects, or click **Close** to close the Define New Test Object dialog box.

Copying, Pasting, and Moving Objects in the Object Repository

The functionality described in this section is available in the Object Repository window for objects in the local object repository, and the Object Repository Manager for objects in shared object repositories.

Note: You can use the **Edit** > **Undo** and **Edit** > **Redo** options or **Undo** and **Redo** buttons to cancel or repeat your changes. When you save the object repository, you cannot undo and redo operations that were performed before the save operation.

The following procedures describe the ways in which you can copy, paste, and move objects:

To move an object to a different location within an object repository:

Drag the object up or down the tree and drop it at the required location. By default, when you drag an object, any child objects are also moved with it.

To copy an object to a different location within an object repository:

Press the CTRL key while dragging the object and drop it at the required location in the tree. By default, when you drag an object, any child objects are also moved with it.

To move or copy an object without its child objects:

Drag the object using the right mouse button. When you drop the object at the required location, you can choose whether to drop it with or without its children. By default, when you drag an object, any child objects are also moved or copied with it.

To cut, copy, and paste objects within an object repository:

Use the corresponding toolbar buttons or the options in the **Edit** menu. When you cut, copy, and paste objects, the operation is performed also on the child objects of the selected object, if any.

To cut, copy, and paste objects between shared object repositories:

In the Object Repository Manager, use the corresponding toolbar buttons or the options in the **Edit** menu. When you cut, copy, and paste objects, the operation is performed also on the child objects of the selected object, if any.

To copy objects from one shared object repository to another:

In the Object Repository Manager, open the required shared object repositories. Drag the object from one window and drop it at the required location in the other window.

To move objects from one shared object repository to another:

In the Object Repository Manager, open the required shared object repositories. Press the CTRL key while you drag the object from one window and drop it at the required location in the other window. Note that moving an object removes it from one shared object repository and adds it to the other shared object repository.

You can also copy objects from a shared object repository to the local object repository to modify them locally. For more information, see "Copying an Object to the Local Object Repository" on page 195.

Guidelines for Copying, Pasting, and Moving Objects

When copying, pasting, or moving objects, consider the following guidelines:

- You cannot modify the root node of an object repository.
- If you change the object hierarchy, ensure that the new hierarchy is valid.
- If you paste or move an object to a different hierarchical level, you can choose whether to copy all objects up to the shared parent object (in the message displayed when you perform such an operation).
- In the Object Repository window, when you copy, paste, and move objects from a shared object repository associated with a test, the objects are copied, pasted, or moved to the local object repository of the test.

- If you move an object to its immediate parent, QuickTest creates a copy of the object (renamed with an incremental suffix) and pastes it as a sibling of the original object.
- If you cut or copy an object, and then paste it on its parent object, QuickTest creates copy of the object (renamed with an incremental suffix) and inserts it at the same level as the original object.
- You cannot move an object to any of its descendants.
- You cannot copy or move an object to be a child of a bottom-level object (an object that cannot contain a child object) in the object hierarchy.
- You cannot copy, paste, or move objects that have unmapped repository parameters from a shared object repository to the local object repository. If you copy, paste, or move an object from a shared object repository to the local object repository and the object or one of its parent objects are parameterized using one or more repository parameters, the repository parameter values are converted when you copy, paste, or move the object. For example, if the repository parameter is mapped to a Data Table parameter, the property is parameterized using a Data Table parameter. If the value is a constant value, the property receives the same constant value.

Deleting Objects from the Object Repository

The functionality described in this section is available in the Object Repository window for objects in the local object repository, and the Object Repository Manager for objects in shared object repositories.

When you remove a step from your test, its corresponding object remains in the object repository.

If you are working with a local object repository and the object in the step you removed does not occur in any other steps within that action, you can delete the object from the object repository.

If you are working with a shared object repository, confirm that the object does not appear in any other action using the same shared object repository before you choose to delete the object from the object repository.

You delete objects in the local object repository using the Object Repository window, and objects in the shared object repository using the Object Repository Manager.

Note: If your action contains references to an object that you deleted from the object repository, your test run will fail.

To delete an object from the object repository:

1. In the repository tree, select the object you want to delete.
2. Click the **Delete** button or select **Edit** > **Delete**.
3. Click **Yes** to confirm that you want to delete the object. The object is deleted from the object repository.

Tip: The **Delete** button enables you to delete any selected value or item in the object repository, not just test objects. For example, you can use it to delete part of an object name or a property value.

Locating Objects

The functionality described in this section is available in the Object Repository window for objects in the local object repository, and the Object Repository Manager for objects in shared object repositories.

You can search for a specific object in your object repository in several ways. You can search for an object according to its type. For example, you can search for a specific edit box, or you can point to an object in your application to automatically highlight that same object in your repository. You can select an object in your object repository and highlight it in your application to check which object it is. For local objects (and shared objects in an editable shared object repository when using the Object Repository Manager), you can also replace specific property values with other property values. For example, you can replace the property value userName with user name.

Finding Objects in an Object Repository

You can use the Find and Replace dialog box to find an object, property, or property value in an object repository. You can also find and replace specified property values.

You replace property values for objects in the local object repository using the Object Repository window. You replace property values for objects in shared object repositories using the Object Repository Manager.

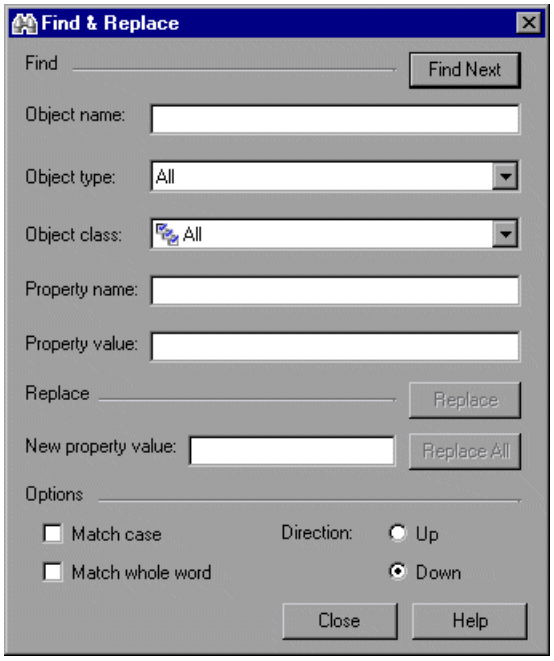
Notes:

- The Find and Replace dialog box can only find checkpoint and output values by searching for the object name.
- You cannot use the Find and Replace dialog box to replace property or object names. You cannot replace property values in a read-only test.

To find an object, property, or property value in the object repository:

1 Make sure that the relevant object repository is open (in the Object Repository window or Object Repository Manager).

2 Click the **Find & Replace** button or select **Edit** > **Find & Replace**. The Find & Replace dialog box opens.



3 Specify one or more criteria by which you want to search for the object, property, or property value:

- **Object name.** Enter the name or partial name of the object you want to find.
- **Object type.** Select the type of object you want to find, for example, **Button**.

Note: The object types in this list are a generic grouping of objects according to the general object characteristics. For example, the **List** type contains list and list view objects, as well as combo boxes; the **Table** type contains both tables and grids.

- **Object class.** Select the class of object you want to find, for example, **WebButton**. The classes available depend on the selection you made in the **Object type** box.
- **Property name.** Specify the name or partial name of the property you want to find.
- **Property value.** Specify the property value or partial property value you want to find.

4 If you specified a property value and want to replace it with a different value, enter the new property value in the **New property value** box.

5 Specify the search parameters, as follows:

- If you want the search to distinguish between upper and lower case letters, select **Match case**.
- If you want the search to find only complete words that exactly match the single word you entered, select **Match whole word**.
- Specify the direction in which you want to search: **Up** or **Down**.

6 Perform the find or replace operation in one of the following ways. The search is performed on the entire object repository, starting with the currently selected object and in the direction you specified. To find the next instance, click **Find Next** again.

- To find the specified object, property, or property value, click **Find Next**. The first instance of the searched word is displayed.
- To individually find and replace each instance of the property value for which you are searching, click **Find Next**. When an instance is found, click **Replace**. The property value is replaced, and the next instance of the property value, if any, is highlighted.
- To replace all instances of the specified property value with the new property value, click **Replace All**. Instances in shared object repositories that are not editable are not changed.

Highlighting an Object in Your Application

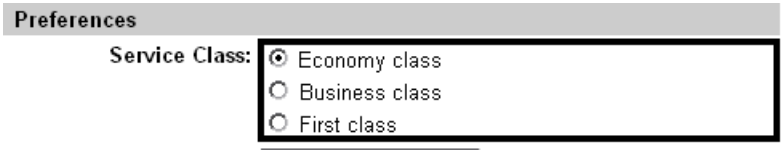
You can select a test object in your object repository and highlight it in the application you are testing. When you choose to highlight a test object, QuickTest indicates the selected object's location in your application by temporarily showing a frame around the object and causing it to flash briefly. The application must be open to the correct context so that the object is visible.

For example, to locate the User Name edit box in a Web page, you can open the relevant page in the Web browser and then select the User Name test object in the object repository. When you choose the **Highlight in Application** option, the User Name edit box in your browser is framed in the Web page and flashes several times.

Note: Both the frame and the flashing behavior are temporary.

To highlight an object in your application:

1. Make sure your application is open to the correct window or page.
2. Select the test object you want to highlight in your object repository.
3.  Click the **Highlight in Application** button or select **View** > **Highlight in Application**. The selected object is highlighted with a border in the application.



Note: If the application is not open to the correct context, the object is not highlighted and a message is displayed.

Locating a Test Object in the Object Repository

You can select an object in the application you are testing and highlight the test object in the object repository.

For example, to locate a Find a Flight image in a Web page, you can select it in your Web page using the pointing hand mechanism. After you select the Find a Flight image object from the selection dialog box and click **OK**, the parent hierarchy in the object repository tree expands and the Find a Flight image test object is highlighted.

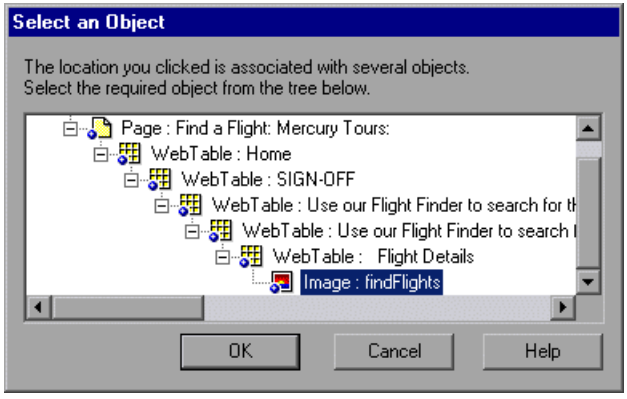
To locate an object in the object repository:

1 Make sure your application is open to the correct window or page.

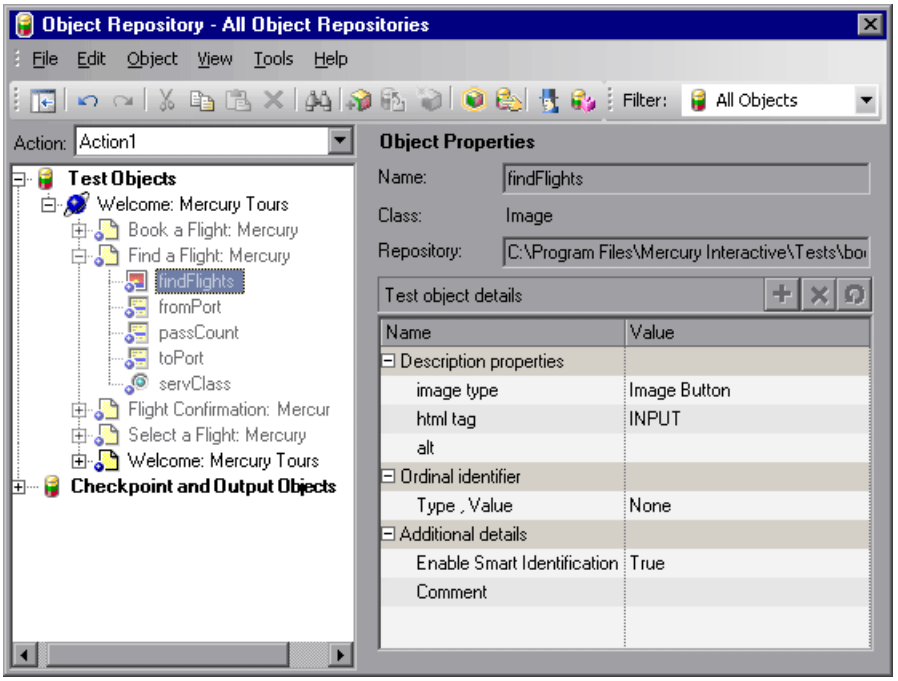
2 Click the **Locate in Repository** button or select **View** > **Locate in Repository**. QuickTest is hidden, and the pointer changes into a pointing hand.

3 Use the pointing hand to click on the required object in your application. For more information on using the pointing hand, see “Tips for Using the Pointing Hand” on page 161.

If the location you clicked is associated with more than one object, the Select an Object dialog box opens.



4 Select the object you want to locate in the object repository and click **OK**. The selected object is highlighted in the object repository.



Tip: If the relevant object repository is not open or the object cannot be found, the object is not highlighted. In the Object Repository Manager, if more than one shared object repository is open, and QuickTest cannot locate the selected object in the active object repository, you can choose whether to look for the object in all of the currently open object repositories.

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.

Maintaining Identification Properties

As applications change, you may need to change the property values of the steps in your test. Suppose an object in your application is modified. If that object is part of your test, you should modify its values so that QuickTest can continue to identify it. For example, if a company Web site contains a **Contact Us** hypertext link, and the text string in this link is changed to **Contact My Company**, you need to update the object's details in the object repository so that QuickTest can continue to identify the link properly.

You can modify identification properties in a number of ways. For an object stored in a local object repository, you can modify its properties directly from the Object Repository window. For an object stored in a shared object repository, you can either open it in the Object Repository Manager and modify its properties, or you can copy it to the local object repository and then modify its properties.

For more information on different ways in which you can modify identification properties, see:

- "Specifying or Modifying Property Values" on page 163
- "Updating Identification Properties from an Object in Your Application" on page 165
- "Restoring Default Mandatory Properties for a Test Object" on page 168
- "Renaming Test Objects" on page 169
- "Adding Properties to a Test Object Description" on page 171
- "Defining New Identification Properties" on page 174
- "Removing Properties from a Test Object Description" on page 177
- "Specifying Ordinal Identifiers" on page 177

Specifying or Modifying Property Values

The functionality described in this section is available in the Object Repository window for objects in the local object repository, and the Object Repository Manager for objects in shared object repositories.

You can specify or modify values for properties in the test object description. You can specify a value using a constant value (either a simple value or a constant value that includes regular expressions) or you can parameterize it. You can also change the set of properties used to identify that object.

You can also automatically update the description of one or more test objects in your object repository based on the actual updated object properties in your application. For more information, see "Updating Identification Properties from an Object in Your Application" on page 165.

You can also find and replace specific identification property values. For more information, see "Finding Objects in an Object Repository" on page 154.

Note: In some cases, the Smart Identification mechanism may enable QuickTest to identify a test object, even when some of its property values change. However, if you know about property value changes for a specific test object, you should try to correct the test object definition so that QuickTest can identify the test object from its basic object description. For more information on the Smart Identification mechanism, see Chapter 4, "Configuring Object Identification."

Tip: You can use the Object Spy at any time to view the native properties and values of the objects in the application you are testing, or the identification properties of the test objects that represent them. You open the Object Spy by choosing **Tools > Object Spy** or clicking the **Object Spy** toolbar button. For more information, see "Viewing Object Properties and Operations Using the Object Spy" on page 97.

To specify an identification property value:

1 In the Object Repository window or Manager, select the test object whose property value you want to specify.

2 In the **Test object details** area, click in the value cell for the required property.

Tips: For a test object in the local object repository, you can also right-click the step containing the test object and select **Object Properties**, and then make the following property value changes in the Object Properties dialog box.

If you want to view all objects in the action, click the **View in Repository** button. The Object Repository window opens and displays all objects stored in the repository in a repository tree.

You can also open the object repository for the selected action by choosing **Resources** > **Object Repository** or by clicking the **Object Repository** toolbar button.

---

3 Specify the property value in one of the following ways:

- If you want to specify a constant value, enter it in the value cell.
- If you want to parameterize the value or specify a constant value using a regular expression, click the parameterization button in the value cell. If you specify a constant value using a regular expression, the  icon is displayed next to the value.

For information on specifying property values, see “Configuring a Selected Value” on page 760.

4 If you specified a constant value, it is shown in the **Value** column of the **Test object details** area. If you parameterized the value, the parameter name is shown with one of the following icons in the **Value** column.

Parameter Icon	Description
	Indicates that the value of the property is currently a test or action parameter.
	Indicates that the value of the property is currently a Data Table parameter.
	Indicates that the value of the property is currently an environment variable parameter.
	Indicates that the value of the property is currently a random number parameter.
	Indicates that the value of the property is currently a repository parameter (in a shared object repository).

Updating Identification Properties from an Object in Your Application

The functionality described in this section is available in the Object Repository window for objects in the local object repository, and the Object Repository Manager for objects in shared object repositories.

You can update a test object in your object repository by selecting the corresponding object in your application and relearning its properties and property values from the application. When you update a test object description in this way, all currently defined properties and values are overwritten, including description properties and values, the ordinal identifier, and Smart Identification information. The updated object description is based on the current definitions in the Object Identifications dialog box. Only the object-specific comments, if any, are retained.

This is useful if an object's properties have changed since you added it to the object repository, since QuickTest may not be able to recognize the object unless you update its description.

You can also use this option to update an object that you defined (using the **Object** > **Define New Test Object** option) before the application was completely developed, and as a result some of the identification properties and values are missing in the test object description, or are no longer sufficient to identify the object. For more information on the **Define New Test Object** option, see "Defining New Test Objects" on page 147.

Note: If you just want to restore the original test object description property set, while retaining any property values you have modified, you can use the **Restore mandatory property set** option. For more information, see "Restoring Default Mandatory Properties for a Test Object" on page 168.

To update identification properties from an object in your application:

1 In the object repository tree, select the test object whose description you want to update.

2 Select **Object** > **Update from Application** or click the **Update from Application** button. QuickTest is hidden, and the pointer changes into a pointing hand. For more information on using the pointing hand, see "Tips for Using the Pointing Hand" on page 167.

3 Find the object in your application whose properties you want to update in the object repository and click it. You must choose an object of the same object class as the test object you selected in the object repository tree.

The properties and property values for the selected object are updated in the object repository, according to the properties and values required to identify the object that were learned by QuickTest when you clicked the object in your application. Note that all properties and property values in the **Test object details** area are updated, together with the ordinal identifier and Smart Identification selections. Any object-specific comments that you may have entered are not removed.

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.

Restoring Default Mandatory Properties for a Test Object

The functionality described in this section is available in the Object Repository window for objects in the local object repository, and the Object Repository Manager for objects in shared object repositories.

You can restore the default properties for a selected test object. When you restore the default properties, it restores the mandatory property set defined for the selected object class, based on the settings that were set in the Object Identification dialog box at the time the object was learned. If you added or removed properties to or from the description, those changes are overwritten. However, if property values were defined or modified for any of the mandatory properties, these values are not modified when you choose this option. In addition, restoring the default mandatory property set does not change the values for the ordinal identifier or Smart Identification settings for the test object.

Note: The **Restore mandatory property set** option restores the object description property set to the mandatory properties that were defined for that class when your object was learned. If the mandatory properties in the Object Identification dialog box is currently different for this test object class than it was when your object was learned, and you want to use the new definition, you can use the **Update From Application** option, which relearns the object properties and values based on the current definitions in the Object Identifications dialog box. For more information, see "Updating Identification Properties from an Object in Your Application" on page 165

To restore the mandatory property set:

1 In the object repository tree, select the test object whose description you want to restore.

2 In the **Test object details** area, click the **Restore mandatory property set** button.

3 Click **Yes** to confirm the operation. The test object's description properties are restored to the mandatory property set for the selected object class at the time that the object was learned.

Renaming Test Objects

The functionality described in this section is available in the Object Repository window for objects in the local object repository, and the Object Repository Manager for objects in shared object repositories.

When an object changes in your application, or if you are not satisfied with the current name of a test object for any reason, you can change the name that QuickTest assigns to the stored object. You can also provide test objects with meaningful names to assist users in identifying them when using them in test steps.

For example, suppose you have a graphics application in which all the tools in the toolbar are saved as WinObjects in the object repository with the names ToolChild1, ToolChild2, ToolChild3, and so forth. You may want to rename all the buttons to their actual labels to make them easier to identify, for example, Color_Picker, Eraser, Airbrush, and so forth.

If you are working with a shared object repository, your change applies to all occurrences of the test object in all tests that use this shared object repository.

If you are working with a local object repository, your change applies to all occurrences of the test object in the selected action. If other actions in your test also include operations on the local test object, you should modify the test object's name in each relevant action.

When you modify the name of a test object in the local object repository, the name is automatically updated in both the Keyword View and the Expert View for all occurrences of the test object. When you modify the name of a test object in a shared repository, the name is automatically updated in all tests open on the same computer that use the object repository as soon as you make the change, even if you have not yet saved the object repository with your changes. If you close the object repository without saving your changes, the changes are rolled back in any open tests that were open at the time. Changes that are saved are also automatically updated in tests that use the object repository as soon as you open them. To load and view saved changes in a test or object repository that is currently open on a different computer, you must open the object repository or lock it for editing on your computer.

Tip: If you do not want to automatically update test object names in the Keyword View and Expert View for all occurrences of the test object, you can clear the **Automatically update test and component steps when you rename test objects** check box in the General pane of the Options dialog box (**Tools** > **Options** > **General** node). If you clear this option, you will need to manually change the test object names in all steps in which they are used, otherwise your test run will fail.

Note: If you rename test objects in a shared object repository and save the changes, when you open another test using the same shared object repository, that test updates the test object name in all of its relevant steps. This process may take a few moments. If you save the changes to the second test, the renamed steps are saved. However, if you close the second test without saving, then the next time you open the same test, it will again take a few moments to update the test object names in its steps.

To rename a test object:

In the object repository tree of the Object Repository window or Manager, select the test object that you want to rename and perform one of the following:

- Select **Edit** > **Rename** and enter the new name for the test object in the selected node in the tree. Then press ENTER or click anywhere else to remove the focus from the test object.
- Press F2 and enter the new name for the test object.
- In the **Name** box in the Object Properties pane, enter the new name for the test object. Then click anywhere else to remove the focus from the object. The name you assign to the test object must be unique within the same class and hierarchy in the object repository. Object names are not case-sensitive.

Adding Properties to a Test Object Description

The functionality described in this section is available in the Object Repository window for objects in the local object repository, and the Object Repository Manager for objects in shared object repositories.

You can add to the list of properties that QuickTest uses to identify an object. For each object class, QuickTest has a default property set that it uses for the object description for a particular test object. You can use the Add Properties dialog box to change the properties that are included in the test object description.

Note: You can also add any valid identification property to a test object description, even if it does not appear in the Add Properties dialog box. For more information, see "Defining New Identification Properties" on page 174.

Adding to the list of properties is useful when you want to create and run tests on an object that changes dynamically. An object may change dynamically if it is frequently updated, or if its property values are set using dynamic content (for example, from a database).

You can also change the properties that identify an object if you want to reference objects using properties that QuickTest did not learn automatically when it learned the object. For example, suppose you are testing a Web site that contains an archive of newsletters. The archive page includes a hypertext link to the current newsletter and additional hypertext links to all past newsletters. The text in the first hypertext link on the page changes as the current newsletter changes, but it always links to a page called **current.html**. Suppose you want to create a step in your test in which you always click the first hypertext link in your archive page. Since the news is always changing, the text in the hypertext link keeps changing. You need to modify how QuickTest identifies this hypertext link so that it can continue to find it.

The default properties for a Link object (hypertext link) are **text** and **HTML tag**. The text property is the text inside the link. The HTML tag property is always **A**, which indicates a link.

You can modify the default properties for a hypertext link for the learned object so that QuickTest can identify it by its destination page, rather than by the text in the link. You can use the **href** property to check the destination page instead of using the **text** property to check the link by the text in the link.

Tip: You can use the Object Spy at any time to view the native properties and values of the objects in the application you are testing, or the identification properties of the test objects that represent them. You open the Object Spy by choosing **Tools** > **Object Spy** or clicking the **Object Spy** toolbar button. For more information, see "Viewing Object Properties and Operations Using the Object Spy" on page 97.

Note: You can also modify the set of properties that QuickTest learns when it learns objects from a particular object class using the Object Identification dialog box. Such a change generally affects only those objects that QuickTest learns after you make the change. For more information, see "Configuring Object Identification" on page 105. You can also apply the changes you make in the Object Identification dialog box to the descriptions of all objects in an existing test using the **Update Run Mode** option. For more information, see "Updating a Test Using the Update Run Mode Option" on page 1125.

To add properties to a test object description:

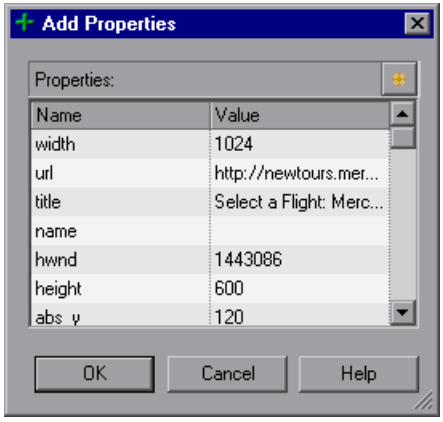
1 In the object repository tree of the Object Repository window or Manager, select the test object whose description you want to modify.

2 In the **Test object details** area, click the **Add description properties** button.

Tip: For a test object in the local object repository, you can also select the required test object and select **Edit** > **Step Properties** > **Object Properties**, click the **Add description properties** button, and then perform the following steps in the Add Properties dialog box.

The Add Properties dialog box opens listing the properties that can be used to identify the object (properties that are not already part of the test object description).

The value for each property is displayed in the **Value** column.



Notes:

- Values for all properties are displayed only if the application that contains the object is currently open. If the application is closed, only values for properties that were part of the object description when the object was learned are shown.
- You can resize the Add Properties dialog box to enable you to view long property values.
-  You can click the **Define new property** button to add valid identification properties to this properties list. For more information, see "Defining New Identification Properties" on page 174.

3 Select one or more properties to add to the test object description and click **OK**. You can also double-click a property to add it to the test object description. You can type the first letters of a property to highlight the first property in the list that matches the pattern.

Tip: After you add a new property to the object description, you can modify its value. For more information on modifying object property values, see "Specifying or Modifying Property Values" on page 163.

Defining New Identification Properties

The functionality described in this section is available in the Object Repository window for objects in the local object repository, and the Object Repository Manager for objects in shared object repositories.

You can add any valid identification property to a test object description, even if it does not appear in the Add Properties dialog box.

For example, suppose you want QuickTest to use a specific property to identify your object, but that property is not listed in the Add Properties dialog box. You can open the Add Properties dialog box and add that property to the list.

Tip: You can use the Properties tab of the Object Spy to view a complete list of valid identification properties for a selected object. You open the Object Spy by choosing **Tools** > **Object Spy** or clicking the **Object Spy** toolbar button. For more information, see "Viewing Object Properties and Operations Using the Object Spy" on page 97.

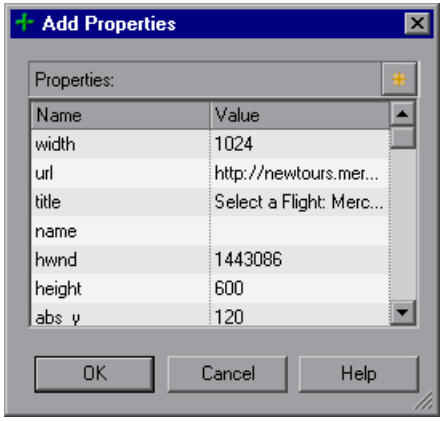
To define a new identification property:

1 In the object repository tree of the Object Repository window or Manager, select the test object for which you want to define a new property.

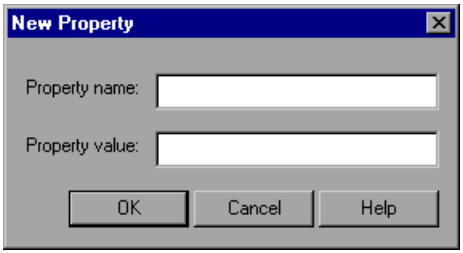
2 In the **Test object details** area, click the **Add description properties** button.

Tip: For a test object in the local object repository, you can also select the required test object and select **Edit** > **Step Properties** > **Object Properties**, click the **Add description properties** button, and then perform the following steps in the Add Properties dialog box.

The Add Properties dialog box opens.



3 Click the **Define new property** button. The New Property dialog box opens.



4 Specify a valid identification property:

- **Property name**. Enter the property name.
- **Property value**. Enter the value for the property.

Note: You must enter a valid identification property. If you enter an invalid property and then select it to be part of the object description, your run session will fail.

---

5 Click **OK** to add the property to the list and close the New Property dialog box. The new property is highlighted in the Add Properties dialog box.

6 Click **OK** while the new property is highlighted to include it in the object description and close the Add Properties dialog box.

Removing Properties from a Test Object Description

The functionality described in this section is available in the Object Repository window for objects in the local object repository, and the Object Repository Manager for objects in shared object repositories.

You can remove properties from the description of a test object if you no longer want them to be part of the description.

To remove a property from a test object description:

1 In the object repository tree of the Object Repository window or Manager, select the test object whose description you want to modify.

2 In the **Test object details** area, select one or more properties that you want to remove from the test object description.

Tip: For an object in the local object repository, you can also select the required test object and select **Edit** > **Step Properties** > **Object Properties**, and then perform the following steps in the Object Properties dialog box.

3 Click the **Remove selected description properties** button. The selected properties are removed from the test object description.

Specifying Ordinal Identifiers

The functionality described in this section is available in the Object Repository window for objects in the local object repository, and the Object Repository Manager for objects in shared object repositories.

An ordinal identifier assigns a numerical value to a test object that indicates its order or location relative to other objects with an otherwise identical description (objects that have the same values for all properties). This ordered value provides a backup mechanism that enables QuickTest to create a unique description to recognize an object when the defined properties are not sufficient to do so.

For more information on ordinal identifiers, see “Selecting an Ordinal Identifier” on page 113.

To specify an ordinal identifier:

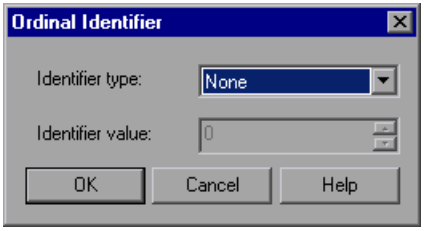
1 In the object repository tree of the Object Repository window or Manager, select the test object whose ordinal identifier you want to specify.

2 In the **Test object details** area, click in the cell to the right of the **Type, Value** cell under the **Ordinal identifier** row.

Tip: For an object in the local object repository, you can also select the required test object and select **Edit** > **Step Properties** > **Object Properties**, click in the cell to the right of the **Type, Value** cell under the **Ordinal identifier** row, and then perform the following steps in the Object Properties dialog box.

---

3 Click the browse button. The Ordinal Identifier dialog box opens.



4 In the **Identifier type** box, select one of the following options:

- **Location.** Indicates the order in which the object appears within the parent window, frame, or dialog box relative to other objects with an otherwise identical description.
- **Index.** Indicates the order in which the object appears in the application code relative to other objects with an otherwise identical description.
- **CreationTime** (Browser objects only). Indicates the order in which the browser was opened relative to other open browsers with an otherwise identical description. This identifier type is only available if more than one Browser object was open when the test object was learned.
- **None.** Does not specify an ordinal identifier. This is the default value if QuickTest did not learn an ordinal identifier.

5 In the **Identifier value** box, enter the numeric value of the ordinal identifier.

6 Click **OK**. The ordinal identifier appears in the relevant row of the **Test object details** area for the selected object.

6

Using Object Repositories in Your Test

This chapter explains how to use object repositories in your test. It describes how to use the Object Repository Window, manage shared repository associations, map repository parameter values, and create or modify test objects during a run session.

This chapter includes:

Understanding the Object Repository Window

The Object Repository window displays a tree of all test objects and all checkpoint and output objects in the selected action (including all local objects and all objects in any shared object repositories associated with the selected action).

For each object you select in the tree, the Object Repository window displays information on the object, its type, the repository in which it is stored, and its object details. Local objects are editable (black); shared objects are in read-only format (gray).

Note: Test objects of environments that are not installed with QuickTest will be displayed with a question mark icon in the object repository.

While the Object Repository window is open, you can continue using QuickTest, and you can continue modifying objects and object repositories. You can also resize the Object Repository window if needed. The Object Repository window reflects any changes you make to an associated object repository in realtime. For example, if you add objects to the local object repository, or if you associate an additional object repository with the current action, the Object Repository window immediately displays the updated content.

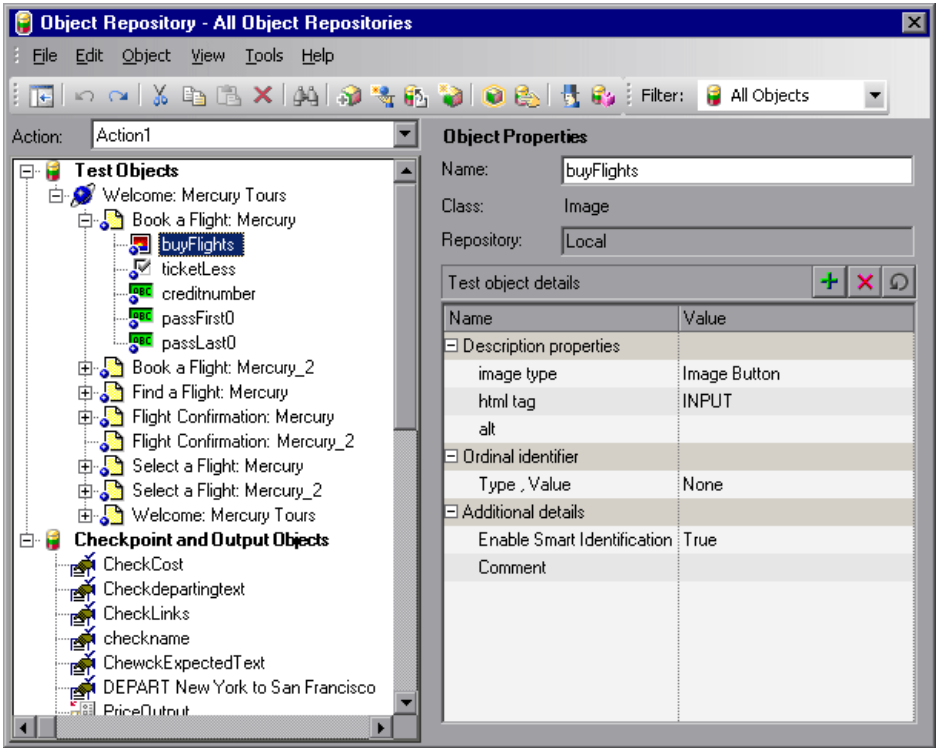
You can use the Object Repository window to view the object description of any object in the repository (in local and shared object repositories), to modify local objects and their properties, and to add test objects to your local object repository. You can also drag and drop test objects from the Object Repository window to your test. When you drag and drop a test object to your test, QuickTest inserts a step with the default operation for that test object in your test. Checkpoint and output objects cannot be dragged and dropped from the Object Repository window.

For example, if you drag and drop a button object to your test, a step is added to your test using the button object, with a **Click** operation (the default operation for a button object).

The Object Repository Window

Description	Enables you to manage identification properties and object repository associations for your action.
How to Access	➤ Click the **Object Repository** button  ➤ Double-click the repository in the Resources pane, or right-click it and choose **Open Repository** ➤ Right-click an action in the Test Flow pane and choose **Object Repository** ➤ Right-click an object in the repository in the Available Keywords pane and choose **Open Resource** ➤ Choose **Resources** > **Object Repository**
Learn More	**Conceptual overview:** "Understanding the Object Repository Window" on page 182 **Primary tasks:** ➤ "Adding Test Objects to a Local or Shared Object Repository" on page 136 ➤ "Copying, Pasting, and Moving Objects in the Object Repository" on page 150 ➤ "Deleting Objects from the Object Repository" on page 153 ➤ "Locating Objects" on page 154 ➤ "Maintaining Identification Properties" on page 162 **Additional related topics:** "Additional References" on page 189

Below is an image of the Object Repository window:



The Object Repository Window - Edit Toolbar

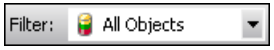
The Object Repository window Edit toolbar contains the following buttons:

Button	Name	Description
	**Compact View**	**Compact View** mode displays only the object repository tree, while **Full View** mode displays the object repository tree together with the object details area.
	**Full View**	
	**Undo**	All changes you make to a local object are automatically updated in all steps that use the local object as soon as you make the change. You can use the **Edit** > **Undo** and **Edit** > **Redo** menu options or **Undo** and **Redo** toolbar buttons to cancel or repeat your changes. After you save the current test, you cannot undo or redo operations that were performed before the save operation.
	**Redo**	
	**Cut**	Cuts the selected object from the object repository tree. For more information, see “Copying, Pasting, and Moving Objects in the Object Repository” on page 150.
	**Paste**	Pastes the object in the clipboard into the object repository tree as a child of the object selected in the tree. Bottom level objects cannot contain children. For more information, see “Copying, Pasting, and Moving Objects in the Object Repository” on page 150.
	**Copy**	Copies the selected object from the object repository tree into the clipboard. For more information, see “Copying, Pasting, and Moving Objects in the Object Repository” on page 150.
	**Delete**	Deletes the selected object from the object repository tree. For more information, see “Deleting Objects from the Object Repository” on page 153.
	**Find & Replace**	Finds and replaces an object in the object repository. For more information, see “Finding Objects in an Object Repository” on page 154.

Button	Name	Description
	**Add Objects to Local**	Adds an object to the local object repository. For more information, see "Adding Test Objects to a Local or Shared Object Repository" on page 136.
	**Update from Application**	Updates the identification properties from an object in the application. For more information, see "Updating Identification Properties from an Object in Your Application" on page 165.
	**Define New Test Objects**	Defines a new test object. For more information, see "Defining New Test Objects" on page 147.
	**Highlight in Application**	Highlights the selected object in the object repository tree, in the application. For more information, see "Highlighting an Object in Your Application" on page 157.
	**Locate in Repository**	Enables you to select an object in the application you are testing and highlight the test object in the object repository. For more information, see "Locating a Test Object in the Object Repository" on page 159.
	**Object Spy**	Enables you to view the native properties and operations of any object in an open application, as well as the test object hierarchy, identification properties, and operations of the test object that QuickTest uses to represent that object. For more information, see "The Object Spy Dialog Box" on page 100.
	**Associate Repositories**	Enables you to manage the shared object repository associations of your action. For more information, see "Managing Shared Object Repository Associations" on page 199.

The Object Repository Window - Filter Toolbar

The Filter toolbar contains the following options:

Option	Description
	You can use the Filter toolbar to filter the objects shown in the Object Repository window. You can choose to show objects that meet one of the following criteria: ➤ All objects in the selected action (all local objects and all objects in any shared object repositories associated with the selected action) ➤ Only the local objects in the selected action ➤ Only the objects in a specific shared object repository associated with the current action To filter the Object Repository window: In the **Filter** toolbar list, select one of the following options: ➤ **All Objects** ➤ **Local Objects** ➤ The name of a specific shared object repository associated with the current action The object repository tree is filtered to display only the objects from the location that you selected. The title bar of the Object Repository window indicates the current filter.

Object Repository Window Options

The Object Repository window contains the following options:

Option	Description
Action	Enables you to select the action whose objects you want to view.
Test Objects tree	Contains all test objects in the selected action (all local test objects and all test objects in any shared object repositories associated with the selected action). **Note:** If there are test objects in different associated object repositories with the same name, object class, and parent hierarchy, the object repository tree shows only the first one it finds based on the priority order defined. For information on object repository priorities, see "Associating Object Repositories with Actions" on page 446. You can filter the objects shown in the object repository tree. For more information, see "The Object Repository Window - Filter Toolbar" on page 187.
Checkpoint and Output Objects tree	Contains all the checkpoint and output objects in the selected action (all local checkpoint and output objects and all checkpoint and output objects in any shared object repositories associated with the selected action).
Name	The name that QuickTest assigns to the object. You can change the name of a object in the local object repository. For more information, see "Renaming Test Objects" on page 169.
Class	The class of the object.

Option	Description
Repository	The location (file name and path) of the object repository in which the object is located. If the object is located in the local object repository, **Local** is displayed.
Object details	Enables you to view the properties and property values used to identify a test object during a run session or the properties of a checkpoint or output object. You can also modify the object details for an object in the local object repository. For more information, see "Understanding the Object Details Area" on page 190. You can choose whether to show or hide the object details area. For more information, see "The Object Repository Window - Edit Toolbar" on page 185.

Additional References

Related Tasks	➤ "Mapping Repository Parameter Values" on page 202 ➤ "Exporting Local Objects to a Shared Object Repository" on page 193 ➤ "Copying an Object to the Local Object Repository" on page 195 ➤ "Renaming Test Objects" on page 169 ➤ You can drag and drop test objects from other locations. For more information, see "Understanding the Available Keywords Pane" on page 1165 and "Adding Test Objects to Your Test Using the Object Repository Manager" on page 225. ➤ You can modify the properties of a test object during a test run. For more information, see "Working with Test Objects During a Run Session" on page 206. ➤ You can view and modify object properties from other locations. For more information, see "Maintaining Identification Properties" on page 162.

Understanding the Object Details Area

The object details area in the lower right side of the Object Repository window enables you to view and modify the properties and property values used to identify an object during a run session or the properties of a checkpoint or output object.

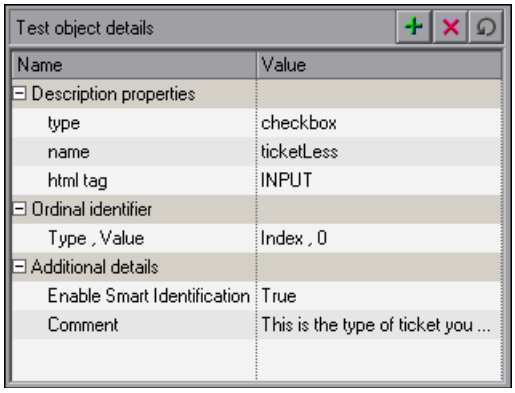
In the Object Repository window, objects in a shared object repository are displayed in the Object Properties pane (including the object details area) in read-only format. To modify objects in a shared object repository, open the shared object repository using the Object Repository Manager. For more information, see Chapter 7, "Managing Object Repositories." You can also modify an object in a shared object repository by copying to the local object repository and then modifying the local copy. For more information, see "Copying an Object to the Local Object Repository" on page 195.

Tips:

- You can view object properties and property values using the Object Properties dialog box. For more information, see "The Object Properties Dialog Box" on page 197.

- You can use the Object Spy at any time to view native or identification properties and values of the objects in the application you are testing. You open the Object Spy by choosing **Tools** > **Object Spy** or clicking the **Object Spy** toolbar button. For more information, see "Viewing Object Properties and Operations Using the Object Spy" on page 97.

You can modify test object details for objects saved in the local object repository.



The object details area contains the following items for test objects:

Item	Description
Description properties	The properties and property values used to identify the object during a run session. You can add and remove properties to or from the test object description. For more information, see “Adding Properties to a Test Object Description” on page 171. You can specify a property value as a constant, or you can parameterize the value. For more information, see “Specifying or Modifying Property Values” on page 163.
Ordinal identifier	A numerical value that indicates the object’s order or location relative to other objects with an otherwise identical description (objects that have the same values for all properties). For more information, see “Specifying Ordinal Identifiers” on page 177.

Item	Description
Additional details	Contains the following options: ➤ **Enable Smart Identification.** Enables you to select **True** or **False** to specify whether QuickTest should use Smart Identification to identify the test object during the run session if it is not able to identify the object using the test object description. **Note:** This option is available only if Smart Identification properties are defined for the test object's class in the Object Identification dialog box. For more information on Smart Identification, see "Configuring Smart Identification" on page 121. ➤ **Comment.** Enables you to add textual information about the test object.

For checkpoints and output objects, the object details area contains the checkpoint or output value object properties. The object details area enables you to modify these properties.

Tips:

- ➤ You can modify checkpoint and output value details for objects saved in the local object repository.
- ➤ You can copy an object from a shared object repository to the local object repository, and then modify it.

For more information, see:

- ➤ "Understanding the Checkpoint Properties Dialog Box" on page 508
- ➤ "Understanding the Image Checkpoint Properties Dialog Box" on page 512
- ➤ "The Bitmap Checkpoint Properties Dialog Box" on page 522
- ➤ "Understanding the Table Checkpoint Properties Dialog Box" on page 535
- ➤ "The Text / Text Area Checkpoint Properties Dialog Box" on page 557

- "Understanding the Database Checkpoint Properties Dialog Box" on page 581
- "Understanding the XML Checkpoint Properties Dialog Box" on page 607
- "About Outputting Values" on page 669
- The Web section of the *HP QuickTest Professional Add-ins Guide* (for Page and Accessibility checkpoints)

Exporting Local Objects to a Shared Object Repository

The functionality described in this section is available only when working in the Object Repository window.

You can export all of the test objects, checkpoint objects, and output value objects contained in an action's local object repository to a new shared object repository in the file system or to a Quality Center project (if QuickTest is connected to Quality Center). This enables you to make the local objects accessible to other actions.

You can choose to only export the local objects to a shared object repository, or to export and replace the local objects. The **Export and Replace Local Objects** option exports the local objects to a shared object repository, associates the new shared object repository with your action, and deletes the objects in the local object repository.

When you export local objects to a shared object repository, the parameters of any parameterized objects are converted to repository parameters using the same name as the source parameter. The default (mapped) value of each repository parameter is the corresponding source parameter. You can modify the mapping used within your action using the Map Repository Parameters dialog box (described in "Mapping Repository Parameter Values" on page 202). For more information on repository parameters, see Chapter 7, "Managing Object Repositories."

Tip: After you export the local objects, you can use the Object Repository Merge Tool to merge the test objects from the shared object repository containing the exported objects with another shared object repository. For more information, see Chapter 8, "Merging Shared Object Repositories."

To export local objects to a new shared object repository:

1 Open the test that has the local objects you want to export.

2 Open the Object Repository window by selecting **Resources** > **Object Repository** or clicking the **Object Repository** button.

3 In the Object Repository window, in the **Action** box, choose the action whose local objects you want to export.

4 Select **File** > **Export Local Objects,** or **File** > **Export and Replace Local Objects**. The Save Shared Object Repository dialog box opens.

5 In the sidebar, select the location in which you want to save the file, for example, File System or Quality Center Test Resources.

6 Browse to and select the folder in which you want to save the file.

7 In the **File name** box, enter a name for the file. Use a descriptive name that will help you easily identify the file. Do not use any of the following characters in the object repository name:
\ / : * " ? < > | '

If you save the object repository to Quality Center, the file path must not contain two consecutive semicolons (;;).

8 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**.

If you chose **Export Local Objects**, the local objects are exported to the specified shared object repository (a file with a **.tsr** extension). Your test continues to use the objects in the local object repository, and the new shared object repository is not associated with your test.

If you chose **Export and Replace Local Objects**, the new shared object repository (a file with a **.tsr** extension) is associated with your test, and the objects in the local object repository are deleted. The objects in the Object Repository window are read-only (gray), as they are now in a shared object repository. In the Object Properties section of the Object Repository window, the repository location indicates the path and filename of the new shared object repository instead of **Local**.

You can now use the new shared object repository like any other shared object repository.

Copying an Object to the Local Object Repository

The functionality described in this section is available only when working in the Object Repository window.

If you want to modify an object stored in a shared object repository, you can modify it using the Object Repository Manager, or you can modify it locally using the Object Repository window.

If you modify it using the Object Repository Manager, the changes you make will be reflected in all actions that use the shared object repository. If you make a local copy of the object and modify it in the Object Repository window, the changes you make will affect only the action in which you make the change. If you later modify the same object in the shared object repository, your changes will not affect the local copy of the object in your action.

When copying an object to the local object repository, consider the following:

- When you copy an object to the local object repository, its parent objects are also copied to the local object repository.
- If an object or its parent objects use unmapped repository parameters, you cannot copy the object to the local object repository. You must make sure that all repository parameters are mapped before copying an object to the local object repository.
- If an object or its parent objects are parameterized using one or more repository parameters, the repository parameter values are converted when you copy the object to the local object repository. For example, if the repository parameter is mapped to a Data Table parameter, the property is parameterized using a Data Table parameter. If the value is a constant value, the property receives the same constant value.
- If you are copying multiple objects to the local object repository, during the copy process you can choose to skip a specific object if it has unmapped repository parameters, or if it has mapped repository parameters whose values you do not want to convert. You can then continue copying the next object from your original selection.

To copy an object to the local object repository:

1. In the Object Repository window, select an object from a shared object repository that you want to copy to the local object repository. Objects in a shared object repository are colored gray. You can select more than one object to copy, as long as the selected objects have the same parent objects.
2. Select **Object** > **Copy to Local** or right-click the objects and select **Copy to Local**. The objects (and parent objects) are copied to the local object repository and are made editable.

The Object Properties Dialog Box

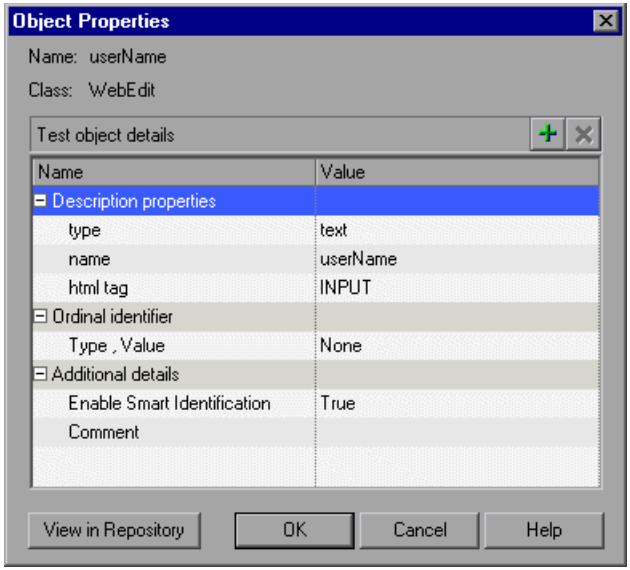
You can view identification properties and property values for objects in your test steps. You can also view identification properties and property values for objects in the Active Screen, regardless of whether the objects are stored in the object repository.

To view object properties and property values in your test:

➤ Click in the step of the object whose properties you want to view and choose **Edit > Step Properties > Object Properties.**

➤ In the Active Screen, right-click the object whose properties you want to view and choose **View / Add Object.**

The Object Properties dialog box opens.



Note: There are slight differences in the Object Properties dialog box, depending on whether the selected object is currently stored in the local object repository or a shared object repository associated with the current test. This section describes options shown in the dialog box for objects in the local object repository. For objects stored in a shared object repository the information is in read-only format.

The Object Properties dialog box shows the name and class of the selected object and enables you to:

- View the object's properties and property values—its description properties, ordinal identifier, and other settings.
- Modify the properties and property values used to identify the object (for objects that are stored in the local object repository). You modify the properties and values in the Object Properties dialog box in the same way as you modify the test object details in the Object Repository window. For more information, see "Maintaining Identification Properties" on page 162.
- Click the **View in Repository** button (for objects that are stored in the object repository) to open the Object Repository window and display the selected object in the object hierarchy.
- Click the **Add to Repository** button (for objects that are not stored in the object repository) to add the selected object to the local object repository.

Managing Shared Object Repository Associations

You can manage the shared object repository associations of a selected test using the Associate Repositories dialog box. The Associate Repositories dialog box enables you to associate one or more shared object repositories with one or more actions in a test. You can also remove object repository associations from selected actions, or from all actions in the test. For more information on shared object repository associations, see "Associating Object Repositories with Actions" on page 446.

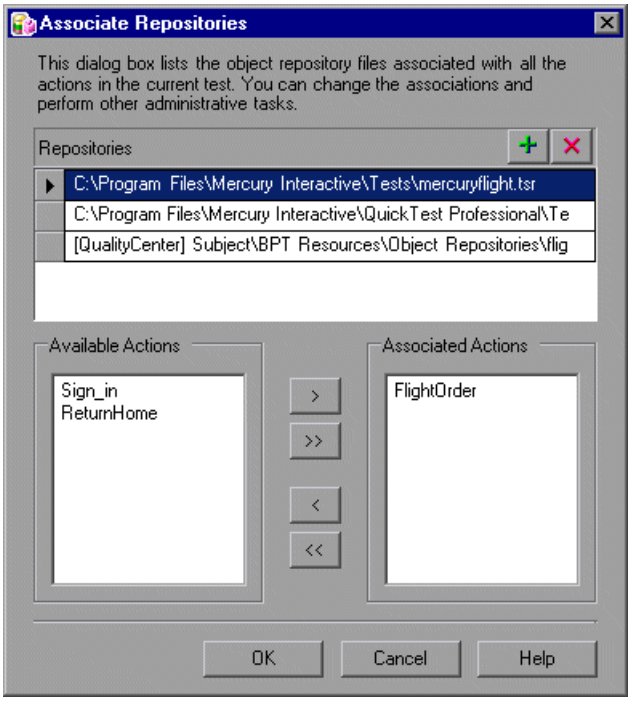
You can also associate, remove, prioritize, and view the properties of shared object repositories in the Resources pane. For more information, see "The Resources Pane" on page 1161.

To manage object repository associations in the Associate Repositories dialog box:

1 Perform one of the following:

- Choose **Resources** > **Associate Repositories**.
- In the Object Repository window, choose **Tools** > **Associate Repositories**.
-  In the Object Repository window, click the **Associate Repositories** button.

The Associate Repositories dialog box opens.



The Associate Repositories dialog box lists all the shared object repositories associated to each of the actions in the current test, and shows to which actions each repository is currently associated. You can add or remove object repositories from the list, and change the associations to actions in the test.

2 To add a shared object repository to the list so you can associate it to one or more actions in the current test, click the **Add Repository** button. The Open Shared Object Repository dialog box opens. In the sidebar, select the location of the object repository file, for example, File System or Quality Center Resources. Browse to and select the object repository file you want to open, and click Open. The new object repository is displayed at the bottom of the **Repositories** list.

3 To modify the name or path of an associated shared object repository, click a shared object repository in the **Repositories** list and then click the browse button to open a file selection dialog box in which you can select a different shared object repository. Alternatively, you can modify the shared object repository name or path directly in the **Repositories** list. The modified shared object repository remains associated with the same actions as the previous shared object repository.

4 To associate an object repository with one or more actions, or remove existing associations, select the object repository in the **Repositories** list, and then double-click the action names or select the action names and click the arrow buttons (> and <) to move them between the **Available Actions** and the **Associated Actions** lists.

Tip: Click the double arrow buttons (>> and <<) to move all the actions from one list to the other. Select multiple actions (using the SHIFT and/or CONTROL keys) and click the arrow buttons (> and <) to move only the selected actions from one list to the other.

Note: You cannot define the priorities of the object repositories associated with an action using the Associate Repositories dialog box. You prioritize the object repositories using the Associated Repositories tab of the Action Properties dialog box. For more information, see "Associating Object Repositories with Actions" on page 446.

5 To remove an object repository from the list and thereby remove all of its associations to any actions in the current test, select the object repository and click the **Remove Repository** button.

6 Click **OK**. The changes you made to the object repository associations are applied. You can view the new associations and change the object repository priorities in the Associated Repositories tab of the Action Properties dialog box. For more information, see "Associating Object Repositories with Actions" on page 446.

Mapping Repository Parameter Values

You can map repository parameters that are used in shared object repositories that are associated with your action. Mapping a repository parameter to a value or parameter specifies the property values used to identify the test object during a run session. You can specify that the property value is taken from a constant value, or parameterize it using a Data Table, random number, environment, or test parameter.

You can map each repository parameter as required in each test that has an associated object repository containing repository parameters. For example, in one test you may want to retrieve the username object's text property value from an environment variable parameter, and in another test you may want the same object property value to use a constant value or a Data Table parameter.

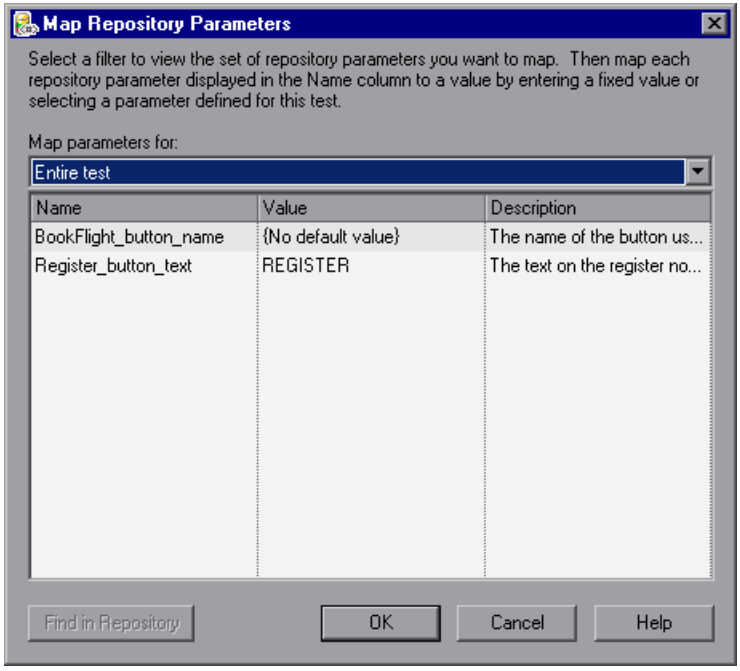
Before you map repository parameters, if you have more than one repository parameter with the same name in different shared object repositories that are associated with the same test, the repository parameter from the shared object repository with the highest priority (as defined in the shared object repositories list) is used. After you map repository parameters, QuickTest uses the mappings you defined. In addition, changing the priority or default values has no effect after the parameters are mapped.

When you open a test that uses an object repository with an object property value that is parameterized using a repository parameter with no default value, an indication that there is a repository parameter that needs mapping is displayed in the Missing Resources pane. You can then map the repository parameter as needed in the test. You can also map repository parameters that have default values, and change mappings for repository parameters that are already mapped.

If you do not map a repository parameter, the default value that was defined with the parameter, if any, is used during the action run. If the parameter is unmapped, meaning no default value was specified for it, the test run may fail if a test object cannot be identified because it has an unmapped parameter value.

To map repository parameter values:

1 Choose **Resources** > **Map Repository Parameters**. The Map Repository Parameters dialog box opens.



Tip: If you have unmapped repository parameters (repository parameters without a default value) in your test, you can also open this dialog box by double-clicking the **Repository Parameters** row in the Missing Resources pane. For more information, see Chapter 41, "Handling Missing Resources."

The Map Repository Parameters dialog box contains the following options:

Option name	Description
Map parameters for filter	Enables you to filter the list of parameters that is displayed. You can choose to display: ➤ **All unmapped parameters.** Displays all of the parameters in your test with unmapped values. ➤ **Entire test.** Displays all of the parameters in your test (with mapped or unmapped values). ➤ **<Action name>.** (For example, LogIn) Displays all of the parameters in the specified action (with mapped or unmapped values).
Name column	The name of the repository parameter.
Value column	The parameter's current value, if any. This column shows either the new value you defined, or the default value that was defined when the parameter was created. If no default value was defined, then the parameter is currently unmapped, and the text **{No default value}** is shown. You can perform one of the following: ➤ Enter a new constant value. ➤ Parameterize the value by clicking in the **Value** cell of the relevant parameter and then clicking the parameterization button  . ➤ Reset a parameter to its default value by clicking in the **Value** cell of the relevant parameter and then clicking the **Reset to Default Value** button  .
Description column	A textual description of the parameter, if any.
Find in Repository button	Opens the Object Repository window and highlights the first test object in the object repository tree that uses the selected repository parameter. You can click this button again to find the next occurrence of the selected parameter, and so forth.

Note: The repository parameter names, default values, and descriptions are defined in the Manage Repository Parameters dialog box. In addition, the names and descriptions can only be modified there. For more information, see "Managing Repository Parameters" on page 229.

---

2 Click the **Map parameters for** arrow to select the list of parameter groups for which you want to define values. You can choose to display:

- **All unmapped parameters**. Displays all of the parameters in your test with unmapped values.
- **Entire test.** Displays all of the parameters in your test (with mapped or unmapped values).
- **<Action name>**. (For example, LogIn) Displays all of the parameters in the specified action (with mapped or unmapped values).

3 Click in the **Value** cell of the parameter you want to map. You can choose to map the value in one of the following ways:

- Enter a new constant value or modify an existing constant value by typing directly in the **Value** cell. You can also enter a constant value in the Value Configuration Options dialog box by clicking the parameterization button. For information on using this dialog box, see "Configuring a Selected Value" on page 760.

- Parameterize the value by clicking the parameterization button. The Value Configuration Options dialog box opens. You can parameterize the value using a Data Table (Global sheet only), random number, environment, or test parameter. For information on using this dialog box, see "Configuring a Selected Value" on page 760.

- Restore the default value by clicking the **Clear Default Value** button. The default value, if any, that was defined in the Add Repository Parameter dialog box is displayed in the cell. For information on the Add Repository Parameter dialog box, see "Adding Repository Parameters" on page 230.

4 Repeat step 3 for any additional parameter values that you want to map. Then click **OK** to close the Map Repository Parameter dialog box.

Working with Test Objects During a Run Session

The first time QuickTest encounters an object during a run session, it creates a temporary version of the test object for that run session. QuickTest uses the object description to create this temporary version of the object. For the remainder of the test, QuickTest refers to the temporary version of the test object rather than to the test object in the object repository.

Note: The Object Repository window is read-only during record and run sessions.

Creating Test Objects During a Run Session

You can use programmatic descriptions to create temporary versions of test objects that represent objects from your application. You can perform operations on those objects without referring to the object repository. For example, suppose an edit box was added to a form on your Web site. You can use a programmatic description to add a statement in the Expert View or in a user-defined function that enters a value in the new edit box. QuickTest could then identify the object even though the object was never added to the object repository. For more information on programmatic descriptions, see "Using Programmatic Descriptions" on page 863.

Modifying Identification Properties During a Run Session

You can modify the properties of the temporary version of the object during the run session without affecting the permanent values in the object repository by adding a SetTOProperty statement in the Keyword View, Expert View, or in a user-defined function.

Use the following syntax for the SetTOProperty method:

Object(*description***).SetTOProperty** *Property, Value*

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

7

Managing Object Repositories

The Object Repository Manager enables you to manage all of the shared object repositories used in your organization from a single, central location, including adding and defining objects, modifying objects and their descriptions, parameterizing repositories to make them more generic, maintaining and organizing repositories, merging repositories, and importing and exporting repositories in XML format.

This chapter includes:

About Managing Object Repositories

The Object Repository Manager enables you to create and maintain shared object repositories. You can work with object repositories saved both in the file system and in a Quality Center project.

Each object repository contains the information that enables QuickTest to identify the objects in your application. QuickTest enables you to maintain the reusability of your tests by storing all the information regarding your test objects in a shared object repository. When objects in your application change, the Object Repository Manager provides a single, central location in which you can update test object information for multiple tests.

Note: Instead of, or in addition to, shared object repositories, you can choose to store all or some of the objects in a local object repository for each action. For more information on local object repositories, see Chapter 5, "Managing Test Objects in Object Repositories."

If one or more of the property values of an object in your application differ from the property values QuickTest uses to identify the object, your test may fail. Therefore, when the property values of objects in your application change, you should modify the corresponding identification property values in the corresponding object repository so that you can continue to use your existing tests.

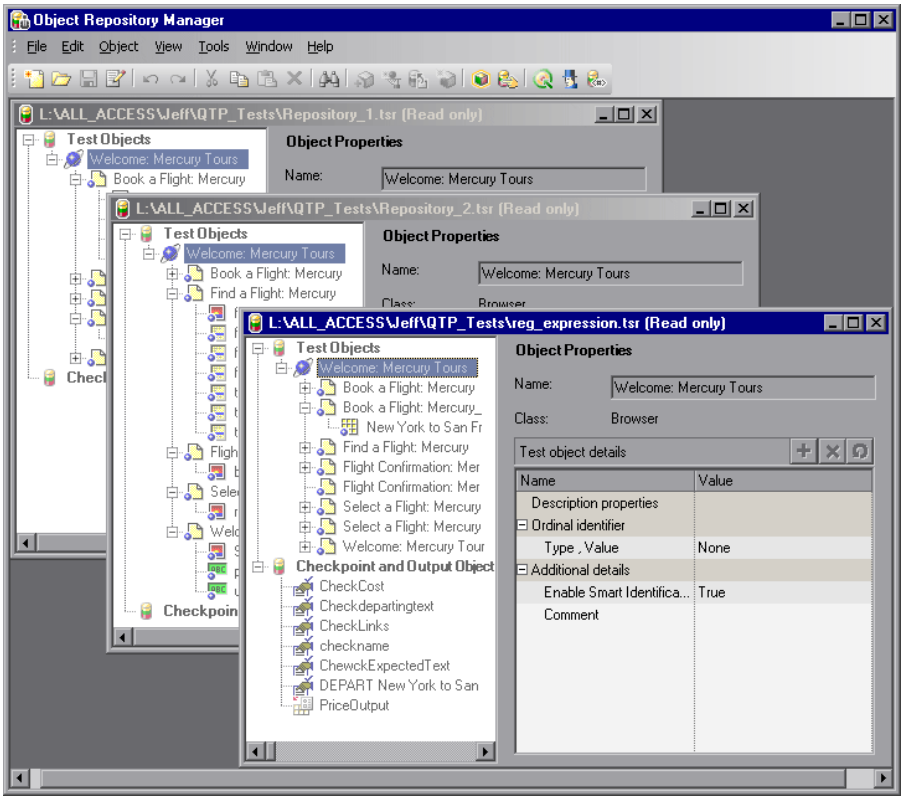
If an object with the same name and description is located in both the local object repository and in a shared object repository that is associated with the same action, the action uses the local object definition. If an object with the same name and description is located in more than one shared object repository, and these shared object repositories are all associated with the same action, QuickTest uses the object definition from the first occurrence of the object, according to the order in which the shared object repositories are associated with the action. For more information on associating shared object repositories, see "Associating Object Repositories with Actions" on page 446.

You can use the same shared object repository with multiple actions. You can also use multiple object repositories with each action. In addition, you can save objects directly with an action in a local object repository. This enables them to be accessed only from that action. If your shared object repositories are stored in Quality Center, you can apply version control to them. For more information, see "Managing Assets Using Version Control" on page 1479.

You can modify objects in a shared object repository using the Object Repository Manager, as described in this chapter. You can modify objects stored in a local object repository using the Object Repository window. For information on the Object Repository window, see Chapter 5, "Managing Test Objects in Object Repositories."

The Object Repository Manager

You open the Object Repository Manager by choosing **Resources** > **Object Repository Manager**. The Object Repository Manager enables you to open multiple shared object repositories and modify them as needed. You can open shared object repositories both from the file system and from a Quality Center project.



Tip: While the Object Repository Manager is open, you can continue working with other QuickTest windows.

You can open as many shared object repositories as you want. Each shared object repository opens in a separate document window. You can then resize, maximize, or minimize the windows to arrange them as you require to copy, drag, and move objects between different shared object repositories, as well as perform operations on a single object repository. For more information on the details shown in the shared object repository windows, see "Understanding the Shared Object Repository Windows" on page 215.

You open shared object repositories from the Open Shared Object Repository dialog box. In this dialog box, the **Open in read-only mode** check box is selected, by default. If you clear this check box, the shared object repository opens in editable mode. Otherwise, the shared object repository opens in read-only mode and you must click the **Enable Editing** button to modify it. For more information, see "Editing Object Repositories" on page 224.

When you choose a menu item or click a toolbar button in the Object Repository Manager, the operation you select is performed on the shared object repository whose window is currently active (in focus). The name and file path of the shared object repository is shown in the title bar of the window. For more information on the Object Repository Manager toolbar buttons, see "Using the Object Repository Manager Toolbar" on page 212.

If QuickTest is connected to a Quality Center project with version control enabled, you can view and manage versions of your shared object repositories, view comparisons of two shared object repository versions, and view baseline history. For more information, see "Managing Assets Using Version Control" on page 1479 and "Viewing and Comparing Versions of QuickTest Assets" on page 1461.

Many of the shared object repository operations you can perform in the Object Repository Manager are done in a similar way to how you modify objects stored in a local object repository (using the Object Repository window). For this reason, many of the procedures are actually described in Chapter 5, "Managing Test Objects in Object Repositories." Most of the procedures apply equally to the Object Repository Manager and the Object Repository window, but the windows and options may differ slightly.

Using the Object Repository Manager Toolbar

You can access frequently performed operations using the Object Repository Manager toolbar. The Object Repository Manager toolbar contains the following buttons:

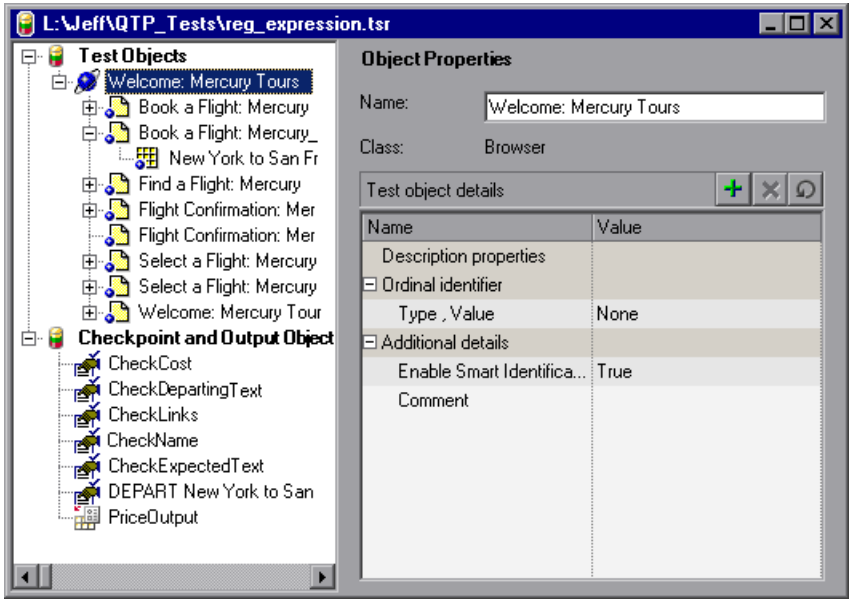
Button	Description
	Enables you to create a new shared object repository. For more information, see "Creating New Object Repositories" on page 217.
	Enables you to open a shared object repository from the file system or from Quality Center. For more information, see "Opening Object Repositories" on page 217.
	Enables you to save the active shared object repository to the file system or to Quality Center. For more information, see "Saving Object Repositories" on page 219.
	Enables you to edit the active shared object repository, by making the shared object repository editable. For more information, see "Editing Object Repositories" on page 224.
	Enables you to undo the previous operation performed in the active shared object repository. You do this in the same way as in a local object repository. For more information, see "Copying, Pasting, and Moving Objects in the Object Repository" on page 150.
	Enables you to redo the operation that was previously undone in the active shared object repository. You do this in the same way as in a local object repository. For more information, see "Copying, Pasting, and Moving Objects in the Object Repository" on page 150.
	Enables you to cut the selected item or object in the active shared object repository. You do this in the same way as in a local object repository. For more information, see "Copying, Pasting, and Moving Objects in the Object Repository" on page 150.
	Enables you to copy the selected item or object to the Clipboard in the active shared object repository. You do this in the same way as in a local object repository. For more information, see "Copying, Pasting, and Moving Objects in the Object Repository" on page 150.

Button	Description
	Enables you to paste the data from the Clipboard to the active shared object repository. You do this in the same way as in a local object repository. For more information, see “Copying, Pasting, and Moving Objects in the Object Repository” on page 150.
	Enables you to delete the selected item or object in the active shared object repository. You do this in the same way as in a local object repository. For more information, see “Deleting Objects from the Object Repository” on page 153.
	Enables you to find an object, property, or property value in the active shared object repository. You can also find and replace specified property values. You do this in the same way as in a local object repository. For more information, see “Finding Objects in an Object Repository” on page 154.
	Enables you to add objects to the active shared object repository. You do this in the same way as in a local object repository. For more information, see “Adding Test Objects to a Local or Shared Object Repository” on page 136.
	Enables you to update identification properties in the active shared object repository according to the actual properties of the object in your application. You do this in the same way as in a local object repository. For more information, see “Updating Identification Properties from an Object in Your Application” on page 165.
	Enables you to define a test object that does not yet exist in your application and add it to the active shared object repository. You do this in the same way as in a local object repository. For more information, see “Defining New Test Objects” on page 147.
	Enables you to select an object in the active shared object repository and highlight it in your application. You do this in the same way as in a local object repository. For more information, see “Highlighting an Object in Your Application” on page 157.
	Enables you to select an object in your application and highlight it in the active shared object repository. You do this in the same way as in a local object repository. For more information, see “Locating a Test Object in the Object Repository” on page 159.

Button	Description
	Enables you to connect to Quality Center to work with object repository files stored in a Quality Center project. You can connect to Quality Center from the main QuickTest window or from the Object Repository Manager. For more information, see “Connecting to and Disconnecting from Quality Center” on page 1418.
	Opens the Object Spy dialog box, enabling you to view the native properties and operations of any object in an open application, as well as the test object hierarchy, identification properties, and operations of the test object that QuickTest uses to represent that object. For more information, see “Viewing Object Properties and Operations Using the Object Spy” on page 97.
	Enables you to add, edit, and delete repository parameters in the active shared object repository. For more information, see “Managing Repository Parameters” on page 229.

Understanding the Shared Object Repository Windows

Each shared object repository that you open in the Object Repository Manager is displayed in a standalone document window. Each shared object repository window displays a tree of all the objects in the object repository, together with object information for the selected object.



For each object you select in the tree, the Object Repository window displays information about the selected object. You can view the object description of any object in the shared object repository, modify objects and their properties, and add test objects to the shared object repository.

Notes:

- You cannot add checkpoint or output value objects to a shared object repository via the Object Repository Manager.
- Test objects of environments that are not installed with QuickTest are displayed with a question mark icon in the test object tree.

For more information, see "Managing Objects in Shared Object Repositories" on page 222 and "Modifying Object Details" on page 234.

Each object repository window contains the following information:

Information	Description
Object Repository tree	Located on the left side of the Object Repository window. Contains all objects in the shared object repository.
Name	Specifies the name that QuickTest assigns to the selected object. You can change the object name. For more information, see "Renaming Test Objects" on page 169.
Class	Specifies the class of the selected object.
Object details	Located on the lower right side of the Object Repository window. Enables you to view the properties and property values used to identify a test object during a run session or the properties of a checkpoint or output object. For more information, see "Modifying Object Details" on page 234.

Note: Even when steps containing an object are deleted from your action, the objects remain in the object repository. You can delete objects from a shared object repository using the Object Repository Manager, in much the same was as you delete objects from a local object repository. For more information, see "Deleting Objects from the Object Repository" on page 153.

Working with Object Repositories

You can use the Object Repository Manager to create new object repositories, open and modify existing object repositories, and save and close them when you are finished.

Creating New Object Repositories

The functionality described in this section is available only when working in the Object Repository Manager.

You can create a new object repository, add objects to it, and then save it. You can then associate one or more actions with the object repository from within QuickTest. For more information on associating shared object repositories, see "Associating Object Repositories with Actions" on page 446.

To create a new object repository:

In the Object Repository Manager, select **File** > **New** or click the **New** button. A new object repository opens. You can now add objects to it, modify it, and save it. For more information, see "Managing Objects in Shared Object Repositories" on page 222 and "Saving Object Repositories" on page 219.

Opening Object Repositories

The functionality described in this section is available only when working in the Object Repository Manager.

You can open existing object repositories to view or modify them. You can open object repositories from the file system or from a Quality Center project.

You connect to a Quality Center project either from QuickTest or from the Object Repository Manager by choosing **File** > **Quality Center Connection** or clicking the **Quality Center Connection** button. For more information on connecting to Quality Center, see "Connecting to and Disconnecting from Quality Center" on page 1418.

Note for users of previous QuickTest versions:

When you open an object repository that is stored in the file system and was created using a version of QuickTest earlier than version 9.0, QuickTest converts it to the current format when you make it editable.

If the object repository contains test objects from add-ins, the relevant add-in must be installed to convert the object repository to the current format. Otherwise, you can open it only in read-only format.

If you do not want to convert the object repository, you can view it in read-only format. After the file is converted and you save it, you cannot use it with earlier versions of QuickTest.

To open an object repository:

1 In the Object Repository Manager, select **File** > **Open** or click the **Open** button. The Open Shared Object Repository dialog box opens.

2 In the sidebar, select the location of the object repository file, for example, File System or Quality Center Test Resources. Browse to and select the object repository file you want to open, and click **Open**. The object repository opens.

By default, the object repository opens in read-only mode. You can open it in editable format by clearing the **Open in read-only mode** check box in the Open Shared Object Repository dialog box. You can also enable editing for an object repository as described in "Editing Object Repositories" on page 224.

If the object repository is editable, you can add objects to it, modify it, and save it. For more information, see "Managing Objects in Shared Object Repositories" on page 222 and "Saving Object Repositories" on page 219.

Tip: You can also open an object repository from the **Recent Files** list in the **File** menu.

Saving Object Repositories

The functionality described in this section is available only when working in the Object Repository Manager.

After you finish creating or modifying an object repository, you should save it. When you modify an object repository, an asterisk (*) is displayed in the title bar until the object repository is saved.

You can save an object repository to the file system or to a Quality Center project (if you are connected to a Quality Center project). You connect to a Quality Center project either from QuickTest or from the Object Repository Manager by choosing **File** > **Quality Center Connection** or clicking the **Quality Center Connection** button. For more information on connecting to Quality Center, see "Connecting to and Disconnecting from Quality Center" on page 1418.

All changes you make to an object repository are automatically updated in all tests open on the same computer that use the object repository as soon as you make the change—even if you have not yet saved the object repository with your changes. If you close the object repository without saving your changes, the changes are rolled back in any open tests that were open at the time.

When you open a test on the same computer on which you modified the object repository, the test is automatically updated with all saved changes made in the associated object repository. To see saved changes in a test or repository open on a different computer, you must open the test or object repository file or lock it for editing on your computer to load the changes.

To save an object repository:

1. Make sure that the object repository you want to save is the active window.
2. 
Select **File** > **Save** or click the **Save** button. If the file has already been saved, the changes you made are saved. If the file has not yet been saved, the Save Shared Object Repository dialog box opens.
3. In the sidebar, select the location in which you want to save the file, for example, File System or Quality Center Test Resources.
4. Browse to and select the folder in which you want to save the file.

5 In the **File name** box, enter a name for the file. Use a descriptive name that will help you easily identify the file. Do not use any of the following characters in the object repository name:
\ / : * " ? < > | '

If you save the object repository to Quality Center, the file path must not contain two consecutive semicolons (;;).

6 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.

QuickTest saves the object repository with a **.tsr** extension in the specified location and displays the object repository name and path in the title bar of the repository window.

Closing Object Repositories

The functionality described in this section is available only when working in the Object Repository Manager.

After you finish modifying or using an object repository, you should close it. While an object repository is being edited, it is locked so that it cannot be modified by others. When you close the object repository, it is automatically unlocked. You can also choose to close all open object repositories.

Note: If you close QuickTest, the Object Repository Manager also closes. If you have made changes that are not yet saved, you are prompted to do so before the Object Repository Manager closes.

To close an object repository:

1 Make sure that the object repository you want to close is the active window.

2 Select **File** > **Close** or click the **Close** button in the object repository window's title bar. The object repository is closed and is automatically unlocked. If you have made changes that are not yet saved, you are prompted to do so before the file closes.

To close all open object repositories:

Select **File** > **Close All Windows**, or **Window** > **Close All Windows**. All open object repositories are closed and are automatically unlocked. If you have made changes that are not yet saved, you are prompted to do so before the files close.

Managing Objects in Shared Object Repositories

You can modify your shared object repositories in a variety of ways to either prepare them for initial use or update them throughout the testing process. You can add and modify objects and object properties in a shared object repository, copy or move objects from one object repository to another, drag objects to a different location in the hierarchy, delete objects, and rename objects. You can also drag and drop test objects from the Object Repository manager to your test. When you modify a shared object repository, an asterisk (*) is displayed in the title bar until the object repository is saved.

The following are tips and guidelines for working with the Object Repository Manager:

- You can use the **Edit** > **Undo** and **Edit** > **Redo** options or **Undo** and **Redo** buttons to cancel or repeat your changes as necessary. The **Undo** and **Redo** options are related to the active document. When you save an object repository, you cannot undo and redo operations that were performed on that file before the save operation.
- If you opened the object repository in read-only mode, you must enable editing for the object repository before you can modify it. This locks the object repository and prevents it from being modified simultaneously by multiple users.
- All changes you make to an object repository are automatically updated in all tests open on the same computer that use the object repository as soon as you make the change—even if you have not yet saved the object repository with your changes.

  If you close the object repository without saving your changes, the changes are rolled back in any open tests that were open at the time.
- When you open a test on the same computer on which you modified the object repository, the test is automatically updated with all saved changes made in the associated object repository. To see saved changes in a test or repository open on a different computer, you must open the test or object repository file or lock it for editing on your computer to load the changes.

- You can also modify a shared object repository by merging it with another shared object repository. When you merge two shared object repositories, a new shared object repository is created, containing the content of both object repositories. If you merge a shared object repository with a local object repository, the shared object repository is updated with the content of the local object repository. For more information, see Chapter 8, "Merging Shared Object Repositories."
- After making sure that your shared object repository is editable, and that it is the active window, you can modify it in the same way as you modify a local object repository. In addition to adding objects to a shared object repository in the same manner as to a local repository, you can also add objects to a shared object repository using the **Navigate and Learn** option.

  For more information, see:

- "Editing Object Repositories" on page 224
- "Adding Test Objects to Your Test Using the Object Repository Manager" on page 225
- "Adding Test Objects to a Local or Shared Object Repository" on page 136
- "Adding Test Objects Using the Navigate and Learn Option" on page 225
- "Copying, Pasting, and Moving Objects in the Object Repository" on page 150
- "Deleting Objects from the Object Repository" on page 153

Editing Object Repositories

The functionality described in this section is available only when working in the Object Repository Manager.

When you open an object repository, it is opened in read-only mode by default. You can open it in editable format by clearing the **Open in read-only mode** check box in the Open Shared Object Repository dialog box when you open it.

If you opened the object repository in read-only mode, you must enable editing for the object repository before you can modify it. You do not need to enable editing for an object repository if you only want to view it or copy objects from it to another object repository.

When you enable editing for an object repository, the object repository is locked so that it cannot be modified by other users. To enable other users to modify the object repository, you must first unlock it (by disabling edit mode, or by closing it). If an object repository is already locked by another user, if it is saved in read-only format, or if you do not have the permissions required to open it, you cannot enable editing for it.

Note for users of previous QuickTest versions: If you want to edit an object repository stored in the file system, and the object repository was created using a version of QuickTest earlier than version 9.0, QuickTest must convert it to the current format before you can edit it. If you do not want to convert it, you can view it in read-only format. After the file is converted and saved, you cannot use it with earlier versions of QuickTest.

To enable editing for an object repository:

1 Make sure that the object repository you want to edit is the active window.

2 Select **File** > **Enable Editing** or click the **Enable Editing** button. The object repository becomes editable.

Adding Test Objects to Your Test Using the Object Repository Manager

The functionality described in this section is available in the Object Repository window for objects in the local object repository, and the Object Repository Manager for objects in shared object repositories.

You can drag and drop test objects from the Object Repository Manager to your test. When you drag and drop a test object to your test, QuickTest inserts a step with the default operation for that test object in your test. You cannot drag and drop checkpoint or output objects from the Object Repository Manager.

For example, if you drag and drop a button object to your test, a step is added to your test using the button object, with a **Click** operation (the default operation for a button object).

You can also drag and drop test objects from other locations. For more information, see:

➤ "Understanding the Available Keywords Pane" on page 1165

➤ "The Object Repository Window" on page 183

Adding Test Objects Using the Navigate and Learn Option

The functionality described in this section is available only when working in the Object Repository Manager.

The **Navigate and Learn** option enables you to add multiple test objects to a shared object repository while navigating through your application.

Each time you select a window to learn, the selected window and its descendant objects are added to the active shared object repository according to a predefined object filter. You can change the object filter definitions at any time to meet your requirements. The object filter is used for both the **Navigate and Learn** option and the **Add Objects** option. The settings you define are used in both places when QuickTest learns objects. For more information on modifying the filter definitions, see "Understanding the Define Object Filter Dialog Box" on page 144.

Note: The Navigate and Learn option is not supported for environments with mixed hierarchies (object hierarchies that include objects from different environments), for example, Browser("Homepage").Page("Welcome").AcxButton("Save") or Dialog("Edit").AcxEdit("MyEdit"). To add objects within mixed hierarchies, use other options, as described in "Adding Test Objects to a Local or Shared Object Repository" on page 136.

You can use the following keyboard shortcuts when learning objects using the **Navigate and Learn** option:

- **Learn Focused Window.** ENTER
- **Define Object Filter.** CTRL+F
- **Help.** F1
- **Return to Object Repository Manager.** ESC

Note: Minimized windows are not learned when using the **Navigate and Learn** option.

To add test objects using the Navigate and Learn option:

1. In the Object Repository Manager, make sure that the object repository to which you want to add objects is the active window and that it is editable.
2. Select **Object** > **Navigate and Learn** or press F6. The **Navigate and Learn** toolbar opens.

Note: If this is the first time you are adding objects to the object repository, you may want to change the filter definitions before you continue. You can view the current filter definitions in the **Define Object Filter** button tooltip (displayed in parentheses after the button name). You can change the filter definitions at any time by clicking the **Define Object Filter** button or pressing CTRL+F. For more information, see "Understanding the Define Object Filter Dialog Box" on page 144.

3. Click the parent object (for example, Browser, Dialog, Window) you want to add to the object repository to focus it. The **Learn** button in the toolbar is enabled.
4. Click the **Learn** button or focus the **Navigate and Learn** toolbar and press ENTER. A flashing highlight surrounds the focused window and the object and its descendants are added to the object repository according to the defined filter.
5. Navigate in your application to the next window you want to add and then repeat step 4.
6. When you finish adding the required objects to the object repository, click the **Close** button in the **Navigate and Learn** toolbar or press ESC. The **Navigate and Learn** toolbar closes and the Object Repository Manager is redisplayed, showing the objects you just added to the shared object repository.

Working with Repository Parameters

Repository parameters enable you to specify that certain property values should be parameterized, but leave the actual parameterization to be defined in each test that is associated with the object repository that contains the parameterized identification property values.

Repository parameters are useful when you want to create and run tests on an object that changes dynamically. An object may change dynamically if it is frequently updated in the application, or if its property values are set using dynamic content, for example, from a database.

For example, you may have a button whose text property value changes in a localized application depending on the language of the user interface. You can parameterize the name property value using a repository parameter, and then in each test that uses the object repository you can specify the location from which the property value should be taken. For example, in one test that uses this object repository you can specify that the property value comes from an environment variable, in another test it can come from the Data Table, and in a third test you can specify it as a constant value.

You define all the repository parameters for a specific object repository using the Manage Repository Parameters dialog box. You define each repository parameter together with an optional default value and meaningful description. For more information, see "Managing Repository Parameters" on page 229.

When you open a test that uses an object repository with a repository parameter that has no default value, an indication that there is a repository parameter that needs mapping is displayed in the Missing Resources pane. You can then map the repository parameter as needed in the test. You can also map repository parameters that have default values, and change mappings for repository parameters that are already mapped. For more information on mapping repository parameters, see "Handling Unmapped Shared Object Repository Parameter Values" on page 1194.

Managing Repository Parameters

The functionality described in this section is available only when working in the Object Repository Manager.

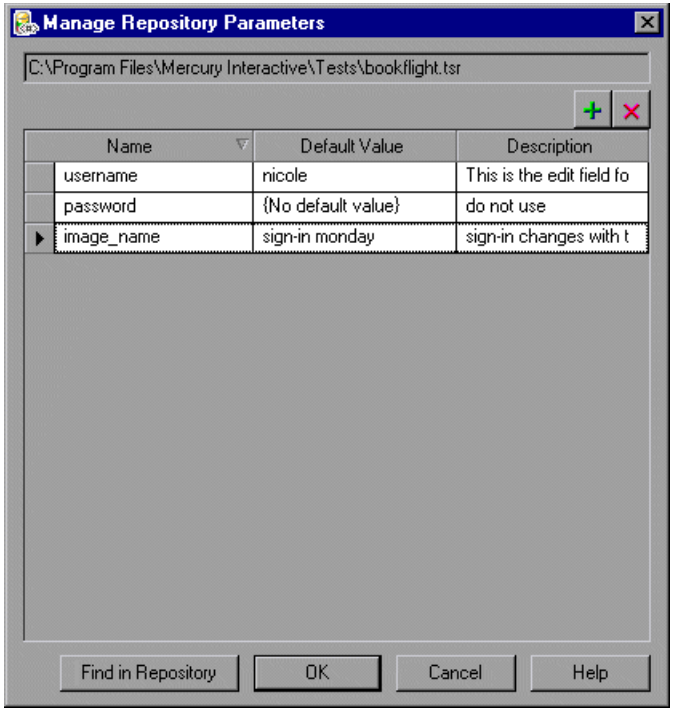
The Manage Repository Parameters dialog box enables you to add, edit, and delete repository parameters for a single shared object repository.

To manage repository parameters:

1 Make sure that the object repository whose parameters you want to manage is the active window.

2 If the object repository is in read-only format, select **File** > **Enable Editing** or click the **Enable Editing** button. The object repository becomes editable.

3 Select **Tools** > **Manage Repository Parameters** or click the **Manage Repository Parameters** button. The Manage Repository Parameters dialog box opens.



The Manage Repository Parameters dialog box contains the following information and options:

Option	Description
Repository name	Displays the name and path of the object repository whose repository parameters you are managing.
	Enables you to add a new repository parameter. For more information, see "Adding Repository Parameters" on page 230.
	Enables you to delete the currently selected repository parameters. For more information, see "Deleting Repository Parameters" on page 233.
Parameter list (**Name**, **Default Value**, and **Description**)	Displays the list of repository parameters currently defined in this object repository. You can modify a parameter's default value and description directly in the parameter list. For more information, see "Modifying Repository Parameters" on page 232.
Find in Repository button	Searches for and highlights the first test object in the object repository tree that uses the selected repository parameter. You can click this button again to find the next occurrence of the selected parameter, and so forth.

Adding Repository Parameters

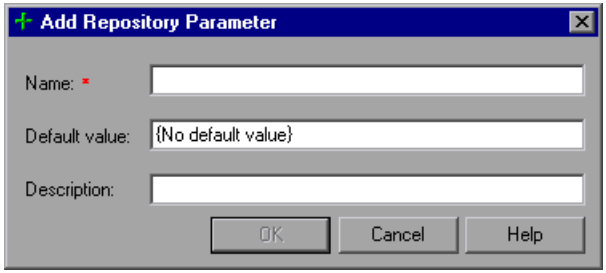
The functionality described in this section is available only when working in the Object Repository Manager.

The Add Repository Parameter dialog box enables you to define a new repository parameter. You can also specify a default value for the parameter, and a meaningful description to help identify it when it is used in a test step.

For more information on repository parameters, see "Working with Repository Parameters" on page 228.

To add a repository parameter:

1 In the Manage Repository Parameters dialog box, click the **Add Repository Parameter** button. The Add Repository Parameter dialog box opens.



2 In the **Name** box, specify a meaningful name for the parameter. Parameter names must start with an English (Roman) letter and can contain only English (Roman) letters, numbers, and underscores.

3 In the **Default value** box, you can specify a default value to be used for the repository parameter. This value is used if you do not map the repository parameter to a value or parameter type in a test that uses this object repository. If you do not specify a default value, the repository parameter will appear as unmapped in any tests that use this shared object repository.

Tip: If you specify a default value, you can later remove it by clicking in the **Default Value** cell of the relevant parameter in the Manage Repository Parameters dialog box and then clicking the **Clear Default Value** button. The text **{No Default Value}** is displayed in the cell.

---

4 In the **Description** box, you can enter a description of the repository parameter. The description will help you identify the parameter when mapping repository parameters within a test.

5 Click **OK** to add the parameter to the list of parameters in the Manage Repository Parameters dialog box.

Modifying Repository Parameters

The functionality described in this section is available only when working in the Object Repository Manager.

You can modify the default value of a repository parameter or modify a repository parameter description directly in the Manage Repository Parameters dialog box. However, you cannot modify a repository parameter name.

To modify a repository parameter:

1. In the Manage Repository Parameters dialog box, select the required parameter.

2. To modify the default value, click in the **Default Value** cell of the required parameter. You can either modify the default value by entering a new value, or you can remove the default value by clicking the **Clear Default Value** button. If you remove the default value, the text **{No Default Value}** is displayed in the cell. If you do not specify a default value, the repository parameter will appear as unmapped in any tests that use this shared object repository.

Note: If you delete the text manually, it does not remove the default value. It creates a default value of an empty string. You must click the **Clear Default Value** button if you want to remove the default value.

---

3. To modify the parameter description, click in the **Description** cell of the required parameter and enter the required description.

Deleting Repository Parameters

The functionality described in this section is available only when working in the Object Repository Manager.

You can delete a repository parameter definition if it is no longer needed. When you delete a repository parameter that is used in a test object definition, the identification property value remains mapped to the parameter, even though the parameter no longer exists. Therefore, before deleting a repository parameter, you should make sure that it is not used in any test object descriptions, otherwise tests that have steps using these test objects will fail when you run them.

Tip: You can use the **Find in Repository** button in the Manage Repository Parameters dialog box to see where a repository parameter is being used.

To delete a repository parameter:

1 In the Manage Repository Parameters dialog box, select the repository parameters that you want to delete by clicking in the selection area to the left of the parameter name.

2 Click the **Delete Repository Parameter** button. The selected repository parameter is deleted.

Modifying Object Details

The object details area for shared object repositories in the lower right side of the document window enables you to view and modify the properties and property values used to identify an object during a run session or the properties of a checkpoint or output object.

After making sure that your shared object repository is editable, and that it is the active window, you modify object details for objects in a shared object repository in the same way as you modify them for local objects. For more information, see:

- "Adding Properties to a Test Object Description" on page 171
- "Defining New Identification Properties" on page 174
- "Updating Identification Properties from an Object in Your Application" on page 165
- "Restoring Default Mandatory Properties for a Test Object" on page 168
- "Removing Properties from a Test Object Description" on page 177
- "Specifying Ordinal Identifiers" on page 177
- "Renaming Test Objects" on page 169

Note: You can use the **Edit** > **Undo** and **Edit** > **Redo** options or **Undo** and **Redo** buttons to cancel or repeat your changes as necessary. The **Undo** and **Redo** options are related to the active document. When you save a repository, you cannot undo and redo operations that were performed on that file before the save operation.

You use the Object Repository Manager to specify property values for object descriptions in a shared object repository. The options available when specifying property values for objects in shared object repositories are different from those available when specifying properties for objects in local repositories. For more information on specifying property values for objects in shared object repositories, see "Specifying a Property Value" on page 235.

Specifying a Property Value

The functionality described in this section is available only when working in the Object Repository Manager.

You can specify or modify values for properties in the test object description. You can specify a value using a constant value (either a simple value or a constant value that includes regular expressions) or you can parameterize it using a repository parameter. For more information on repository parameters, see "Working with Repository Parameters" on page 228.

You can also specify or modify values for properties of a checkpoint or output object.

Specifying and Modifying Values for Properties of a Test Object

You specify or modify the values for properties of a test object in the **Test object details** area.

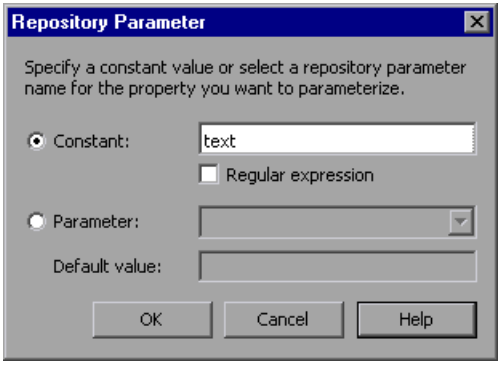
To specify a property value of a test object:

1. Select the test object whose property value you want to specify.
2. In the **Test object details** area, click in the **Value** cell for the required property.

3 Specify the property value in one of the following ways:

- If you want to specify a simple constant value, enter it in the **Value** cell. The remaining steps in this procedure are not necessary if you specify a constant value in the **Value** cell. You can also specify a constant value using a regular expression in the Repository Parameter dialog box, as described below.

- If you want to parameterize the value using a repository parameter, click the parameterization button in the **Value** cell. The Repository Parameter dialog box opens.



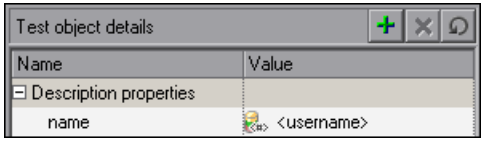
4 Select one of the following options to specify a value for the property:

- Select the **Constant** radio button and specify a constant value. You can also enter a constant value directly in the **Value** cell of the **Test object details** area. If you used a regular expression in the constant value, select the **Regular expression** check box.
- Select the **Parameter** radio button and select a repository parameter from the list of defined parameters. If a default value is defined for the parameter, it is also shown.

Note: You define repository parameters using the Manage Repository Parameters dialog box. For more information, see "Managing Repository Parameters" on page 229.

---

5 Click **OK** to close the Repository Parameter dialog box. If you parameterized the value, the parameter name is shown with an icon in the **Value** column of the **Test object details** area, as shown below. Otherwise, the constant value you specified is shown in the **Value** column.



Specifying and Modifying Values for Properties of a Checkpoint Object

You specify or modify the values for properties of a checkpoint object in the **Object Properties** pane.

To specify or modify the values for properties of a checkpoint object:

1 Select the checkpoint object whose property values you want to specify or modify from the **Checkpoint and Output Objects** tree.

2 Specify or modify the values for properties of a checkpoint object the same way as you do in the relevant checkpoint properties dialog box.

For more information on specifying and modifying values for properties of a checkpoint object, see:

- "Understanding the Checkpoint Properties Dialog Box" on page 508
- "Understanding the Image Checkpoint Properties Dialog Box" on page 512
- "The Bitmap Checkpoint Properties Dialog Box" on page 522
- "Understanding the Table Checkpoint Properties Dialog Box" on page 535
- "The Text / Text Area Checkpoint Properties Dialog Box" on page 557
- "Understanding the Database Checkpoint Properties Dialog Box" on page 581

- "Understanding the XML Checkpoint Properties Dialog Box" on page 607
- The Web section of the *HP QuickTest Professional Add-ins Guide* (for Page and Accessibility checkpoints)

Specifying and Modifying Values for Properties of an Output Object

You specify or modify the values for properties of an output object in the **Object Properties** pane.

To specify or modify the values for properties of an output object:

1. Select the output object whose property values you want to specify or modify from the **Checkpoint and Output Objects** tree.
2. Specify or modify the values for properties of an output object the same way as you do in the relevant output value properties dialog box.

   For more information on specifying and modifying values for properties of an output object, see:

   - "Defining Standard Output Values" on page 679
   - "Defining Text and Text Area Output Values" on page 692
   - "Outputting Table Content" on page 703
   - "Defining Database Output Values" on page 715
   - "Understanding the XML Output Properties Dialog Box" on page 727

Locating Test Objects

The functionality described in this section is available in the Object Repository window for objects in the local object repository, and the Object Repository Manager for objects in shared object repositories.

You can search for a specific test object in your object repository in several ways. You can search for a test object according to its type. For example, you can search for a specific edit box, or you can point to an object in your application to automatically highlight that same object in your repository. You can replace specific property values with other property values. For example, you can replace a property value userName with the value user name. You can also select an object in your object repository and highlight it in your application to check which object it is.

After making sure that your shared object repository is the active window, you locate an object in a shared object repository in the same way as you locate it in a local object repository. If you want to replace property values, you must also make sure that the object repository is editable.

For more information, see:

- "Finding Objects in an Object Repository" on page 154
- "Highlighting an Object in Your Application" on page 157
- "Locating a Test Object in the Object Repository" on page 159

Performing Merge Operations

The functionality described in this section is available only when working in the Object Repository Manager.

The Object Repository Merge Tool enables you to merge test objects from the local object repository of one or more actions to a shared object repository using the **Update from Local Repository** option in the Object Repository Manager (**Tools** > **Update from Local Repository**). For example, you may have learned test objects locally in a specific action in your test and want to add them to the shared object repository so they are available to all actions in different tests that use that object repository. You can also use the Object Repository Merge Tool to merge two shared object repositories into a single shared object repository.

You open the Object Repository Merge Tool by choosing **Tools** > **Object Repository Merge Tool** in the Object Repository Manager. For more information on performing merge operations and updating object repositories with local objects, see Chapter 8, "Merging Shared Object Repositories."

Notes:

- While the Object Repository Merge Tool is open, you cannot work with the Object Repository Manager.
- The Object Repository Merge Tool does not merge checkpoint and output objects.

Performing Import and Export Operations

You can import and export object repositories from and to XML files. XML provides a structured, accessible format that enables you to make changes to object repositories using the XML editor of your choice and then import them back into QuickTest. You can view the required format for the object repository in the *HP QuickTest Professional Object Repository Schema Help* (**Help** > **QuickTest Professional Help** > **HP QuickTest Professional Advanced References** > **HP QuickTest Professional Object Repository Schema**), or by exporting a saved object repository.

You can import and export files either from and to the file system or a Quality Center project (if QuickTest is connected to Quality Center).

You connect to a Quality Center project either from QuickTest or from the Object Repository Manager by choosing **File** > **Quality Center Connection** or clicking the **Quality Center Connection** button. For more information on connecting to Quality Center, see "Connecting to and Disconnecting from Quality Center" on page 1418.

For more information, see:

- "Importing from XML" on page 242
- "Exporting to XML" on page 243
- "Understanding the XML File Structure" on page 244

Importing from XML

The functionality described in this section is available only when working in the Object Repository Manager.

You can import an XML file (created using the required format) as an object repository. For information on the XML format, see "Understanding the XML File Structure" on page 244. The XML file can either be an object repository that you exported to XML format using the Object Repository Manager, or an XML file created using a tool such as QuickTest Siebel Test Express or a custom built utility. You must adhere to the XML structure and format.

Tip: To view the required XML structure and format, see the *HP QuickTest Professional Object Repository Schema Help* (**Help > QuickTest Professional Help > HP QuickTest Professional Advanced References > HP QuickTest Professional Object Repository Schema**). You can also export an existing shared object repository to XML and then use the XML file as a guide. For more information, see "Exporting to XML" on page 243.

To import from XML:

1 Select **File > Import from XML**. The Open XML File dialog box opens.

Note: Checkpoint and output objects are not included when importing the contents of an object repository from an XML file.

---

2 In the sidebar, select the location of the file, for example, File System or Quality Center Test Resources. Browse to and select the XML file you want to import, and click **Open**.

The XML file is imported and a summary message box opens showing information regarding the number of objects, parameters, and metadata that were successfully imported from the specified file.

3 Click **OK** to close the message box. The imported XML file is opened as a new object repository. You can now modify it as required and save it as an object repository.

Exporting to XML

The functionality described in this section is available only when working in the Object Repository Manager.

You can export the test objects in an object repository to an XML file. This enables you to edit it using any XML editor, and also enables you to save it in an accessible, versatile format.

To export to XML:

1 Make sure that the object repository whose test objects you want to export is the active window.

2 Make sure that the object repository is saved.

3 Select **File** > **Export Test Objects to XML**. The Save XML File dialog box opens.

Note: Checkpoint and output objects are not included when exporting the contents of an object repository to an XML file.

---

4 In the sidebar, select the location in which you want to save the file, for example, File System or Quality Center Test Resources.

5 Browse to and select the folder in which you want to save the file.

6 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**.

The test objects of the object repository are exported to the specified XML file, and a summary message box opens showing information regarding the number of objects, parameters, and metadata that were successfully exported to the specified file.

7 Click **OK** to close the message box. You can now open the XML file and view or modify it with any XML editor.

Understanding the XML File Structure

QuickTest uses a defined XML schema for object repositories. You must follow this schema when creating or modifying object repository files in XML format. The schema of this file is documented in the *HP QuickTest Professional Object Repository Schema Help* (**Help > QuickTest Professional Help > HP QuickTest Professional Advanced References > HP QuickTest Professional Object Repository Schema**).

Managing Object Repositories Using Automation

QuickTest provides an Object Repository automation object model that enables you to manage QuickTest shared object repositories and their contents from outside of QuickTest. The automation object model enables you to use a scripting tool to access QuickTest shared object repositories via automation.

Just as you use the QuickTest Professional automation object model to automate your QuickTest operations, you can use the objects and methods of the Object Repository automation object model to write scripts that manage shared object repositories, instead of performing these operations manually using the Object Repository Manager. For example, you can add, remove, and rename test objects; import from and export to XML; retrieve and copy test objects; and so forth.

After you have retrieved a test object, you can manipulate it using the methods and properties available for that test object class. For example, you can use the GetTOProperty and SetTOProperty methods to retrieve and modify its properties. For more information on available test object methods and properties, see the *HP QuickTest Professional Object Model Reference*.

Automation programs are especially useful for performing the same tasks multiple times or on multiple object repositories. You can write your 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. For general information on controlling QuickTest using automation, see "Automating QuickTest Operations" on page 1403.

Using the QuickTest Professional Object Repository Automation Reference

The QuickTest Professional Object Repository Automation Reference is a Help file that provides detailed descriptions, syntax information, and examples for the objects and methods in the QuickTest object repository automation object model.

The Help topic for each automation object includes a list and description of the methods associated with that object. Method Help topics include detailed description, syntax, return value type, and argument value information.

You can open the *QuickTest Professional Object Repository Automation Reference* from the main QuickTest Help (**Help > QuickTest Professional Help > HP QuickTest Professional Advanced References > HP QuickTest Professional Object Repository Automation**).

Note: The syntax and examples in the Help file are written in VBScript-style. If you are writing your automation program in another language, the syntax for some methods may differ slightly from what you find in the corresponding Help topic. For information on syntax for the language you are using, see the documentation included with your development environment or to general documentation for the programming language.

8

Merging Shared Object Repositories

QuickTest Professional enables you to merge two shared object repositories into a single shared object repository using the Object Repository Merge Tool. You can also use this tool to merge objects from the local object repository of one or more actions into a shared object repository.

This chapter includes:

About Merging Shared Object Repositories

When you have multiple shared object repositories that contain test objects from the same area of your application, it may be useful to combine those test objects into a single object repository for easier maintenance. You could do this by moving or copying objects in the Object Repository Manager. However, if you have test objects in different object repositories that represent the same object in your application, and the descriptions for these objects in the different object repositories are not identical, it may be difficult to recognize and handle these conflicts.

The Object Repository Merge Tool helps you to solve the above problem by merging two selected object repositories for you and providing options for addressing test objects with conflicting descriptions. Using this tool, you merge two shared object repositories (called the **primary** object repository and the **secondary** object repository), into a new third object repository, called the **target** object repository. Objects in the primary and secondary object repositories are automatically compared and then added to the target object repository according to preconfigurable rules that define the defaults for how conflicts between objects are resolved.

After the merge process, the Object Repository Merge Tool provides a graphic presentation of the original objects in the primary and secondary object repositories, which remain unchanged, as well as the objects in the merged target object repository. Objects that had conflicts are highlighted. The conflict of each object that you select in the target object repository is described in detail. The Object Repository Merge Tool provides specific options that enable you to keep the default resolution for each conflict, or modify conflict resolutions individually, according to your requirements.

The Object Repository Merge Tool also enables you to merge objects from the local object repository of one or more actions into a shared object repository. For example, if QuickTest learned objects locally in a specific action in your test, you may want to add the objects to the shared object repository, so that they are available to all actions in different tests that use that object repository.

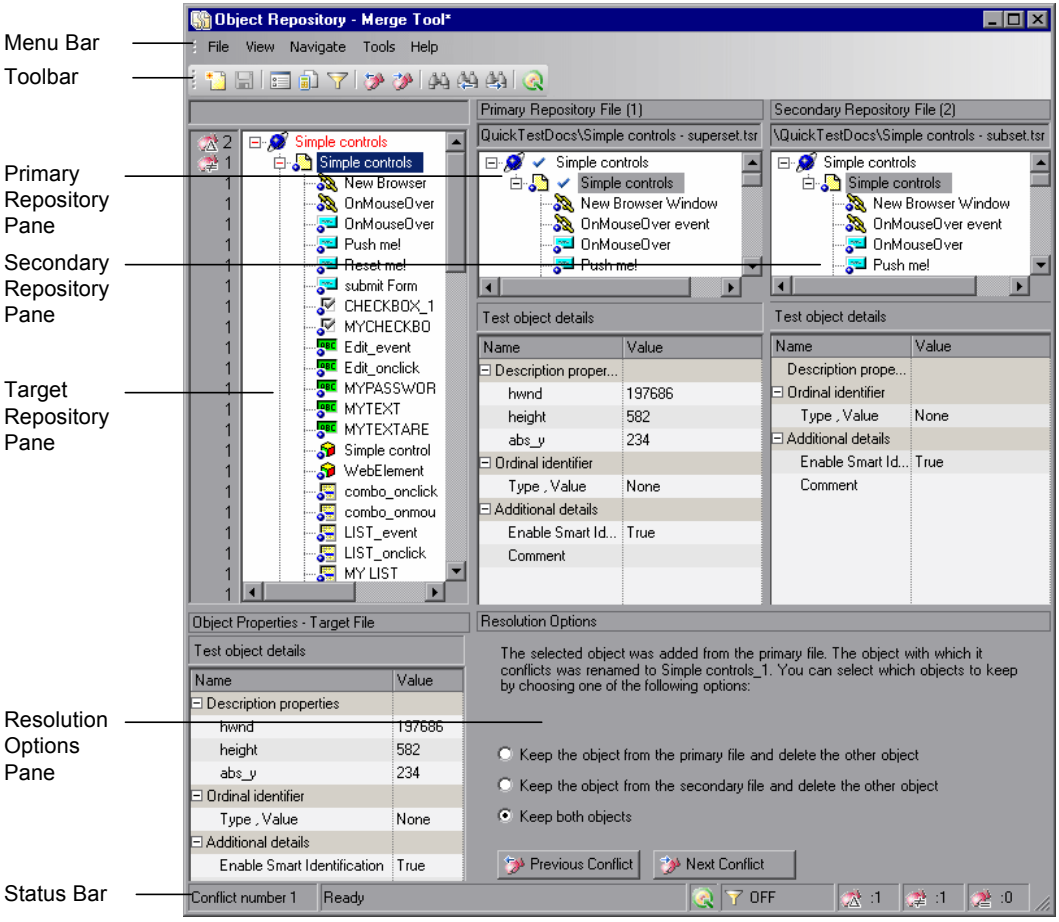
Notes:

- The Object Repository Merge Tool does not merge checkpoint or output objects from the primary and secondary object repositories into the target shared object repository. You can copy or manually move these objects to your target object repository after you complete the merge process, using the Object Repository Manager.
- When the Object Repository Merge Tool is open, you cannot work with the Object Repository Manager or Object Repository Comparison Tool. For more information on the Object Repository Manager, see Chapter 7, "Managing Object Repositories."

Understanding the Object Repository Merge Tool

You open the Object Repository Merge Tool by choosing **Tools** > **Object Repository Merge Tool** in the Object Repository Manager.

An example of the Object Repository - Merge Tool window is shown below:



Note: For information about changing the view presented by the Object Repository Merge Tool, see “Changing the View” on page 252.

The Object Repository - Merge Tool window contains the following key elements:

- **Menu bar.** Displays menus of Object Repository Merge Tool commands. These commands are described in various places throughout this chapter. Shortcut keys for menu commands are described in "File Menu Commands" on page 258.
- **Toolbar.** Contains buttons of commonly used menu commands to assist you in merging, managing, and saving object repositories. Toolbar buttons are described in "Using Toolbar Commands" on page 257.
- **Target Repository Pane.** Displays the objects that were merged from the primary and secondary object repositories. You can also choose to show or hide the Target Repository Object Properties pane, which displays the properties of any object that is selected in the Target Repository pane. For more information, see "Target Repository Pane" on page 252.
- **Primary Repository Pane.** Displays the objects in the primary object repository. For more information, see "Primary and Secondary Repository Panes" on page 254.
- **Secondary Repository Pane.** Displays the objects in the secondary object repository. For more information, see "Primary and Secondary Repository Panes" on page 254.
- **Resolution Options Pane.** Provides source, conflict, and resolution details about the objects in the target object repository pane, and enables you to modify how a selected conflict is resolved. For more information, see "Resolution Options Pane" on page 254.
- **Status Bar.** Provides source, conflict, and resolution details about the object selected in the target object repository pane, the filter status, and an icon legend. For more information, see "Status Bar" on page 255.

Changing the View

You can change the view presented by the Object Repository Merge Tool according to your working preferences.

- Drag the edges of the panes to resize them in the Object Repository Merge Tool window.
- Select **Primary Repository**, **Secondary Repository**, **Target Repository Object Properties**, or **Resolution Options** from the **View** menu to hide or show these panes in the Object Repository Merge Tool.
- Select **View** > **Set as Default Layout** to set your current view as the default view, which displays each time you open the Object Repository Merge Tool. You can select **View** > **Restore Default Layout** to restore the view to the default settings after you make changes.

Target Repository Pane

The target object repository pane displays a hierarchy of the objects, as well as their respective properties and values, that were merged from the primary and secondary object repositories. In the column to the left of the object hierarchy, the pane displays the source file of each object (**1** is displayed for the primary file and **2** for the secondary file), and an icon representing the type of conflict, if any.

When you save the target object repository, the file path is displayed above the object hierarchy.

Note: To make it easier to see the status of an object at a glance, the text colors of the object names in the target object repository can be set according to their source and whether they caused a conflict. For more information, see “Specifying Color Settings” on page 265.

The target object repository pane provides the following functionality:

- When you select an object in the target object repository, the corresponding object in the primary and/or secondary source file hierarchy is located and indicated by a check mark.
- When you select an object in the target object repository, its properties and values are displayed in the **Object Properties - Target File** area at the bottom of the target object repository pane (**View** > **Target Repository Object Properties**).
- If the merge results in a conflict, an icon is displayed to the left of the conflicting object in the target object repository. You can see a tooltip description of the conflict type by positioning your pointer over the icon.
- When you right-click an object, a context-sensitive menu opens. You can expand an option or collapse the entire hierarchy in the target object repository, or, when applicable, you can change the conflict resolution method and result.
- You can expand or collapse the hierarchy of the node by double-clicking a node. You can also expand or collapse the entire hierarchy in the target object repository by choosing **Collapse All** or **Expand All** from the **View** menu.
-  You can jump directly to the next or previous conflict in the target object repository hierarchy by choosing **Next Conflict** or **Previous Conflict** from the **Navigate** menu, or by clicking the **Next Conflict** or **Previous Conflict** buttons in the toolbar or Resolution Options pane.
- You can locate one or more objects in the target object repository by using the Find dialog box. For more information, see “Finding Specific Objects” on page 284.
- You can show or hide the target object repository object properties by choosing **View** > **Target Repository Object Properties**.

Primary and Secondary Repository Panes

The primary and secondary object repository panes display the hierarchies of the objects, and their properties and values, in the original source object repositories that you chose to merge. The file path is shown above each object hierarchy.

The panes provide the following functionality:

- You can expand or collapse the hierarchy of a selected item by double-clicking the item.
- You can view the properties and values of an object in the **Test object details** area by selecting it in the relevant pane.
- You can show or hide the panes by selecting or clearing **Primary Repository** or **Secondary Repository** in the **View** menu.

Resolution Options Pane

The Resolution Options pane provides information about any conflict encountered during the merge for the object selected in the target object repository. The pane also provides options that enable you to keep or change the conflict resolution method that was applied using the default resolution options.

The Resolution Options pane provides the following functionality:

- When you select a conflicting object in the target object repository, the pane displays a textual description of the conflict and the resolution method used by the Object Repository Merge Tool. A choice of alternative resolution methods is offered.
- You can select a radio button to choose an alternative resolution method for the conflict. Every time you make a change, the target object repository is automatically updated and is redisplayed.
- You can jump directly to the next or previous conflict in the target object repository hierarchy by clicking the **Previous Conflict** or **Next Conflict** buttons.

- For a local object repository merge, you can click the **Ignore Object** button to exclude a specific local object repository object from the merge process. The object remains in the local object repository when the merge is complete.
- You can show or hide the pane by selecting or clearing **Resolution Options** in the **View** menu.

Status Bar

The status bar shows the following information about the merge process and the results that are displayed:

- The conflict number (if any) of the object selected in the target object repository pane.
- A progress bar is displayed during the merge process. **Ready** is displayed when the is complete.

- The Quality Center icon is displayed when QuickTest is connected to a Quality Center project.

- The filter status is shown next to the Filter icon: **OFF** indicates that the object repositories are not filtered and all objects are shown. **ON** indicates a filter is active and that some objects may have been filtered out of the display.
- A legend of the icons used in the target object repository pane. The following icons may be displayed:

  - Similar Description Conflict

  - Same Name Different Description Conflict

  - Same Description Different Name Conflict

For more information on conflict types, see “Understanding Object Conflicts” on page 277.

Tips:

- Position your pointer over a conflict icon in the status bar to see a tooltip description of the conflict type.
- Click any of the conflict icons to view the Statistics dialog box. For more information, see "Viewing Merge Statistics" on page 276.
- Click the **Filter** icon in the status bar to view the Filter dialog box. The filter is shown as **ON** in the status bar when a filter is currently in use. For more information, see "Filtering the Target Repository Pane" on page 282.

Using Object Repository Merge Tool Commands

You can select Object Repository Merge Tool commands from the menu bar or from the toolbar. You can perform certain commands by pressing shortcut keys. You can also select an object in the target object repository pane and choose commands from the context (right-click) menu.

Using Toolbar Commands

You can perform frequently used commands by clicking buttons in the Object Repository Merge Tool toolbar.

	Description
	New Merge (described in “File Menu Commands” on page 258)
	Save (described in “File Menu Commands” on page 258)
	Settings (described in “Tools Menu Commands” on page 261)
	Statistics (described in “View Menu Commands” on page 259)
	Filter (described in “Tools Menu Commands” on page 261)
	Previous Conflict (described in “Navigate Menu Commands” on page 261)
	Next Conflict (described in “Navigate Menu Commands” on page 261)
	Find (described in “Navigate Menu Commands” on page 261)
	Find Previous (described in “Navigate Menu Commands” on page 261)
	Find Next (described in “Navigate Menu Commands” on page 261)
	Quality Center Connection (described in “File Menu Commands” on page 258)

Performing Object Repository Merge Tool Commands

You can perform frequently-used commands by clicking toolbar buttons or choosing the relevant menu option. You can also perform some commands by pressing the relevant shortcut keys.

File Menu Commands

You can manage your merged object repository using the following **File** menu commands:

	Command	Shortcut Key	Function
	**New Merge**	CTRL+N	Enables you to specify two object repositories with which to perform a new merge operation.
	**Save**	CTRL+S	Saves the merged shared object repository.
	Save As		Opens the Save Shared Object Repository dialog box, enabling you to specify a name, file type, and storage location for the merged shared object repository.
	**Quality Center Connection**		Enables you to connect QuickTest to a Quality Center project. For more information, see "Connecting to and Disconnecting from Quality Center" on page 1418.
	Exit		Closes the Object Repository - Merge Tool window. (Also prompts you to save the merged object repository if you did not yet save it.)

View Menu Commands

You can manage the way that the Object Repository Merge Tool is displayed on your screen using the following **View** menu commands:

	Command	Function
	Primary Repository	Displays the Primary Repository File pane, containing a hierarchical view of the objects from the first source object repository that you chose to merge. Also displays the details for each object selected in this pane. For more information, see "Primary and Secondary Repository Panes" on page 254 and "Merging Two Object Repositories" on page 267.
	Secondary Repository	Displays the Secondary Repository File pane, containing a hierarchical view of the objects from the second source object repository that you chose to merge. Also displays the details for each object selected in this pane. For more information, see "Primary and Secondary Repository Panes" on page 254 and "Merging Two Object Repositories" on page 267.
	Target Repository Object Properties	Displays the Object Properties - Target File pane, which displays the details for each test object selected in the target repository pane. For more information, see "Target Repository Pane" on page 252.
	Resolution Options	Displays the Resolution Options pane, which provides information about any conflict that occurred during the merge. For more information, see "Resolution Options Pane" on page 254 and "Resolving Object Conflicts" on page 280.
	Restore Default Layout	Restores the view that you saved using the **Set as Default Layout** option (described below). This is useful if you resize a pane, or show or hide specific panes and then want to restore your saved view. For more information, see "Changing the View" on page 252.

	Command	Function
	Set as Default Layout	Enables you to save the current view so that each time you open the Object Repository - Merge Too, this view is displayed. If you later modify this view by resizing panes, or showing or hiding them, you can restore your default view using the **Restore Default Layout** option (described above). For more information, see "Changing the View" on page 252.
	**Statistics**	Opens the Statistics dialog box, which describes how the files were merged, and the number and type of any conflicts that were resolved during the merge. For more information, see "Viewing Merge Statistics" on page 276.
	Collapse All	Collapses the entire hierarchy in the Target Object Repository pane. **Tip:** You can collapse a single node by double-clicking it.
	Expand All	Expands the entire hierarchy in the Target Object Repository pane. **Tip:** You can expand a single node by double-clicking it.

Navigate Menu Commands

You can perform the following **Navigate** menu commands:

	Command	Shortcut Key	Function
	**Next Conflict**	F4	Finds the next conflicting object in the merged object repository.
	**Previous Conflict**	SHIFT+F4	Finds the previous conflicting object in the merged object repository.
	**Find**	CTRL+F	Opens the Find dialog box.
	**Find Next**	F3	Finds the next object in the merged object repository according to the search specifications in the Find dialog box.
	**Find Previous**	SHIFT+F3	Finds the previous object in the merged object repository according to the search specifications in the Find dialog box.

Tools Menu Commands

You can perform the following **Tools** menu commands:

	Command	Function
	**Settings**	Opens the Settings dialog box, enabling you to: ➤ Configure how the Object Repository Merge Tool deals with conflicting objects during a merge ➤ Specify the text color of the object names displayed in the target object repository For more information, see "Defining Default Settings" on page 262.
	**Filter**	Opens the Filter dialog box, enabling you to show all of the test objects in the Target Repository pane, or show only the objects that had conflicts that were resolved during the merge. For more information, see "Filtering the Target Repository Pane" on page 282.

Help Menu Command

You can perform the following **Help** menu command:

Command	Shortcut Key	Function
Object Repository Merge Tool Help	F1	Opens the Object Repository Merge Tool Help.

Defining Default Settings

The Object Repository Merge Tool is supplied with predefined settings that are used when merging object repositories or when updating a shared object repository from local object repositories. These settings:

- Configure how the Object Repository Merge Tool deals with conflicting objects in the primary and secondary object repositories (or local and shared object repositories when updating a shared object repository from local object repositories).
- Specify the text color of the object names that are displayed in the target object repository.

You can change these settings at any time to create new default settings. After you change the settings, all new merges are performed according to the new default settings.

Tip: If you want to change the settings before merging two object repositories, you must click **Cancel** to close the New Merge dialog box, change the settings as described in the next sections, and then perform the merge.

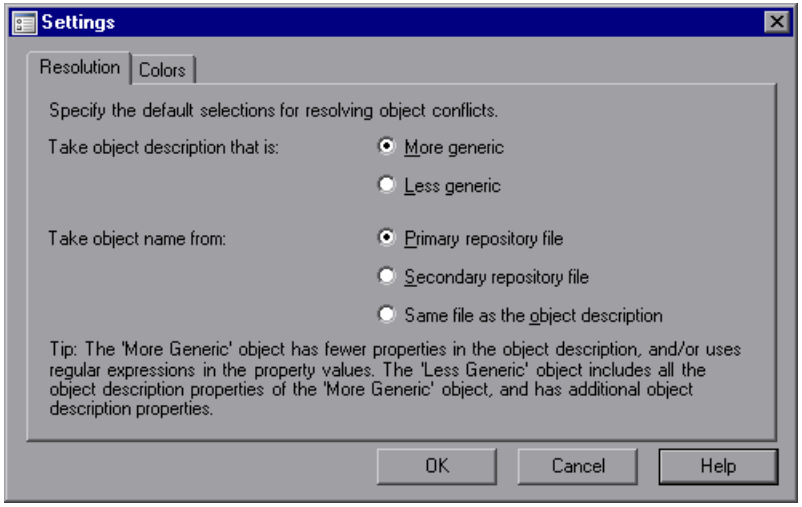
Specifying Default Resolution Settings

You can configure how the Object Repository Merge Tool automatically deals with conflicting objects during the merge process or when performing an **Update from Local Repository** operation.

To specify default resolution settings:

1 Select **Tools** > **Settings** or click the **Settings** button. The Settings dialog box opens.

2 Click the **Resolution** tab.



3 Select the appropriate radio buttons to specify the default resolution settings that the Object Repository Merge Tool applies when dealing with conflicting objects.

- **Take object description that is.** Specifies how to resolve conflicts in which two objects have the same name, but their descriptions differ. You can specify that the target object repository takes the object description that is more generic or less generic.
  - **More generic.** Instructs the Object Repository Merge Tool to take the object that has fewer identifying properties than the object with which it conflicts, or uses regular expressions in its property values. This is the default setting.
  - **Less generic.** Instructs the Object Repository Merge Tool to take the object that has all the identifying properties of the object with which it conflicts, plus additional identifying properties.
- **Take object name from.** Specifies how to resolve conflicts where two objects have the same or similar descriptions, but their names differ. You can select the source from which the target object repository takes the object name:
  - **Primary repository file.** The target object repository takes the object name from the object in the primary object repository. This is the default setting. (When updating a shared object repository from a local object repository, this option is for the **Local object repository**.)
  - **Secondary repository file.** The target object repository takes the object name from the object in the secondary object repository. (When updating a shared object repository from a local object repository, this option is for the **Shared object repository**.)
  - **Same file as the object description.** The target object repository takes the object name from the object in the same object repository from which it took the object description.

Note: When updating a shared object repository from a local object repository, the object repositories are referred to as the Local and Shared object repository.

---

4 Click **OK**. The Object Repository Merge Tool will apply your selections when resolving conflicts between objects in all future object repository merges.

Note: If you make any change to the resolution settings while a merged object repository is open, you are asked whether you want to merge the open files again with the new settings. Click **Yes** to merge the files again with the new settings, or click **No** to keep the existing merge created with the previous settings. If you click **No**, the new settings will apply only to future merges.

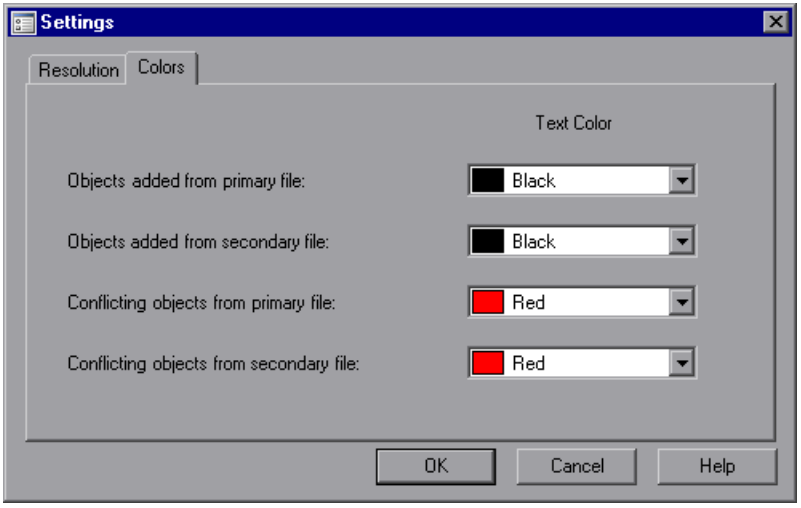
Specifying Color Settings

You can specify the color in which object names are displayed in the target object repository according to their source, and whether they caused a conflict. This enables you to see the status of each object more easily.

Note: The options in the Colors tab of the Settings dialog box apply equally to objects added from the local (primary) and shared (secondary) object repositories, when performing an **Update from Local Repository** operation.

To specify color settings:

1 Select **Tools** > **Settings** or click the **Settings** button. The Settings dialog box opens.



2 For each item in the Colors tab, click the down arrow next to the text box and select an identifying color from the Custom, Web, or System tabs.

3 Click **OK**. Object names in the target object repository are displayed in the selected color according to your selections.

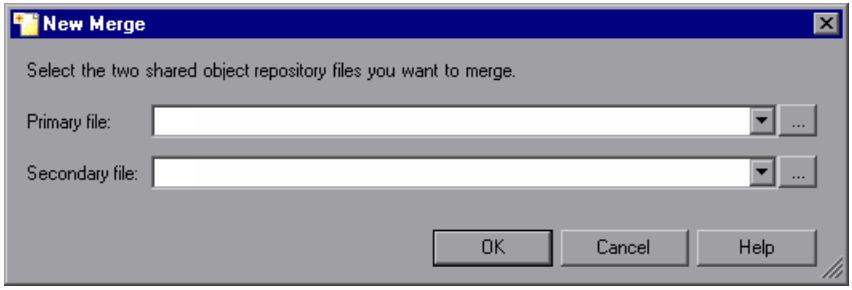
Merging Two Object Repositories

Using the Object Repository Merge Tool, you can merge two source object repositories to create a new shared object repository. Objects in the object repositories are automatically compared and added to the new object repository according to configurable rules that define how conflicts between objects are resolved. The original source files are not changed.

Note: An object repository that is currently open by another user is locked. If you try to merge the locked file, a warning message displays, but you can still perform the merge because the merge process does not modify the source files. Note that changes made to the locked file by the other user may not be included in the merged object repository.

To merge two object repositories:

1 In the Object Repository Manager, select **Tools** > **Object Repository Merge Tool**. The New Merge dialog box opens on top of the Object Repository - Merge Tool window.



Tips:

- If the Object Repository - Merge Tool window is already open, you can select **File** > **New Merge** or click the **New Merge** button to open the New Merge dialog box.
- If you want to change the configured settings before merging the object repositories, click **Cancel** to close the New Merge dialog box, change the settings as described in "Defining Default Settings" on page 262, and then perform the merge.

2 In the **Primary file** and **Secondary file** boxes, enter (or browse to) the **.tsr** object repositories that you want to merge into a single object repository. You can click the down arrow  next to each box to view and select recently used files.

Notes:

- It is recommended that you select as your primary object repository the object repository in which you have invested the most effort, meaning the object repository with more objects, object properties, and values.

- A warning icon is displayed next to the relevant text box if you enter the name of a file without a **.tsr** suffix, a file with an incorrect path, or a file that does not exist. You can position your pointer over the icon to see a tooltip explanation of the error. Enter or select an existing **.tsr** file with the correct path.
- If you want to merge an object repository that was created using a version of QuickTest earlier than version 9.0, you must first open and save it in the Object Repository Manager to update it to the new format.
- If you are connected to Quality Center, you can enter (or browse to) object repositories from Quality Center as well as from the file system.

3 Click **OK**. The Object Repository Merge Tool automatically merges the selected object repositories into a new target object repository according to the configured resolution settings, and displays the results in the Statistics dialog box on top of the Object Repository - Merge Tool window.

4 Review the merge statistics, as described in "Viewing Merge Statistics" on page 276, and click **Close**.

In the Object Repository - Merge Tool window, you can:

- Modify any conflict resolutions between objects from the source object repositories, if necessary, as described in "Resolving Object Conflicts" on page 280.
- Filter the objects in the target object repository, as described in "Filtering the Target Repository Pane" on page 282.
- Save the target object repository to the file system or to a Quality Center project, as described in "Saving the Target Object Repository" on page 285.

Updating a Shared Object Repository from Local Object Repositories

You can update a shared object repository by merging local object repositories associated with actions in one or more tests into the shared object repository. The objects that are merged from the local object repositories are then available to any actions that use that shared object repository in any tests.

In the merge process, the objects in the local object repository for the selected action are moved to the target shared object repository. The action then uses the objects from the updated shared object repository.

You can view or change how conflicting objects are dealt with during the update process in the Settings dialog box. For more information, see "Defining Default Settings" on page 262.

If you choose to add local object repositories for more than one action, QuickTest performs multiple merges, merging each action's local object repository with the target object repository one at a time, for all the actions in the list. You can view and modify the results of each merge if necessary.

Notes:

- The Object Repository Merge Tool does not merge checkpoint or output objects from a local object repository into the target shared object repository. You can export checkpoint or output objects from a local object repository to a shared object repository and then manually move the checkpoint and output objects from the exported object repository to your target object repository after you complete the merge process, using the Object Repository Manager.
- You can merge local object repositories only from actions that are associated with the shared object repository you are updating.

To update a shared object repository from a local object repository:

1 Select **Resources** > **Object Repository Manager**. The Object Repository Manager opens.

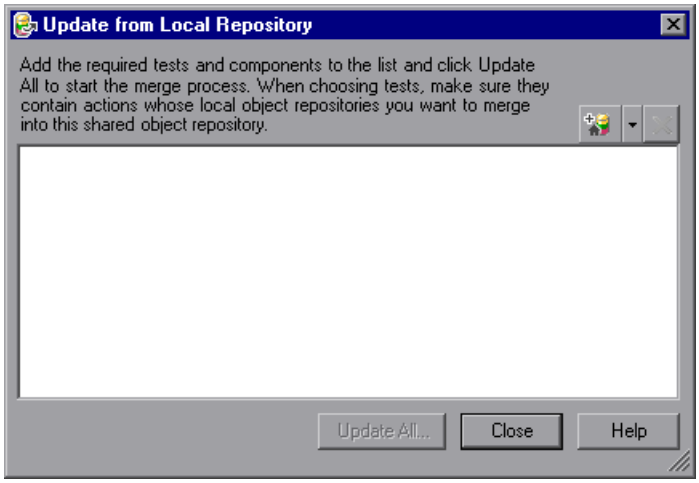
Note: For more information on the Object Repository Manager, see Chapter 7, "Managing Object Repositories."

2 In the Object Repository Manager, select **File** > **Open** or click the **Open** button. The Open Shared Object Repository dialog box opens.

3 In the sidebar, select the location of the object repository file, for example, File System or Quality Center Resources. Browse to and select the **.tsr** file that contains the shared object repository you want to update, clear the **Open in read-only mode** check box, and click **Open**. The file opens with the objects and properties displayed in editable format.

Tip: If you opened the object repository in read-only mode, select **File** > **Enable Editing** or click the **Enable Editing** button in the Object Repository Manager toolbar. The object repository file is made editable.

4 Select **Tools** > **Update from Local Repository**. The Update from Local Repository dialog box opens.



5 Click the down arrow next to the **Add Tests** button, and select **Browse for Test**. The Open Test dialog box opens.

In the sidebar, select the location of the test containing actions whose local object repositories you want to merge into the shared object repository, for example, File System or Quality Center Test Plan, and then select the test.

You can only add a test containing actions that are associated with the shared object repository you are updating and whose local object repositories contain objects.

6 Repeat step 5 to add additional tests if required.

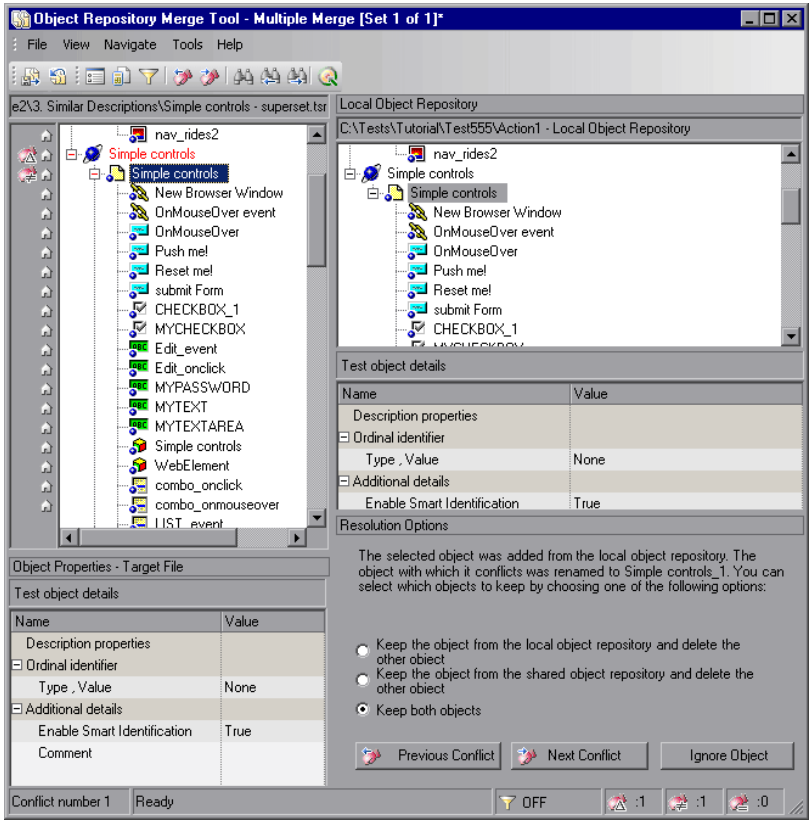
Note: The local object repositories associated with all the actions contained in the listed tests are included in the merge. If you want to remove an action from the merge, select it in the list and click **Delete**.

7 Click **Update All**. QuickTest automatically merges the first action local object repository into the shared object repository according to the configured settings, and displays the results in the Statistics dialog box on top of the Object Repository Merge Tool window.

Note: Before each merge, QuickTest checks whether the local object repository is in use by another user. If so, the local object repository is locked and the objects for the selected action cannot be moved to the target shared object repository. A warning message is displayed. The merge can be performed when the local object repository is no longer in use by the other user.

8 Review the merge statistics, as described in "Viewing Merge Statistics" on page 276, and click **Close**.

The Object Repository - Merge Tool window for a local object repository merge displays the local object repository as the primary object repository, and the shared object repository as the target object repository.



At the left of each object in the target object hierarchy is an icon that indicates the source of the objects:

 indicates that the object was added from the local object repository.

 indicates that the object already existed in the shared object repository.

Note: If you specified more than one action in the Update from Local Repository dialog box, QuickTest performs multiple merges, merging each action's local object repository with the target object repository one at a time. The Statistics dialog box and the Object Repository Merge Tool - Multiple Merge window displayed after this step show the merge results of the first merge (the local object repository of the first action being merged into the shared object repository). QuickTest enables you to view, and modify if necessary, the results of each merge in sequence. The number of each merge set in a multiple merge is displayed in the title bar, for example, [Set 2 of 3].

---

9 For each object merged into the shared object repository, you can accept the automatic merge or use the Resolution Options pane to:

- ➤ Keep a specific object from the shared object repository and delete the conflicting object from the local object repository.
- ➤ Keep a specific object from the local object repository and delete the conflicting object from the shared object repository.
- ➤ Keep conflicting objects from both the shared object repository and the local object repository.
- ➤ Exclude a specific local repository object from the merge process so that it is not included in the shared object repository. Select the object in the Shared Object Repository pane and click **Ignore Object** at the bottom of the Resolution Options pane. The object is removed from the shared object repository and grayed in the local object repository tree. It remains in the action's local object repository when the merge is complete.

Notes:

- ➤ The **Ignore Object** button is only visible in the Merge Tool window for a local object repository merge, and is only enabled when an object in the local object repository is selected.

- ➤ The **Ignore Object** operation cannot be reversed. To include the object again in the merge process, you must repeat the merge by clicking **Revert to Original Merged Files** in the toolbar.

For more information, see "Resolving Object Conflicts" on page 280.

10 If you are performing multiple merges, click the **Save and Merge Next** button in the Object Repository Merge Tool toolbar to perform the next merge (the local object repository of the next action being merged into the shared object repository).

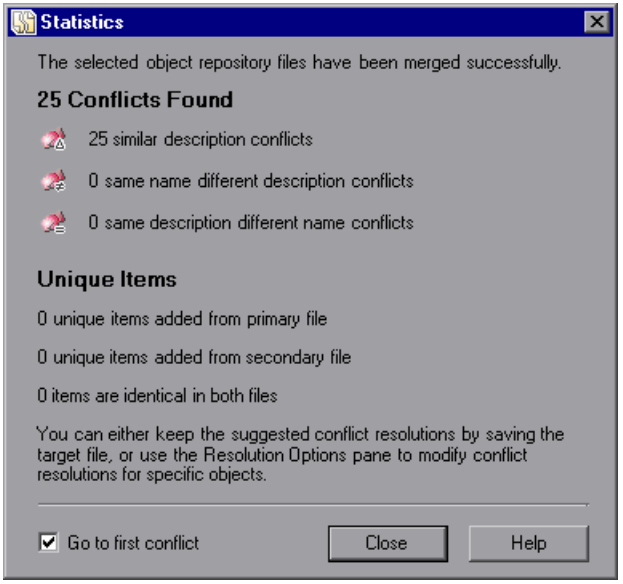
11 Click **Yes** to save your changes between merges. If you click **No**, the current merge (objects merged from the last action) will not be saved.

12 Repeat steps 8 to 11 to complete the multiple merges.

13 Select **File** > **Exit**, then click **Yes** to save the updated object repository.

Viewing Merge Statistics

After you merge two object repositories, the Object Repository Merge Tool displays the Statistics dialog box, which describes how the files were merged, and the number and type of any conflicts that were resolved during the merge.



Note: The Statistics dialog box shown after performing an **Update from Local Repository** merge differs slightly from the dialog box shown above.

Tip: You can view the merge statistics in the Statistics dialog box at any time by choosing **View > Statistics** in the Object Repository - Merge Tool window, by clicking the **Statistics** button in the toolbar, or by clicking a conflict icon in the status bar.

The Statistics dialog box displays the following information:

- The number and type of any conflicts between the objects added to the target object repository. Conflict types are described in "Resolving Object Conflicts" on page 280.
- The number of items added to the target object repository that are unique in each of the primary or secondary (or local) files, or are identical in both files.

Tip: Select the **Go to first conflict** check box to jump to the first conflict in the target object repository immediately after you close the Statistics dialog box.

Understanding Object Conflicts

Merging two object repositories can result in conflicts arising from similarities between the objects they contain. The Object Repository Merge Tool identifies three possible conflict types:

- **Similar Description Conflict.** Two objects that have the same name and the same object hierarchy, but that have slightly different descriptions. In this conflict type, one of the objects always has a subset of the properties set of the other object. These conflicts are described on page 278.

  By default, the conflict resolution settings for conflicts of this type are configured so that the target object repository takes the object that has fewer identifying properties than the object with which it conflicts. For information on changing the default settings, see "Defining Default Settings" on page 262.

- **Same Name Different Description Conflict.** Two objects that have the same name and the same object hierarchy, but differ somehow in their description (for example, they have different properties, or the same property with different values). These conflicts are described on page 279.

By default, the conflict resolution settings for conflicts of this type are configured so that the target object repository takes the object from both files. The object that is added from the secondary file is renamed by adding an incremental numeric suffix to the name, for example, Edit_1. For information on changing the default settings, see "Defining Default Settings" on page 262.

➤ **Same Description Different Name Conflict.** Two objects that have identical descriptions and have the same object hierarchy, but differ in their object names. These conflicts are described on page 280.

By default, the conflict resolution settings for conflicts of this type are configured so that the target object repository takes the object name from the primary source file. For information on changing the default settings, see "Defining Default Settings" on page 262.

Note: Objects that do not have a description, such as Page or Browser objects, are compared by name only. If the same object is contained in both the source object repositories but with different names, they will be merged into the target object repository as two separate objects.

Similar Description Conflict

An object in the primary object repository and an object in the secondary object repository have the same name, and they have similar, but not identical, description properties and values. One of the objects always has a subset of the properties set of the other object. For example, an object named Button_1 in the secondary object repository has the same description properties and values as an object named Button_1 in the primary object repository, but also has additional properties and values.

You can resolve this conflict type by:

- Keeping the object added from the primary object repository only.
- Keeping the object added from the secondary object repository only.
- Keeping the objects from both object repositories. In this case, the Object Repository Merge Tool automatically renames the object that is added from the secondary file by adding an incremental numeric suffix to the name, for example, Edit_1.
- Ignoring the object from the local object repository and keeping the object from the shared object repository (when updating a shared object repository from a local object repository).

Same Name Different Description Conflict

An object in the primary object repository and an object in the secondary object repository have the same name, but completely different description properties and values.

You can resolve this conflict type by:

- Keeping the object added from the primary object repository only.
- Keeping the object added from the secondary object repository only.
- Keeping the objects from both object repositories. In this case, the Object Repository Merge Tool automatically renames the object that is added from the secondary file by adding an incremental numeric suffix to the name, for example, Edit_1.
- Ignoring the object from the local object repository and keeping the object from the shared object repository (when updating a shared object repository from a local object repository).

Same Description Different Name Conflict

An object in the primary object repository and an object in the secondary object repository have different names, but the same description properties and values.

You can resolve this conflict type by:

- Taking the object name from the object in the primary object repository.
- Taking the object name from the object in the secondary object repository.
- Ignoring the object from the local object repository and keeping the object from the shared object repository (when updating a shared object repository from a local object repository).

Resolving Object Conflicts

Conflicts between objects in the primary and secondary object repositories are resolved automatically by the Object Repository Merge Tool according to the default resolution settings that you can configure before performing the merge. For more information, see "Defining Default Settings" on page 262.

However, the Object Repository Merge Tool also allows you to change the way the merge was performed for each individual object that causes a conflict.

For example, an object in the primary object repository could have the same name as an object in the secondary object repository, but have a different description. You may have defined in the default settings that in this case, the object with the more generic object description, meaning the object with fewer properties, should be added to the target object repository. However, when you review the conflicts after the automatic merge, you could decide to handle the specific conflict differently, for example, by keeping both objects.

Note: Changes that you make to the default conflict resolution can themselves affect the target object repository by causing new conflicts. In the above example, keeping both objects would cause a name conflict. Therefore, the target object repository is updated after each conflict resolution change and redisplayed.

You can identify objects that caused conflicts, and the conflict type, by the icon displayed to the left of the object name in the target object repository pane of the Object Repository Merge Tool and the text color. When you select a conflicting object, a full description of the conflict, including how it was automatically resolved by the Object Repository Merge Tool, is displayed in the Resolutions Options pane.

The Resolutions Options pane offers alternative resolution options. You can choose to keep the default resolution if it suits your needs, or use the alternative options to resolve the conflict in a different way. In addition, for a local object repository merge, you can click the **Ignore Object** button to exclude a specific local object repository object from the target shared object repository.

Tip: You can also change the default resolution settings and merge the files again. For more information, see "Defining Default Settings" on page 262.

To change the way in which object conflicts are resolved:

1 In the target object repository, select an object that had a conflict, as indicated by the icon to the left of the object name. The conflicting objects are highlighted in the source object repositories.

A description of the conflict and the resolution method used by the Object Repository Merge Tool is described in the Resolution Options pane. A radio button for each possible alternative resolution method is displayed. For information on each of the conflict types, see "Understanding Object Conflicts" on page 277.

2. In the Resolution Options pane, select a radio button to choose an alternative resolution method. The target object repository is updated according to your selection and redisplayed.
3. In the Resolution Options pane, click the **Previous Conflict** or **Next Conflict** buttons to jump directly to the next or previous conflict in the target object repository hierarchy.
4. Repeat steps 1 to 3 to modify additional conflict resolutions, as necessary.
5. Save the target object repository, as described in "Saving the Target Object Repository" on page 285.

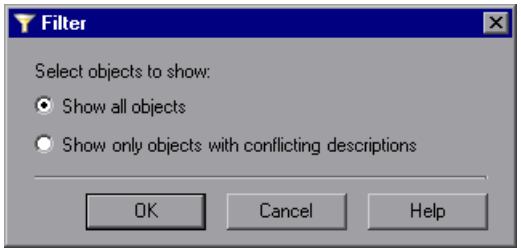
Filtering the Target Repository Pane

Merging two object repositories can result in a target object repository containing a large number of objects. To make navigation and the location of specific objects easier in the target object repository pane, the Object Repository Merge Tool enables you to filter the objects in the pane and show only the objects that had conflicts that were resolved during the merge.

Note: The filter only affects which objects are displayed in the target object repository pane. It does not affect which objects are included in the target object repository.

To filter the objects in the target object repository pane:

1 Select **Tools** > **Filter** or click the **Filter** button. The Filter dialog box opens.



Tip: You can also click the **Filter** icon in the status bar to view the Filter dialog box. The Filter is shown as **ON** in the status bar when a filter is currently in use.

---

2 Select a radio button according to the objects you want to view in the target object repository.

- **Show all objects.** Shows all objects in the target object repository
- **Show only objects with conflicting descriptions.** Shows only objects in the target object repository that have description conflicts

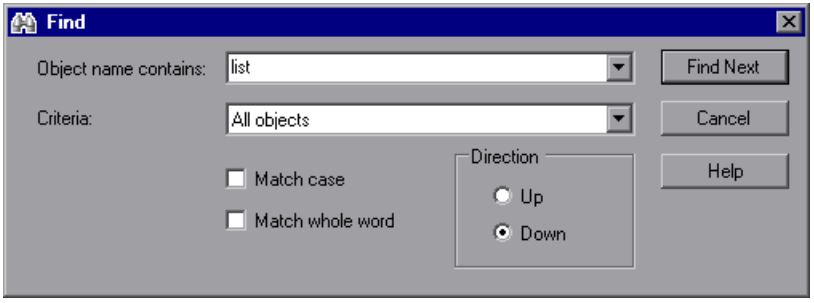
3 Click **OK**. The objects in the pane are filtered and the target object repository displays only the requested object types. A progress bar is displayed in the status bar during the filter process.

Finding Specific Objects

You can use the Find feature in the Object Repository Merge Tool to locate one or more objects in the target object repository whose name contains a specified string. The located object is also highlighted in the relevant primary and/or secondary object repositories.

To find an object:

1 Select **Navigate** > **Find** or click the **Find** button. The Find dialog box opens.



2 In the **Object name contains** box, enter the full or partial name of the object you want to find.

3 In the **Criteria** box, refine your search by selecting which objects to search. The following criteria are available:

- **All objects**
- **Objects from one source**
- **Objects with conflicts**
- **Objects with conflicts or from one source**

4 Select one or both 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 **Object name contains** box.
- **Match whole word.** Searches for occurrences that are whole words only and not part of larger words.

5 Specify the direction from the current cursor location in which you want to search: **Up** or **Down**. The Find operation will continue to search the entire object repository after it reaches the beginning or end of the file.

6 Click **Find Next** to highlight the next object that matches the specified criteria in the target object repository.

You can also close the Find dialog box and use the following commands:

➤ Click the **Find Next** button or select **Navigate** > **Find Next** to highlight the next object that matches the specified criteria.

➤ Click the **Find Previous** button or select **Navigate** > **Find Previous** to highlight the previous object that matches the specified criteria.

Saving the Target Object Repository

When you are sure that the object conflicts are resolved satisfactorily, you can save the target object repository to the file system or to a Quality Center project (if QuickTest is currently connected to the Quality Center project).

Saving the Object Repository

You can save the new merged shared object repository to the file system. If you are connected to Quality Center, you can also save your merged shared object repository in the Test Resources module of your project.

To save an object repository to the file system:

1 Select **File** > **Save** or click the **Save** button. If the file was saved previously, the current changes you made are saved. If the file has not yet been saved, the Save Shared Object Repository dialog box opens.

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. Use a descriptive name that will help you easily identify the file. Do not use any of the following characters in the object repository name:
\ / : * " ? < > | '

If you save the object repository to Quality Center, the file path must not contain two consecutive semicolons (;;).

5 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**.

QuickTest saves the object repository with a **.tsr** extension in the specified location and displays the file name and path above the target object repository in the Object Repository - Merge Tool window.

9

Comparing Shared Object Repositories

QuickTest Professional enables you to compare two shared object repositories using the Object Repository Comparison Tool, and view the differences in their objects, such as different object names, different object descriptions, and so on.

This chapter includes:

Tip: If you are connected to a Quality Center 10.00 project with version control enabled, you can compare two versions of the same object repository. For more information, see "Viewing and Comparing Versions of QuickTest Assets" on page 1461.

About Comparing Shared Object Repositories

QuickTest Professional enables you to compare existing assets from two different object repositories using the Object Repository Comparison Tool. The tool is accessible from the Object Repository Manager, and enables you to compare different object repository resources, or different versions of the same object repository resource, and identify similarities, variations, or changes.

Differences between objects in the two object repository files, named the **First** and **Second** files, are identified according to default rules. During the comparison process, the object repository files remain unchanged. For more information about the types of differences identified by the Object Repository Comparison Tool, see "Understanding Object Differences" on page 297.

After the compare process, the Comparison Tool provides a graphic presentation of the objects in the object repositories, which are shown as nodes in a hierarchy. Objects that have differences, as well as unique objects that are included in one object repository only, can be identified according to a color configuration that you can select. Objects that are included in one object repository only are identified in the other object repository by the text "Does not exist". You can also view the properties and values of each object that you select in either object repository.

You can use the information displayed by the Object Repository Comparison Tool when managing or merging object repositories. For more information, see Chapter 9, "Comparing Shared Object Repositories," or Chapter 8, "Merging Shared Object Repositories."

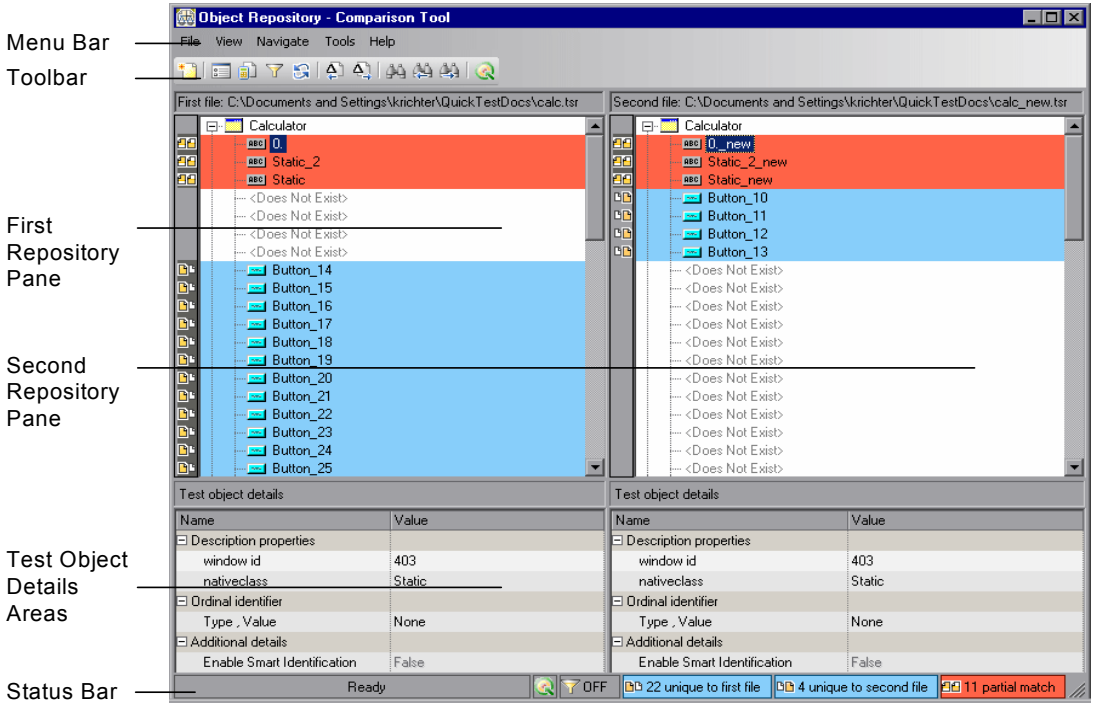
Notes:

- The Object Repository Comparison Tool does not compare checkpoint or output objects.
- You cannot work with the Object Repository Manager or the Object Repository Merge Tool when the Object Repository Comparison Tool is open.

Understanding the Object Repository Comparison Tool

You open the Object Repository Comparison Tool by choosing **Tools** > **Object Repository Comparison Tool** in the Object Repository Manager.

An example window of the Object Repository - Comparison Tool is shown below:



The Object Repository - Comparison Tool window contains the following key elements:

- **Menu bar.** Displays menus of Object Repository Comparison Tool commands. These commands are described in various places throughout this chapter. Shortcut keys for menu commands are described in "Object Repository Comparison Tool Menu Commands and Shortcut Keys" on page 294.
- **Toolbar.** Contains buttons of commonly used menu commands to assist you in comparing your object repositories and viewing the similarities and differences in their objects. Toolbar buttons are described in "Object Repository Comparison Tool Toolbar Commands" on page 293.
- **Repository Panes.** Display a hierarchical view of the objects in the object repositories being compared. In the column to the left of the object hierarchies, each pane displays icons representing the comparison of each object. For more information, see "Understanding the Repository Panes" on page 290.
- **Test Object Details areas.** Show the properties and values of the object selected in an object repository pane. For more information, see "Understanding the Repository Panes" on page 290.
- **Status Bar.** Shows the status of the comparison process and details of the differences found during the object repository comparison. For more information, see "Understanding the Status Bar" on page 292.

Understanding the Repository Panes

The object repository panes display the hierarchies of the objects, and their properties and values, in the object repository files that you are comparing. The file path is shown above each object hierarchy.

To make it easier to see the status of an object at a glance, the text and background of object names in the object repositories are displayed using different colors, according to the type of difference found.

You can change the default colors used in the object repositories to indicate the difference type. For more information, see "Changing Color Settings" on page 298.

Differences can also be identified by the icons used to the left of the objects in the object repository panes, as follows:


➤ Objects that are unique to the first file


➤ Objects that are unique to the second file


➤ Objects in both the first and second file that are not identical but partially match

For more information on all difference types, see "Understanding Object Differences" on page 297.

The object repository panes provide the following functionality:

- When you select an object in one object repository pane, the corresponding object in the other file hierarchy is located and highlighted. You can press the CTRL button when you select an object to highlight only the selected object without highlighting the corresponding object in the other file.
- When you select an object in an object repository pane, its properties and values are displayed in the respective **Test object details** area at the bottom of the pane.
- When you position your cursor over an icon to the left of an object in an object repository pane, the comparison details are displayed as a tooltip, for example, Partial match, or Unique to second file.
- You can expand or collapse the hierarchy of a parent node by double-clicking the node, or by clicking the expand (+) or collapse (-) symbol to the left of the node name. You can also expand or collapse the entire hierarchy in the object repository pane by choosing **Collapse All** or **Expand All** from the **View** menu.


- You can jump directly to the next or previous difference in the object repository hierarchy by choosing **Next Difference** or **Previous Difference** from the **Navigate** menu, by clicking the **Next Difference** or **Previous Difference** buttons in the toolbar, or by using keyboard shortcuts. For more information about shortcuts, see "Object Repository Comparison Tool Menu Commands and Shortcut Keys" on page 294.

- You can locate one or more objects in the object repository panes by using the Find dialog box. For more information, see "Finding Specific Objects" on page 304.
- You can drag the edges of the panes to resize them in the Object Repository Comparison Tool window.

Understanding the Status Bar

The status bar shows information about the comparison process and the results that are displayed:

- A progress bar is displayed on the left of the status bar during the comparison process. **Ready** is displayed when the process is complete.

- The Quality Center icon is displayed when QuickTest is connected to a Quality Center project.

- The filter status is shown next to the Filter icon: **OFF** indicates that the object repositories are not filtered and all objects are shown. **ON** indicates a filter is active and that some objects may have been filtered out of the display. You can click the **Filter** icon to view the Filter dialog box. For more information, see "Filtering the Repository Panes" on page 302.
- The number of differences found during the comparison are displayed, as follows:

  - The number of objects that are unique to the first file

  - The number of objects that are unique to the second file

  - The number of objects in the first and second file that are not identical but partially match

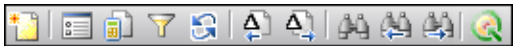
For more information on all difference types, see "Understanding Object Differences" on page 297.

Using Object Repository Comparison Tool Commands

You can select Object Repository Comparison Tool commands from the menu bar or from the toolbar. You can also perform certain commands by pressing shortcut keys.

Object Repository Comparison Tool Toolbar Commands

You can perform frequently used commands by clicking buttons in the toolbar.



	Description
	New Comparison (described in “File Menu Commands” on page 294)
	Color Settings (described in “Tools Menu Commands” on page 296)
	Statistics (described in “View Menu Commands” on page 295)
	Filter (described in “Tools Menu Commands” on page 296)
	Synchronized Nodes (described in “Navigate Menu Commands” on page 295)
	Previous Difference (described in “Navigate Menu Commands” on page 295)
	Next Difference (described in “Navigate Menu Commands” on page 295)
	Find (described in “Navigate Menu Commands” on page 295)
	Find Previous (described in “Navigate Menu Commands” on page 295)
	Find Next (described in “Navigate Menu Commands” on page 295)
	Quality Center Connection (described in “File Menu Commands” on page 294)

Object Repository Comparison Tool Menu Commands and Shortcut Keys

You can perform frequently-used commands by clicking toolbar buttons or choosing the relevant menu option. You can also perform some commands by pressing the relevant shortcut keys.

File Menu Commands

You can manage your object repository comparison using the following **File** menu commands:

	Command	Shortcut Key	Function
	**New Comparison**	CTRL+N	Enables you to specify two object repositories on which to perform a new comparison operation.
	**Quality Center Connection**		Enables you to connect QuickTest to a Quality Center project. For more information, see “Connecting QuickTest to Quality Center” on page 1418.
	Exit		Closes the Object Repository - Comparison Tool window.

View Menu Commands

You can perform the following **View** menu commands:

	Command	Function
	**Statistics**	Opens the Statistics dialog box, which describes the comparison between the two repositories, including the number and type of any differences found. For more information, see "Viewing Comparison Statistics" on page 301.
	Collapse All	Collapses the entire hierarchy in both comparison panes. **Tip:** Double-clicking an expanded node collapses it in both panes simultaneously.
	Expand All	Expands the entire hierarchy in both comparison panes. **Tip:** Double-clicking a collapsed node expands it in both panes simultaneously.

Navigate Menu Commands

You can perform the following **Navigate** menu commands:

	Command	Shortcut Key	Function
	**Next Difference**	F4	Finds the next difference between objects in the object repositories.
	**Previous Difference**	SHIFT+F4	Finds the previous difference between objects in the object repositories.
	**Find**	CTRL+F	Opens the Find dialog box.

	Command	Shortcut Key	Function
	**Find Next**	F3	Finds the next object in the object repositories according to the search specifications in the Find dialog box.
	**Find Previous**	SHIFT+F3	Finds the previous object in the object repositories according to the search specifications in the Find dialog box.

Tools Menu Commands

You can perform the following **Tools** menu commands:

	Command	Function
	**Synchronized Nodes**	Enables you to navigate the two object repository panes simultaneously or independently of one another. For more information, see "Synchronizing Object Repository Views" on page 303.
	**Filter**	Opens the Filter dialog box, enabling you to specify the types of test object matches that you want to show. For more information, see "Filtering the Repository Panes" on page 302.
	**Color Settings**	Opens the Settings dialog box, enabling you to specify the text color and background of the object names and empty nodes displayed in the comparison panes. For more information, see "Changing Color Settings" on page 298.

Help Menu Command

You can perform the following **Help** menu command:

Command	Shortcut Key	Function
Object Repository Comparison Tool Help	F1	Opens the Object Repository Comparison Tool Help.

Understanding Object Differences

The Comparison Tool automatically identifies objects during the comparison process by classifying them into one of the following types:

- **Identical.** Objects that appear in both object repository files. There is no difference in their name or in their properties.
- **Matching description, different name.** Objects that appear in both object repository files that have different names, but the same description properties and values.
- **Similar description.** Objects that appear in both object repository files that have similar, but not identical, description properties and values. One of the objects always has a subset of the properties set of the other object. This implies that it is likely to be a less detailed description of the same object. For example, an object named Button_1 in the second object repository has the same description properties and values as an object named Button_1 in the first object repository, but also has additional properties and values.

  Objects that do not have a description, such as Page or Browser objects, are compared by name only. If the same object is contained in both the object repositories but with different names, they will be shown in the object repositories as two separate objects.

Note: The Object Repository Comparison Tool gives precedence to matching object descriptions over the matching of object names. For this reason, certain object nodes may be linked during the comparison process and not others.

- **Unique to first file**, or **Unique to second file.** Objects that appear in only one of the object repository files.
- **Does not exist.** Objects that do not exist in one of the repository files, but do exist in the other file.

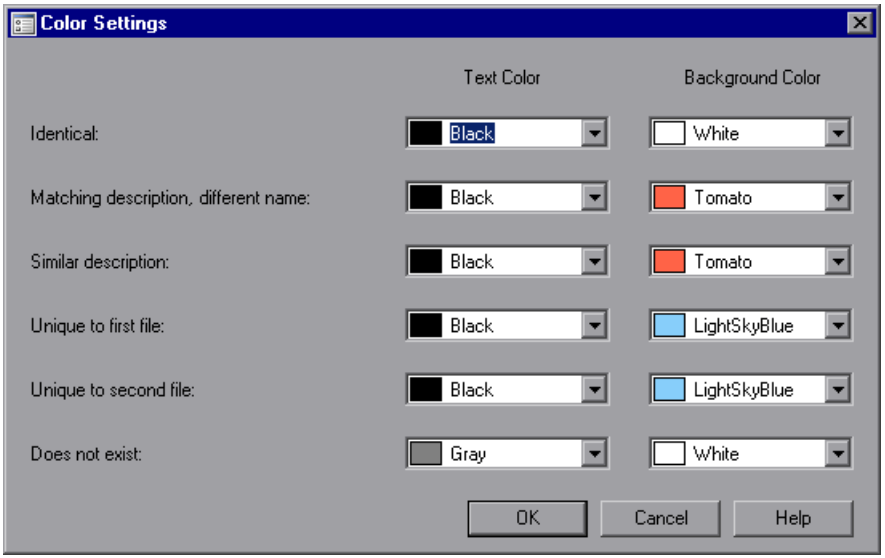
Changing Color Settings

The text and background of object names, and empty nodes representing objects that exist in the other object repository only, are displayed in the Comparison Tool window in default colors, according to their difference types. This enables you to see the status of each object in the object repository panes. These text colors are also used in the Statistics dialog box.

You can change the default color settings if required.

To change color settings:

1 Select **Tools** > **Color Settings** or click the **Color Settings** button in the toolbar. The Color Settings dialog box opens.



2 For each difference type, click the down arrow next to the text box and select an identifying text color and background color from the Custom, Web, or System tabs.

3 Click **OK**. After performing a comparison of object repositories, object names and empty nodes in the respective object repository panes are displayed according to your selections.

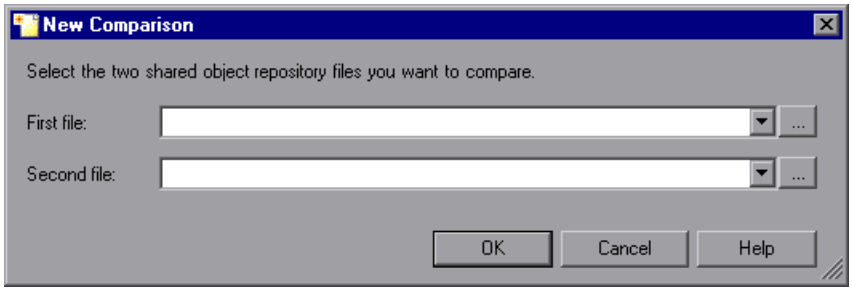
Comparing Object Repositories

Using the Object Repository Comparison Tool, you can compare two object repositories according to predefined settings that define how differences between objects are identified.

To compare two object repositories:

1 In QuickTest Professional, select **Resources** > **Object Repository Manager**.

2 In the Object Repository Manager, select **Tools** > **Object Repository Comparison Tool**. The New Comparison dialog box opens on top of the Object Repository - Comparison Tool window.



Tips:

- If the Object Repository - Comparison Tool window is already open, you can select **File** > **New Comparison** or click the **New Comparison** button in the toolbar to open the New Comparison dialog box.
- If you want to change the color settings before comparing the object repositories, click **Cancel** to close the New Comparison dialog box, change the settings as described in "Changing Color Settings" on page 298, and then perform the comparison.

---

3 In the **First file** and **Second file** boxes, enter or browse to and select the **.tsr** object repository files that you want to compare. The object repository files can be located in the file system or Quality Center. By default, the boxes display the last files selected for comparison using the Object Repository Comparison Tool. You can click the down arrow next to each box to view and select recently used files.

Notes:

- A warning icon is displayed next to the relevant text box if you enter the name of a file without a **.tsr** suffix, a file with an incorrect path, or a file that does not exist. You can position your pointer over the icon to see a tooltip explanation of the error. Enter or select an existing **.tsr** file with the correct path.
- If you want to compare an object repository that was created using a version of QuickTest earlier than version 9.0, you must first open and save it in the Object Repository Manager to update it to the new format.
- If you are connected to Quality Center, you can enter (or browse to) object repositories from Quality Center as well as from the file system.

---

4 Click **OK**. The Object Repository Comparison Tool compares the objects in the selected object repositories and displays the results in the Statistics dialog box on top of the Object Repository - Comparison Tool window.

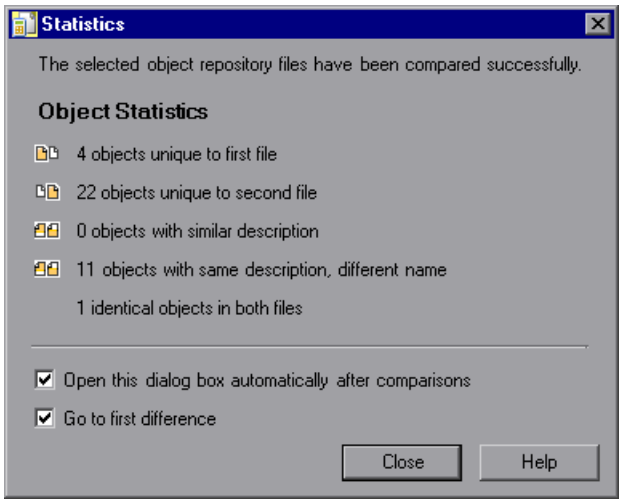
5 Review the statistics, as described in "Viewing Comparison Statistics" on page 301, and click **Close**.

6 In the Object Repository - Comparison Tool window, you can:

- Filter the objects in the object repositories, as described in "Filtering the Repository Panes" on page 302.
- Find specific objects in the object repositories, as described in "Finding Specific Objects" on page 304.

Viewing Comparison Statistics

After you compare two object repositories, the Object Repository Comparison Tool displays the Statistics dialog box, which describes how the files were compared, and the number and type of any differences found.



Tip: You can choose not to view the Statistics dialog box every time you compare object repositories by clearing the **Open this dialog box automatically after comparisons** check box. You can view the comparison statistics in the Statistics dialog box at any time by choosing **View** > **Statistics** in the Comparison Tool window, or by clicking the **Statistics** button in the toolbar.

The Statistics dialog box displays the following information:

- The number and type of any differences between the objects in the object repositories. Difference types are described in "Understanding Object Differences" on page 297.
- The number of items that are unique to the first or the second file, or are identical in both files.

The icons displayed for each difference type in the object statistics are the same as those used in the object repository panes. For more information, see "Understanding the Repository Panes" on page 290.

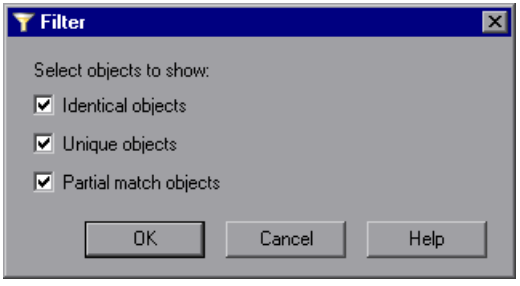
Tip: Select the **Go to first difference** check box to jump to the first difference in the object repositories immediately after you close the Statistics dialog box.

Filtering the Repository Panes

Object repositories can contain a large number of objects. To make navigation and the location of specific objects easier in the object repository panes, the Object Repository Comparison Tool enables you to filter the objects and show only the objects that you want to view.

To filter the objects in the object repository panes:

1 Select **Tools** > **Filter** or click the **Filter** button in the toolbar. The Filter dialog box opens.



Tip: The **Filter** button in the toolbar is surrounded by a border when a filter is currently in use. In addition, the filter is shown as **ON** in the status bar. You can click the **Filter** icon in the status bar to open the Filter dialog box.

---

2 Select one or more check boxes according to the objects you want to view in the object repositories.

- **Identical Objects.** Objects that appear in both object repository files and have no differences in their name or in their properties
- **Unique objects.** Objects that appear only in the first object repository file or only in the second object repository file
- **Partial match objects.** Objects in the object repository files that match but have name or description differences

---

Tip: Select all the check boxes to view all the objects in both object repositories.

---

3 Click **OK**. The objects in the panes are filtered and the object repositories display only the requested object types.

Synchronizing Object Repository Views

The Object Repository Comparison Tool enables you to navigate the two object repositories independently. You can also resize the various panes to display only some of the objects contained in the object repositories. When using large object repositories, this can result in the various panes displaying different areas of the object repository hierarchies, making it difficult to locate and track specific objects affected by the comparison process.

To synchronize the object repositories to display the same object in both views, select the object in the first or second object repository in which it is currently visible and click the **Synchronized Nodes** button in the toolbar. The matching node is highlighted in the other object repository and both object repositories scroll simultaneously.

Tip: The **Synchronized Nodes** button in the toolbar is surrounded by a border when the object repositories are currently synchronized. Click the **Synchronized Nodes** button again to navigate the two object repositories independently. When the object repositories are synchronized, you can also press the CTRL button while selecting an object to highlight the selected object only.

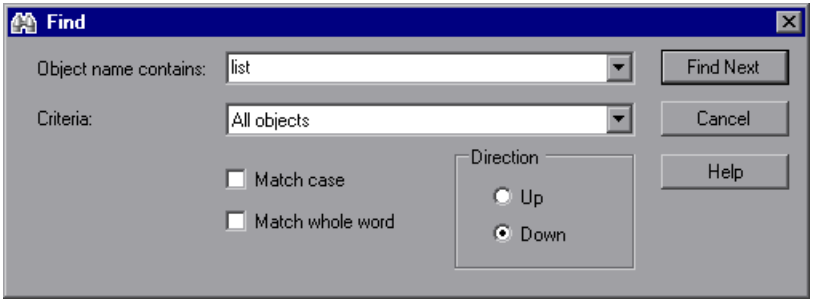
Finding Specific Objects

You can use the Find feature in the Object Repository Comparison Tool to locate one or more objects in a selected object repository whose name contains a specified string. The located object is also highlighted in the other object repository if it exists there.

To find an object:

1 Click the object repository pane that contains the required object.

2 Select **Navigate** > **Find** or click the **Find** button in the toolbar. The Find dialog box opens.



3 In the **Object name contains** box, enter the full or partial name of the object you want to find. You can click the down arrow  next to the box to view and select a recently used string.

4 In the **Criteria** box, refine your search by selecting which objects to search. The following criteria are available:

- **All objects**
- **Unique objects**
- **Partial match objects**
- **Unique or partial match objects**

5 Select one or both 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 **Object name contains** box.
- **Match whole word.** Searches for occurrences that are whole words only and not part of larger words.

6 Specify the direction from the current cursor location in which you want to search: **Up** or **Down**. The Find operation will continue to search the entire file after it reaches the beginning or end of the object repository.

7 Click the **Find Next** button to highlight the next object that matches the specified criteria in the object repository.

You can also close the Find dialog box and use the following commands:

- Click the **Find Next** button in the toolbar, select **Navigate** > **Find Next**, or press F3, to highlight the next object that matches the specified criteria.

- Click the **Find Previous** button in the toolbar, select **Navigate** > **Find Previous**, or press SHIFT+F3, to highlight the previous object that matches the specified criteria.

Part III

Designing Tests

10

Creating Tests — Overview

You can create tests using the keyword-driven methodology, step recording, or a combination of both. The keyword-driven methodology enables you to select keywords to indicate the operations you want to perform on your application. Step recording enables you to record the operations you perform on your application.

After you create your tests, you can enhance them using checkpoints and other special testing options.

This chapter includes:

About Creating Tests

You can create tests using the keyword-driven methodology, step recording, or a combination of both.

Creating tests using the keyword-driven methodology requires an infrastructure for all of the required resources. Resources include shared object repositories, function libraries, and recovery scenarios. Setting up the infrastructure requires in-depth knowledge of your application and a high level of QuickTest expertise.

Although setting up the infrastructure may initially require a longer time-investment in comparison to recording tests, using the keyword-driven methodology enables you to create tests at a more application-specific level and with a more structured design. This enables you to maintain your tests more efficiently and provides you with more flexibility than a recorded test.

In some cases, you may want to let QuickTest generate test steps by recording the typical processes that you perform on your application. As you navigate through your application, QuickTest graphically displays each step you perform as a row in the Keyword View. A step is anything a user does that changes the content of a page or object in your application, for example, clicking a link or typing data into an edit box. Recording may be easier for new QuickTest users or when beginning to design tests for a new application or a new feature.

While creating your test, you can insert checkpoints. A **checkpoint** compares the value of an element captured when the object was saved in the object repository, with the value of the same element captured during the run session. This helps you determine whether or not your application is functioning correctly. For more information, see "Understanding Checkpoints" on page 495.

When you test your application, you may want to check how it performs the same operations with different data. This is called **parameterizing** your test. You can supply data in the Data Table, define environment variables, instruct QuickTest to generate random numbers, and so on. For more information, see "Parameterizing Values" on page 625.

After creating your initial test, you can further enhance it by adding and modifying steps in the Keyword View or Expert View.

Deciding Which Methodology to Use - Keyword-Driven or Recording

You can create the steps in your tests using the keyword-driven methodology, recording, or a combination of both.

Recording Tests

Recording can be useful in the following situations:

- Recording helps novice QuickTest users learn how QuickTest interprets the operations you perform on your application, and how it converts them to QuickTest objects and built-in operations.
- Recording can be useful for more advanced QuickTest users when working with a new application or major new features of an existing application (for the same reasons described above). Recording is also helpful while developing functions that incorporate built-in QuickTest keywords.
- Recording can be useful when you need to quickly create a test that tests the basic functionality of an application or feature, but does not require long-term maintenance.

For information on recording tests, see "Creating Tests Using the Recording Mechanism" on page 361.

Creating Tests Using Keyword-Driven Testing

Keyword-driven testing advantages include the following:

- Keyword-driven testing enables you to design your tests at a business level rather than at the object level. For example, QuickTest may recognize a single option selection in your application as several steps: a click on a button object, a mouse operation on a list object, and then a keyboard operation on a list sub-item. You can create an appropriately-named function to represent all of these lower-level operations in a single, business-level keyword.
- By incorporating technical operations, such as a synchronization statement that waits for client-server communications to finish, into higher level keywords, tests are easier to read and easier for less technical application testers to maintain when the application changes.

- Keyword-driven testing naturally leads to a more efficient separation between resource maintenance and test maintenance. This enables the automation experts to focus on maintaining objects and functions while application testers focus on maintaining the test structure and design.
- When you record tests, you may not notice that new objects are being added to the local object repository. This may result in many testers maintaining local object repositories with copies of the same objects. When using a keyword-driven methodology, you select the objects for your steps from the existing object repository. When you need a new object, you can add it to your local object repository temporarily, but you are also aware that you need to add it to the shared object repository for future use.
- When you record a test, QuickTest enters the correct objects, methods, and argument values for you. Therefore, it is possible to create a test with little preparation or planning. Although this makes it easier to create tests quickly, such tests are harder to maintain when the application changes and often require re-recording large parts of the test.

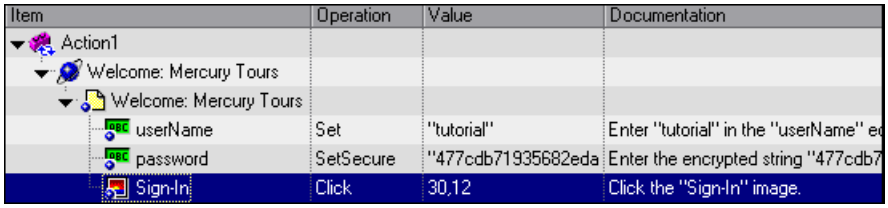
  When you use a keyword-driven methodology, you select from existing objects and operation keywords. Therefore, you must be familiar with both the object repositories and the function libraries that are available. You must also have a good idea of what you want your test to look like before you begin inserting steps. This usually results in well-planned and better-structured tests, which also results in easier long-term maintenance.
- Automation experts can add objects and functions based on detailed product specifications even before a feature has been added to a product. Using keyword-driven testing, you can begin to develop tests for a new product or feature earlier in the development cycle.

  For information on creating tests using the keyword-driven methodology, see "Creating Tests Using the Keyword-Driven Methodology" on page 335.

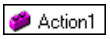
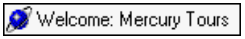
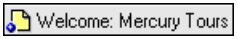
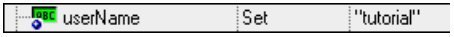
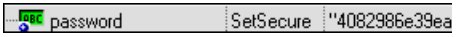
Understanding Your Test

When you create a test, QuickTest creates a graphical representation of the steps you perform on your application. These steps are displayed in the Keyword View tab.

The following is a sample test of a login procedure to the Mercury Tours site, the sample Web site.



The table below provides an explanation of each step in the Keyword View.

Step	Description
	**Action1** is the action name.
	The browser invokes the **Welcome: Mercury Tours** Web site.
	**Welcome: Mercury Tours** is the name of the Web page.
	**userName** is the name of the edit box. Set is the method performed on the edit box. **tutorial** is the value of the edit box.
	**password** is the name of the edit box. SetSecure is an encryption method performed on the edit box. **4082986e39ea469e70dbf8c5a29429fe138c6efc** is the encrypted value of the password.
	**Sign-In** is the name of the image link. Click is the method performed on the image. **2, 2** are the x- and y-coordinates where the image was clicked.

In the Expert View, these same steps are displayed using a VBScript program based on the QuickTest object model.

```
Browser("Welcome: Mercury Tours").Page("Welcome: Mercury Tours").
    WebEdit("userName").Set "tutorial"
Browser("Welcome: Mercury Tours").Page("Welcome: Mercury Tours").
    WebEdit("password").SetSecure
    "4082986e39ea469e70dbf8c5a29429fe138c6efc"
Browser("Welcome: Mercury Tours").Page("Welcome: Mercury Tours").
    Image("Sign-In").Click 2,2
```

Enhancing Your Test

You can use a variety of options to enhance your existing tests. This section describes some of the ways in which you can enhance your existing tests.

Checkpoints

You can add checkpoints to your test. A **checkpoint** is a step in your test that compares the a specified item during a run session with the values stored for the same item within the test. This enables you to identify whether or not your application is functioning correctly. There are several different checkpoint types. For more information on creating checkpoints, see Chapter 17, "Understanding Checkpoints."

Tip: You can also use the CheckProperty method, which enables you to verify the property value of an object without using the checkpoint interface. For more information, see *HP QuickTest Professional Object Model Reference*.

Parameterization

You can parameterize your test to replace fixed values with values from an external source during your run session. The values can come from a Data Table, environment variables you define, or values that QuickTest generates during the run session. For more information, see Chapter 24, "Parameterizing Values."

Output Values

You can retrieve values from your test and store them in the Data Table as output values. You can subsequently use these values as an input parameter in your test. This enables you to use data retrieved during a test in other parts of the test. For more information, see Chapter 25, "Outputting Values."

Actions

You can divide your test into actions to streamline the testing process of your application. For more information, see Chapter 15, "Working with Actions."

Programming Statements

You can use special QuickTest options to enhance your test with programming statements. The Step Generator guides you step-by-step through the process of adding recordable and non-recordable operations (methods and properties) to your test. You can also synchronize your test to ensure that your application is ready for QuickTest to perform the next step in your test, and you can measure the amount of time it takes for your application to perform steps in a test by defining and measuring transactions. For more information, see Chapter 28, "Adding Steps Containing Programming Logic."

You can also manually enter standard VBScript statements, as well as statements using QuickTest test objects and operations, in the Expert View. For more information, see Chapter 29, "Working in the Expert View and Function Library Windows."

Using Relative Paths in QuickTest

QuickTest enables you to define the path to a resource that you are adding to the file system or to Quality Center, as a relative or an absolute path. (For information about relative or absolute paths, see "Understanding Absolute and Relative Paths" on page 319.)

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.

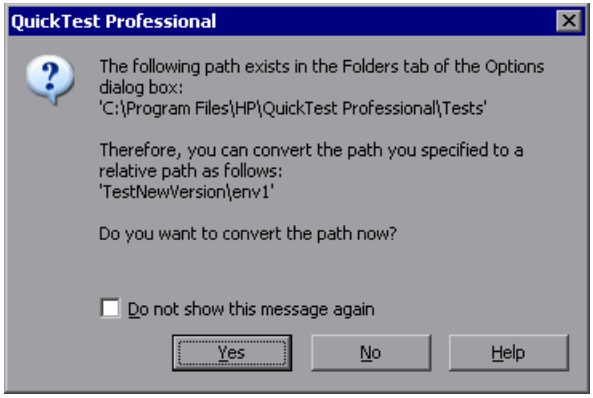
When you specify a path to a function library, shared object repository, recovery scenario, or environment variable file, QuickTest checks if the path, or the initial part of the path, exists in the Folders pane of the Options dialog box (**Tools > Options > Folders** node). The Folders pane contains a search list in which you can define where QuickTest searches for tests, actions, or files.

QuickTest then opens one of the following two dialog boxes, depending on whether the path you specified, or a part of the path, exists in the Folders pane.

Note: If you are connected to Quality Center 10.00, these dialog boxes are displayed only if you select a path in the file system or in a Quality Center 9.x project.

Path Exists in the Folders Pane

If the resource path you specify matches an existing search path in the Folders pane, you are prompted whether to define the path using only the relative part of the path.

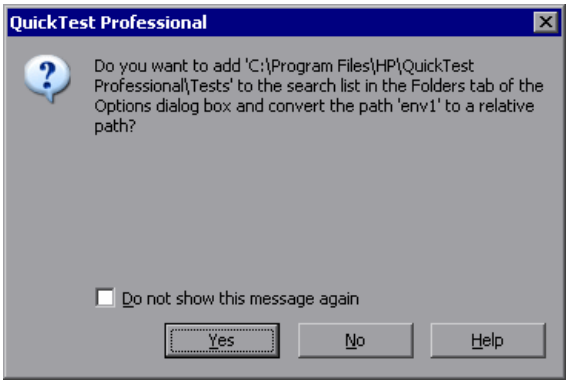


- Click **Yes** to truncate the path to a relative path.
- Click **No** to define the path to the resource as an absolute path.

In cases where a part of the path you enter matches more than one path in the Folders pane, the closest match is applied. For example, if both C:\Current_Version and C:\Current_Version\Libraries are defined in the search path list, the latter is applied.

Path Does Not Exist in the Folders Pane

If the resource path you specify does not match an existing search path in the Folders pane, you are prompted whether to add the resource's location path to the Folders pane and define the path relatively.



- Click **Yes** to add the resource's location path to the Folders pane and truncate the path to a relative path.
- Click **No** to define the path to the resource as an absolute path.

Notes:

- You can choose not to show one or both of these dialog boxes when you enter a path to a resource by selecting the **Do not show this message again** check box. To show these dialog boxes again, select the **Remind me to use relative paths when specifying a path to a resource** check box in the Folders pane of the Options dialog box. This check box is selected by default when you first start QuickTest.
- For more information on the Folders pane, which enables you to enter the folders (search paths) in which QuickTest searches for searches for tests, actions, or files, see “Setting Folder Testing Options” on page 1237.

Understanding Absolute and Relative Paths

You can save QuickTest resources, such as shared object repositories, function libraries, recovery scenarios or environments, using absolute or relative paths.

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.

- An **absolute** path describes the full path to a specific file starting from a fixed location such as the root directory, or the drive on which the file is located, and contains all the other sub-directories in the path. An absolute path always points to the specified file, regardless of the current directory.
- A **relative** path describes the path to a specific file starting from a given directory, and is generally only a portion of the absolute path. A relative path therefore specifies the location of the file relative to the given location in the file system.

  Using relative paths means that the paths remain valid when files or folders containing files are moved or copied to other locations or computers, provided that they are moved within the same folder structure. For this reason, we recommend that you use relative paths when saving resources in QuickTest.

For example, consider a QuickTest resource file named FunctionLibrary1.qfl located in C:\Current_Version\Libraries. The absolute path to the file is C:\Current_Version\Libraries\FunctionLibrary1.qfl. The relative path to the file from within the folder named Libraries is specified using only the name of the file, FunctionLibrary1.qfl. Alternatively, the relative path to the file from within another folder, such as C:\Current_Version\Libraries\MyFiles, would be Libraries\FunctionLibrary1.qfl.

Using a relative path, you could copy the FunctionLibrary1.qfl file from C:\Current_Version\Libraries to an updated version in C:\New_Version\Libraries, and the path used by QuickTest would remain valid.

In addition, relative paths are quicker to type and, being shorter, minimize any chance for error.

For more information, see "Using Relative Paths in QuickTest" on page 316.

Note: Prior to QuickTest 9.0, if you specified a path for a resource starting with \.., it was considered to be a relative path. In QuickTest 9.0 and later, a path that starts with \.. is considered to be a full path, with the backslash representing the root folder of the current drive.

If you defined paths starting with \.. using earlier versions of QuickTest, you should change the path to be a standard relative path by removing the backslash (\).

11

Managing Your Test

You can use the **File** menu to create, open, save, zip, unzip, and print tests, as well as create standalone, portable tests.

Tip: As the content of your application changes, you can update the selected Active Screen display and use the Active Screen to add new steps to your test instead of re-recording steps on new or modified objects. For more information, see "Updating a Test Using the Update Run Mode Option" on page 1125.

This chapter includes:

Creating a New Test

To create a new test, click the **New** button or select **File** > **New** > **Test**. A new test opens, with a new action selected in the Keyword View. You are ready to start creating your test.

Opening an Existing Test

You can open an existing test to enhance or run it.

To open a test stored in Quality Center, QuickTest must be connected to the Quality Center project. For more information, see Chapter 51, "Integrating with Quality Center."

If the test you select was last saved in an older version of QuickTest, you may be asked whether to convert the test to the current version or view it in read-only format. For more information, see "Considerations for Opening Tests Created in Previous Versions of QuickTest" on page 323.

To open an existing test:

1. (Optional) Connect to a Quality Center server and project. For more information, see "Connecting to and Disconnecting from Quality Center" on page 1418.
2. Select **File** > **Open** > **Test**, or click the **Open** down arrow and select **Test**. The Open Test dialog box opens.
3. In the sidebar, select the location of the test, for example, File System or Quality Center Test Plan.
4. Browse to and select a test. You can select the **Open in read-only mode** option at the bottom of the dialog box. Click **Open**. The test opens and the title bar displays the test name.

Note: If the test is stored in a version control-enabled Quality Center project, the **Open** button contains a down arrow, enabling you to open the test and immediately check it out. For more information, see "Checking Assets Out of the Version Control Database" on page 1483.

Tip: You can open a recently used test by selecting it from the Recent Files list in the **File** menu.

Considerations for Opening Tests Created in Previous Versions of QuickTest

- 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. You install this tool from the QuickTest Professional installation DVD. After installation, this tool is available from the **Start** menu by choosing **Start** > **Programs** > **QuickTest Professional** > **Tools** > **QuickTest Professional Asset Upgrade Tool**.
- When you open a test that was created using an older version of QuickTest, you may be asked whether you want to convert it or view it in read-only format.
  - If the test contains objects in the local object repositories of one or more actions in the test, the relevant add-in must be installed to convert the test to the current format. Otherwise, it is opened in read-only format.
  - If you choose to convert the test, it is updated to the current format and you can modify it as needed. If you save the converted test, it cannot be used with earlier versions of QuickTest.
  - If you choose to view the test in read-only format, it appears as it did previously, using all of its original settings, but you cannot modify it.
  - If you have many tests that need to be updated to the current format, you can create an automation script that iterates through all of your tests to open and save each one in the new format.

    For more information on creating automation scripts, see Chapter 50, "Automating QuickTest Operations."

    To view a sample automation script that converts old tests to the current version, see the *QuickTest Professional Automation Object Model Reference* (**Help** > **QuickTest Professional Help** > **HP QuickTest Professional Advanced References** > **HP QuickTest Professional Automation Object Model**).
- You cannot open a test that was created with a later version of QuickTest on a computer running an earlier version of QuickTest. For example, you cannot open a test created in QuickTest 10.00 on a computer running QuickTest 8.0.

Saving a Test

You can save a new test or save changes to an existing test.

Tip: If changes are made to an existing test, an asterisk (*) is displayed in the title bar until the test is saved.

Note: You must use the **Save As** option in QuickTest if you want to save a test under another name or create a copy of a test. You cannot copy a test or change its name directly in the file system or in Quality Center.

To save a new test:

1. (Optional) Connect to a Quality Center server and project. For more information, see "Connecting to and Disconnecting from Quality Center" on page 1418.

2. Click the **Save** button or select **File** > **Save** to save the test. The Save QuickTest Test dialog box opens.
3. In the sidebar, select the location to save the test, for example, File System or Quality Center Test Plan.
4. Browse to and choose the folder in which to save the test.

Note: In the file system, QuickTest suggests a default folder called **Tests**. For all supported operating systems except Windows Vista, this folder is located under your QuickTest Professional installation folder. For Windows Vista, this folder is located under **MyDocuments\HP\QuickTest Professional**.

---

5 Type a name for the test in the **File name** box. Note that the test name cannot exceed 220 characters (including the path), cannot begin or end with a space, and cannot contain the following characters:
\ / : * ? " < > | % '

If you save the test to Quality Center, the file path must not contain two consecutive semicolons (;;).

6 If you are recording and want to save the Active Screen files with your test, confirm that **Save Active Screen files** is selected.

If you clear this box, your Active Screen files will not be saved, and you will not be able to edit your test using the options that are normally available from the Active Screen.

Clearing the **Save Active Screen files** check box can be especially useful for conserving disk space once you have finished designing the test and you are using the test only for test runs.

Tip: If you clear the Save Active Screen files check box and then later want to edit your test using Active Screen options, you can regenerate the Active Screen information by performing an **Update Run** operation. For more information, see "Updating a Test Using the Update Run Mode Option" on page 1125.

Note: You can also instruct QuickTest not to capture Active Screen files while recording or to only capture Active Screen information under certain conditions. You can set these preferences in the Active Screen pane of the Options dialog box. For more information, see "Setting Active Screen Options" on page 1240.

---

7 Click **Save**. QuickTest displays the test name in the title bar.

To save changes to an existing test:

- Click the **Save** button to save changes to the current test.
- Select **File** > **Save As** to save an existing test to a new name or a new location. If you select **File** > **Save As**, the following options are available:
  - Select or clear the **Save Active Screen files** check box to indicate whether or not you want to save the Active Screen files with the new test. For more information, see step 6 above.
  - Select or clear the **Save test results** check box to indicate whether or not you want to save any existing test results with your test.

    Note that if you clear this box, your test result files will not be saved, and you will not be able to view them later. Clearing the **Save test results** check box can be useful for conserving disk space if you do not require the test results for later analysis, or if you are saving an existing test under a new name and do not need the test results.

Creating Portable Copies of Your Tests

Tests and their resource files are often stored on a network drive or in Quality Center, as this enables the reuse of actions and other resources, and helps ease test management.

Sometimes, you may need to open or run a test when you do not have access to a network drive or Quality Center. For example, you may need to create a portable copy of a test for use when travelling to other sites. You can save a standalone copy of your test and its resource files to a local drive or to another storage device using the **File** > **Save Test with Resources** command.

When you save a test in this manner, QuickTest creates a copy of the following and saves the files in the location you specify:

- **Source test**. QuickTest saves a copy of this test in the location you specify.
- **Resource files.** QuickTest saves a copy of all resource files associated with the source test, such as function libraries and shared object repositories. QuickTest stores these files in sub-folders of the copied test.
- **Called actions.** QuickTest saves a copy of any external actions called by the source test. For example, if Test A calls actions that are stored in Test B, QuickTest creates a local copy of the actions stored in Test B and stores them in a sub-folder of Test A. The sub-folder has the same name as the test from which the called actions were copied. In this example, the sub-folder is named Test_B. QuickTest also creates a copy of any resources associated directly with these actions, such as its local shared object repositories and action sheets in the Data Table. QuickTest does not, however, save the resource files associated with Test B, so you must ensure that these resources are associated with the source test, Test A.

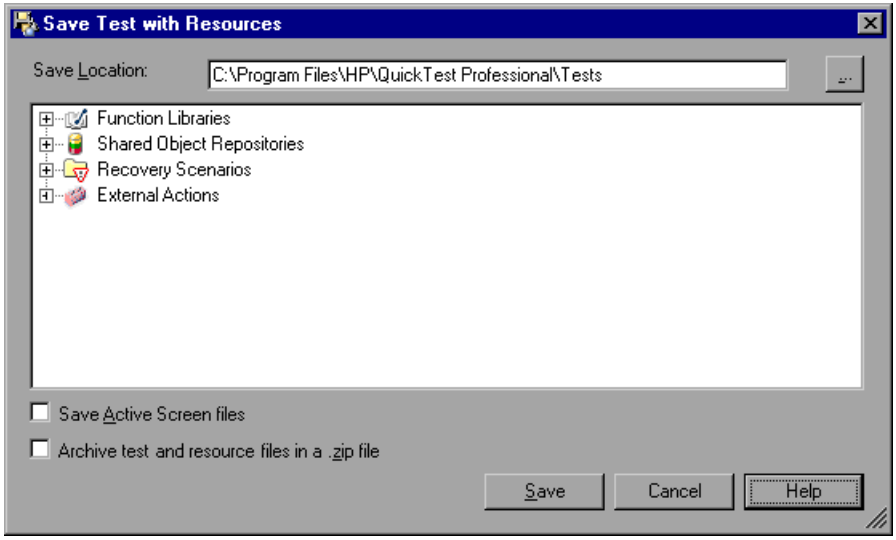
  This enables you to modify or run the test without access to a network drive or Quality Center.

Tip: If you use QuickTest with a concurrent license but do not have access to the concurrent license server (for example, during a business trip), you can install a commuter license. For more information, see the *HP QuickTest Professional Installation Guide*.

The Save Test with Resources Dialog Box

Description	Enables you to save a full copy of a test and its resource files to a local drive or other storage device, eliminating the need for network or Quality Center connections.
How to Access	➤ Select the **File** > **Save Test with Resources** menu command. ➤ Click the **Save Test with Resources** toolbar button  .
Important Information	**Before you create a copy of the test:** ➤ Resolve any missing resources. ➤ Save the original test. ➤ Make sure that all files associated with the source test are writable. ➤ Make sure you have write permissions for the folder in which you want to create a copy of the test. **After you make a copy of the test:** ➤ A report is displayed in HTML format, listing: ➤ the name of the test, the name of the user that saved this copy of the test, and the date on which the test was copied. ➤ a record for each resource that was copied with the test, specifying: -- the name of the resource -- the type of resource (for example, function library) -- the path from which the resource was copied -- the status of the copied resource (for example, the resource was saved successfully) -- the current location of the copied resource You can also open this file from the copied test's root folder. ➤ The copied test becomes the active test in the QuickTest window. ➤ All links to the source files are severed. Therefore, any modifications you make to the copied test are applied only to the copied test.
Learn More	**Conceptual overview**: "Creating Portable Copies of Your Tests" on page 326 **Additional related topic**: "Guidelines for Working with Tests Created Using an Earlier Version of QuickTest" on page 330

Below is an image of the Save Test with Resources dialog box:



Save Test with Resources Dialog Box Options

Option	Description
Save Location	Specifies the root folder in which to save the test. By default, the root folder is **<QuickTest installation folder>\Tests**, however, you can specify any folder on a local, network, or portable drive. **Important:** The folder you specify must not already contain a sub-folder with the same name as the test.
Resources tree	Lists the external resources that are currently associated with or attached to your test.

Option	Description
Save Active Screen files	(Relevant only for recorded tests.) Instructs QuickTest to save any existing Active Screen files with your test. Clearing the Save Active Screen files check box can be especially useful for conserving disk space if you have finished designing the test and are using the test only for test runs. **Note:** If you clear this box, your Active Screen files will not be copied over with the test and its resources, and you will not be able to edit your test using the options that are normally available from the Active Screen. **Tip:** If you clear the Save Active Screen files check box and then later want to edit your test using Active Screen options, you can regenerate the Active Screen information by performing an Update Run operation. For more information, see "Updating a Test Using the Update Run Mode Option" on page 1125.
Archive test and resource files in a .zip file	Creates a .zip file of the test and its resources, and stores the .zip file in the folder you specified in the **Save Location** box. For more information, see "Zipping a Test" on page 331.

Guidelines for Working with Tests Created Using an Earlier Version of QuickTest

Before you can save a standalone copy of a test that was created in an earlier version of QuickTest, you must upgrade the test and its resource files to the current version of QuickTest, as follows:

- Open the test in QuickTest and save it (**Save** or **Save As**). If the test contains calls to external actions (actions stored in other tests), you must open and save those tests, too.
- Alternatively, if your tests are stored in Quality Center, you can use the QuickTest Professional Asset Upgrade Tool for Quality Center. This converts your test's attached resource files to linked assets and upgrades your tests to the current version of QuickTest.

Zipping a Test

QuickTest tests contain a series of configuration, run-time, setup data, and (optionally) Active Screen files. QuickTest saves these files together with the test. You can zip these files to conserve space and make the tests easier to transfer.

To zip a test:

1 Do one of the following:

- Select **File** > **Export Test to Zip File** to open the Export to Zip File dialog box.
- Select the **Archive test and resource files in a .zip file** check box in the Save Test with Resources dialog box (**File** > **Save Test with Resources**). For more information, see "The Save Test with Resources Dialog Box" on page 328.

2 Type a zip file name and path, or accept the default name and path, and click **OK**. QuickTest zips the test and its associated files.

Unzipping a Test

You can unzip a test when needed.

To unzip a zipped test:

1 Select **File** > **Import Test from Zip File**. The Import from Zip File dialog box opens.

2 Enter or select the name of the zip file that you want to unzip, choose a target folder into which you want to unzip the files, and click **OK**. QuickTest unzips the test and its associated files.

Printing a Test

You can print your entire test from the Keyword View (in table format). You can also print a single action either from the Keyword View (in table format) or the Expert View (in statement format). When printing from the Expert View, you can also specify additional information that you want to be included in the printout.

To print from the Keyword View:

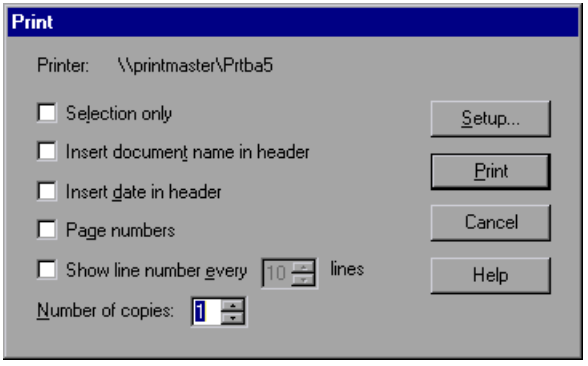
1 Click the **Print** button or select **File** > **Print**. A standard Print dialog box opens.

2 Click **OK** to print the content of the Keyword View to your default Windows printer.

Tip: You can select **File** > **Print Preview** to display the Keyword View on screen as it will look when printed. Note that the **Print Preview** option works only with tests created using QuickTest 8.0 and later.

To print from the Expert View:

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 Expert View.
- **Insert document name in header.** Includes the name of the active test or 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.

12

Creating Tests Using the Keyword-Driven Methodology

You can create a test using the keyword-driven methodology, which enables you to select keywords to indicate the operations you want to perform on your application. This enables you to create structured tests that are easier to update and maintain over time.

The keyword-driven methodology is especially useful for organizations that have both technical and less technical users because it offers a clear division of automation tasks. This enables a few experts to maintain the resource framework while less technical users design and maintain automated test steps. Additionally, once the basic infrastructure is in place, both types of users can often do their tasks simultaneously.

Before you begin creating tests, you need to plan your tests to ensure that your tests cover your testing requirements. For more information on planning your tests, see "Creating Tests — Overview" on page 309.

After you create your test, you can enhance it using checkpoints and other special testing options.

Tip: You can also create a test by recording the operations you perform on your application, as described in "Creating Tests Using the Recording Mechanism" on page 361. After you create your test, you can enhance it using checkpoints and other special testing options.

This chapter includes:

Understanding the Keyword-Driven Methodology

Keyword-driven testing is a technique that separates much of the programming work from the actual test steps so that the test steps can be developed earlier and can often be maintained with only minor updates, even when the application or testing needs change significantly.

This section provides a general overview of the steps you perform when planning and implementing your tests.

Stage 1: Analyzing Your Application

Before you begin creating a test, you need to analyze your application and determine your testing needs.

First, determine the development environments in which your application controls were developed, such as Web, Java, or .NET, so that you can load the required QuickTest add-ins.

Then determine the functionality that you want to test. To do this, consider the various activities that customers perform in your application to accomplish specific tasks. Which objects and operations are relevant for the set of business processes that need to be tested? Which operations require customized keywords to provide additional functionality?

While you are thinking about the business processes you want to test, consider how you can divide these processes into smaller units, which will be represented by your test's actions. Each action should emulate an activity that a customer might perform when using your application.

As you plan, try to keep the amount of steps you plan to include in each action to a minimum. Creating small, modular actions helps make your tests easier to read, follow, and maintain.

Stage 2: Preparing the Testing Infrastructure

To complete the infrastructure that is part of the planning process, you need to build the set of resources to be used by your tests, including shared object repositories containing test objects (which are representations of the objects in your application), function libraries containing functions that enhance QuickTest functionality, and so on. For more information, see Chapter 5, "Managing Test Objects in Object Repositories" and Chapter 31, "Working with User-Defined Functions and Function Libraries."

At this stage you also need to configure QuickTest according to your testing needs. This can include setting up your global testing preferences, your run session preferences, any test-specific preferences, and recovery scenarios. You can also create automation scripts that automatically set the required configurations (such as the add-ins to load) on the QuickTest client at the beginning of a run session. For more information, see Chapter 50, "Automating QuickTest Operations."

Lastly, you create one or more tests that serve as action repositories in which you can store the actions to be used in your tests. Generally, you create an action repository test for each area of your application to be tested. Storing all of your actions in specific tests enables you to maintain your actions in a central location. When you update an action in the action repository, the update is reflected in all tests that contain a call to that action. When you run a test, only the relevant action repository tests are loaded.

You then associate the shared object repositories with the relevant actions. This enables you to later insert steps using the objects stored in the object repositories.

When you create your tests, you insert calls to one or more of the actions stored in this repository.

Stage 3: Adding Steps to Your Actions

In this stage, you add steps to the actions in your test action repository.

Before you begin adding steps, make sure that you associate your function libraries and recovery scenarios with the relevant tests, so that you can insert steps using keywords.

You can create steps using the keyword-driven functionality available in the table-like, graphical Keyword View—or you can use the Expert View, if you prefer to program steps directly in VBScript. You can add steps to your test in one or both of the following ways:

- Drag objects from your object repository or from the Available Keywords pane to add keyword-driven steps in the Keyword View or Expert View. The object repository and Available Keywords pane contain all of the objects that you want to test in your application. (You create one or more object repositories when you prepare the testing infrastructure, as described in "Stage 2: Preparing the Testing Infrastructure" on page 337.)

  When you drag an object into the Keyword View, a step is created in the action with the default operation for that object. For example, if you drag a button object into the Keyword View, the click operation is automatically defined for the step. You can then modify the step as needed. For more information, see Chapter 14, "Working with the Keyword View" and Chapter 39, "Adding Keywords to Your Test." Advanced users can also add steps using the Expert View. For more information, see Chapter 29, "Working in the Expert View and Function Library Windows."

- Record on your application.

  As you navigate through your application during a recording session, QuickTest graphically displays each step you perform as a row in the Keyword View. A step is something that causes or makes a change in your application, such as clicking a link or image, or submitting a data form. In the Expert View, these steps are displayed as lines in a test script (VBScript). The **Documentation** column of the Keyword View also displays a description of each step in easy-to-understand sentences. For more information, see Chapter 14, "Working with the Keyword View."

Stage 4: Enhancing Your Test

You can enhance the testing process by modifying your test with special testing options and/or with programming statements, such as:

- Insert checkpoints and output values into your test.

  A **checkpoint** checks specific properties or other characteristics of an object and enables you to identify whether or not your application is functioning correctly. For more information, see Chapter 17, "Understanding Checkpoints."

  You can also use output values to extract data from your test. An **output value** is a value retrieved during the run session and entered into your Data Table or stored in a variable or a parameter. You can subsequently use this output value as input data in your test. This enables you to use data retrieved during a run session in other parts of the test. For more information, see Chapter 25, "Outputting Values."

- Broaden the scope of your test by replacing fixed values with parameters.

  When you test your application, you can parameterize your steps to check how your application performs the same operations with different data. You may supply data in the Data Table, define environment variables and values, define test or action parameters and values, or instruct QuickTest to generate random numbers for current user and test data.

  When you parameterize your test, QuickTest substitutes the fixed values in your test with the values stored in the relevant parameters. When you use Data Table parameters, QuickTest uses the values from a different row in the Data Table for each iteration of the test or action. (Each run session that uses a different set of parameterized data is called an iteration.) For more information, see Chapter 24, "Parameterizing Values."

- Add user-defined functions by creating function libraries and calling their functions from your test. For more information, see Chapter 31, "Working with User-Defined Functions and Function Libraries."

- Use the many functional testing features included in QuickTest to enhance your test and/or add programming statements to achieve more complex testing goals. For more information, see Chapter 28, "Adding Steps Containing Programming Logic."

Stage 5: Running and Debugging Your Test

After you create your test, you can perform different types of runs to achieve different goals.

- **Run your test to debug it.** You can control your run session to help you identify and eliminate defects in your test. You can use the **Step Into**, **Step Over**, and **Step Out** commands to run your test step by step. You can begin your run session from a specific step in your test, or run the test until a specific step is reached. You can also set breakpoints to pause your test at predetermined points. You can view or change the value of variables in your test each time it stops at a breakpoint in the Debug Viewer. You can also manually run VBScript commands in the Debug Viewer. For more information, see Chapter 35, "Debugging Tests and Function Libraries."
- **Run your test to check your application.** The test starts running from the first line in your test and stops at the end of the test. While running, QuickTest connects to your application and performs each operation in your test, including any checkpoints, such as checking any text strings, objects, tables, and so forth. If you parameterized your test with Data Table parameters, QuickTest repeats the test (or specific actions in your test) for each set of data values in the Data Table. For more information, see Chapter 32, "Running Tests."
- **Run your test to update it.**
  - You can run your test using **Maintenance Run Mode** when you know that your application has changed, and you therefore expect that QuickTest will not be able to identify the objects in your test. When you run a test in Maintenance Run Mode, a wizard opens for steps that fail because an object could not be found in the application. The wizard then guides you through the steps of resolving the issue, and, after you resolve the issue, the run continues. For more information, see Chapter 36, "Maintaining Tests."
  - You can run your test using **Update Run Mode** to update the property sets used for test object descriptions, the expected checkpoint values, the data available to retrieve in output values, and/or the Active Screen images and values.

Stage 6: Analyzing Test Results and Reporting Defects

After you run your test, you can view the results of the run in the Test Results window. You can view a summary of your results as well as a detailed report. If you captured still images or movies of your application during the run, you can view these from the Screen Recorder tab of the Test Results window. For more information, see Chapter 33, "Viewing Run Session Results." If you enabled local system monitoring for your test, you can view the results in the System Monitor tab of the Test Results window. For more information, see "Viewing System Monitor Results" on page 1063.

Finally, you can report defects detected during a run session. If you have access to Quality Center, the HP centralized quality solution, you can report the defects you discover to the project database. You can instruct QuickTest to automatically report each failed step in your test, or you can report them manually from the Test Results window. For more information, see Chapter 51, "Integrating with Quality Center."

Using the Keyword-Driven Methodology

By creating your tests with a keyword-driven methodology in mind, your tests become more modular, focusing on the operations to test using both QuickTest built-in keywords and your own user-defined keywords. Additionally, because it is possible to add objects to the object repository before they exist in an application, it is possible to begin preparing your automated keyword-driven tests even before a software build containing the new objects is available.

One or a few automation experts usually develop the test automation infrastructure that all tests related to a certain application or functionality can use. The automation infrastructure usually includes one or more shared object repositories and one or more function libraries.

The information in the sections below provides guidance on the main tasks involved in creating these resources and describes where you can find detailed documentation for these tasks.

Analyzing Your Application

In this step, you analyze your application to determine your testing needs. This step is divided into multiple tasks:

- **Determine the development environments that QuickTest needs to support.** From the perspective of QuickTest, your application comprises windows containing a hierarchy of objects that were created in one or more development environments. QuickTest provides support for these environments using add-ins.

  You load QuickTest add-ins when QuickTest opens by using the Add-in Manager dialog box. You can check which add-ins are loaded by choosing **Help** > **About QuickTest Professional**. For more information, see the *HP QuickTest Professional Add-ins Guide*.

- **Prepare the information that QuickTest needs to identify objects in your application and to (optionally) open your application at the beginning of a run session.** You need to know the URL, the executable file name and path, or other command-line information. Later, you will enter this in Record and Run Settings dialog box. For more information, see the sections describing the Record and Run options for your testing environment in the *HP QuickTest Professional Add-ins Guide*.

- **Analyze the various business processes that customers perform while using your application to determine the actions you need to create.** You create an action for each sub-process, or task, a customer might perform.

  Navigate through your application from a customer's perspective and perform the tasks that customers might perform. Each process you perform in your application will be represented as a test in QuickTest. You can create your tests now, or you can wait until you are ready to add steps to your tests

  As you perform a process, try to compartmentalize or "chunk" it into modular units.

Example

An application that enables users to purchase items online might contain various business processes, including registering on the site and purchasing items. Each process may require one or more tasks—you create actions based on these tasks. For example, registering on the site may be a simple process requiring only one action, whereas purchasing items may be more complex, requiring several actions, such as a Login action, a Browse action, an AddToCart action, a PurchaseItems action, and a Logout action.

By creating separate reusable actions for each task, you can include calls to the same actions from multiple tests. For example, you may want to include a Login action in many of your tests.

You can create empty actions now to set up a skeleton infrastructure for your tests, or you can create them when you are ready to add steps to your actions. For more information, see "Working with Actions" on page 425.

You may also want to create a single test storing all actions relevant for an application. Then all other tests can call the actions stored in this central repository. This helps with test structure and maintenance.

Tip: As you plan your tests and actions, keep in mind that short tests and actions that check specific functions of the application or complete a transaction are better than long ones that perform several tasks.

Setting Up Object Repositories

In this step, you build one or more object repositories and ensure that all objects have clear names that follow any predetermined naming conventions defined by your organization.

You can create object repositories using QuickTest functionality to recognize and learn the objects in your application, or you can manually define objects. The object repository should contain all the objects that are relevant for the tests using this infrastructure.

By creating and populating shared object repositories that can be associated with multiple actions, you can use the same object repository in multiple tests. By maintaining all objects that are relevant to an area of an application within one shared object repository, and by associating that object repository with all relevant actions, changes to the application can be reflected in the object repository without the need to update tests.

Before you create a new object repository, verify whether an object repository containing the objects you are testing already exists. If not, you can create a new object repository or add objects to an existing one.

Creating shared object repositories for the test automation infrastructure can include the following tasks:

- **Change the way that QuickTest identifies specific objects, if needed.** This is particularly helpful when your application contains objects that change frequently or are created using dynamic content, for example, from a database. This task needs to be done before you create your object repository. For more information, see "Configuring Object Identification" on page 105.
- **Decide how you want to organize your object repositories.** For individual tests, you can work with the individual action's object repositories, or you can work with a common (shared) object repository that can be used with multiple tests. If you are new to testing, you may want to keep the default object repository per-action setting for tests. As you feel more comfortable with the basics of test design, you may want to take advantage of the shared object repository option.

  If you decide to work with shared object repositories, you need to determine how many shared object repository files are required for your application. You also need to determine which shared object repository will be used for each area of your application.

For more information, see "Managing Test Objects in Object Repositories" on page 135.

- **Add (learn) objects from your application.** You instruct QuickTest to learn the objects in your application according to filters that you define. For more information, see "Adding Test Objects to a Local or Shared Object Repository" on page 136.
- **If necessary, create new objects to represent objects that do not yet exist in your application.** Then update the properties and values of these objects as necessary after they exist in the application. For more information, see "Defining New Test Objects" on page 147.
- **Ensure that objects in the object repository have names that are easy for application testers to recognize and that follow any established object naming guidelines.** This helps make both test creation and maintenance easier over time.
- **Copy or move objects from one repository to another, as needed.** For more information, see Chapter 7, "Managing Object Repositories."
- **Merge objects added to local repositories by application testers into the shared object repositories of the automation infrastructure.** You can also merge two or more existing repositories. For more information, see Chapter 8, "Merging Shared Object Repositories."

Creating Function Libraries

Creating function libraries involves developing customized functions for the application you want to test. You may want to develop functions to test special application functionality that is not already supplied by the methods in the QuickTest object model. This enables you to create keywords that perform operations that are not normally available for use with a particular test object class. For example, you may need to add a worksheet to an Excel file, or to generate a text file during a run session.

It may also be useful to wrap existing methods and functions together with additional programming to create application-specific functions for testing operations or sequences that are commonly performed in your application. The functions you create will be available either as extra keywords or as replacements for built-in QuickTest keywords during the test creation stage.

By encapsulating much of the complex programming into function libraries, and by making these functions flexible enough to use in many testing scenarios (through the use of function parameters that control the way the functions behave), one or a few automation experts can prepare the keywords that many application testers (who are less technical) can include in multiple tests. This also makes it possible to update testing functionality without having to update all the tests that use the keywords.

You may perform the following tasks when creating a function library for the test automation infrastructure:

- **Determine whether you need to create any user-defined functions or whether you should associate any existing function libraries with your test.**
- **Determine which keywords are needed.**
- **Develop and document business-level keywords in function libraries using the QuickTest Function Library window.** For more information, see "Working with User-Defined Functions and Function Libraries" on page 905 and "Creating a Function Library" on page 909.
- **Create the actual functions within the function libraries.** You can do this manually, or you can use the Function Definition Generator to generate function definitions and header information. For more information, see "Using the Function Definition Generator" on page 923.
- **Optionally define functions as new or replacement methods for test objects.** For more information, see "Registering User-Defined Functions as Test Object Methods" on page 939.
- **Debug the function libraries.** For more information, see "Debugging a Function Library" on page 916.

Configuring QuickTest According to Your Testing Needs

After you set up the test automation infrastructure, you need to configure QuickTest to use this infrastructure:

- **Define your global testing preferences.** You need to specify configuration settings that affect how you create and run tests in general—these settings are not test-specific. For example, you can instruct QuickTest to record a movie of the run session under certain conditions, and to enable other HP products to run QuickTest tests (for example, if you want to run your tests from Quality Center).

  You can set global testing options using the Options dialog box (**Tools > Options**) or by inserting statements in the Expert View. For more information, see "About Setting Global Testing Options" on page 1231.

- **Determine whether you need to create any recovery scenarios, and create them, if needed.** Although not directly associated with the keyword-driven methodology, the automation experts who maintain the object repositories and function libraries also often maintain a set of recovery scenarios that all application testers can associate with their tests. Recovery scenarios instruct QuickTest how to proceed when a step fails. For more information, see "Defining and Using Recovery Scenarios" on page 1329.

- **Configure the QuickTest IDE to suit your testing preferences.** This enables you to easily access any needed panes, such as the Test Flow pane, the Resources pane, the Available Keywords pane, or the Data Table. For more information, see "QuickTest Window Layout" on page 1135.

Building Your Tests

You can create tests that are as simple or complex as needed. In general, it is best to create tests and actions that check just one or a few simple functions or complete a transaction rather than creating long tests and actions that perform several complex tasks or that perform many tasks.

You may perform the following tasks when creating tests and test steps:

- **Create new tests, if needed.** To do so, select **File** > **New** > **Test**.
- **Create the required actions.** For more information, see "Analyzing Your Application" on page 342.
- **Insert calls to the relevant actions.** For example, if the first task performed in a test logs in to the application, and you already created a Login action, insert a call to that action to include it in your test. For more information, see "Inserting Calls to Existing Actions" on page 464.
- **Associate your object repositories with the relevant actions.** This enables you to insert steps that perform operations on those objects. For more information, see "Associating Object Repositories with Actions" on page 446.
- **Associate your function libraries with the relevant tests.** This enables you to use your special keywords in any of the associated tests. For more information, see "Associating a Function Library with a Test" on page 921.
- **Optionally associate recovery scenarios with your test.** For more information, see "Associating Recovery Scenarios with Your Tests" on page 1372.

Adding Steps to Your Test Actions

When your actions are ready, you can add steps to them.

➤ **Add steps by selecting the keywords (operations) that represent the application functionality you want to test.** For more information, see "Working with the Keyword View" on page 383.

You can insert steps in the Keyword View, the Expert View, or a combination of both. You can add steps by dragging test objects from the Available Keywords pane, using the **New Step** option, using the Step Generator, entering steps manually, and so on. Make sure to fill in any missing values, as needed.

For more information, see "Adding a Standard Step to Your Test" on page 392, "Adding Other Types of Steps to Your Test" on page 407, and "Generating Statements in the Expert View or in a Function Library" on page 833.

➤ **Consider enhancing your tests by inserting checkpoint and output value steps to verify that your application is behaving as expected during a run session.**

You can insert checkpoints to check for differences in the text strings, objects, and tables in your application. For more information, see "Understanding Checkpoints" on page 495.

You can insert output value steps that retrieve values in your test and store them for use as input values at a different stage in the run session. For more information, see "Outputting Values" on page 669.

➤ **Consider data-driving your test to check how your application behaves with different data input during subsequent run sessions.** You can also data-drive your test to check how your application behaves during multiple iterations of the same action during a single run session. For more information, see "Working with Data Tables" on page 1197.

➤ **Consider increasing the power and flexibility of your test by replacing fixed values with parameters, if applicable.** When you parameterize your test, you can check how it performs the same operations with multiple sets of data, or from data stored or generated by an external source. For more information, see "Parameterizing Values" on page 625.

Note: If you have useful WinRunner assets, you may want to link to WinRunner tests and call WinRunner TSL functions from your QuickTest test. For more information, see "Working with WinRunner" on page 1517.

Running and Troubleshooting Your Tests

When your tests are ready, you run them, view the run results, and troubleshoot your tests, as needed.

- **Before you run a test, ensure that all of the required settings are configured as needed and that the required QuickTest add-ins are loaded.** Make sure that your application is open to the appropriate location for the beginning of the test, or that you instructed QuickTest to open it for you. Additionally, make sure that the Test Settings dialog box (**File** > **Settings**) and Record and Run Settings dialog box (**Automation** > **Record and Run Settings**) are configured for your test. For more information, see "Running Tests" on page 953.
- **After your test runs, view the run results.** Expand the nodes in the Test Results window to see where steps failed and to try to understand why. For more information, see "Viewing Run Session Results" on page 969.
- **Troubleshoot your test so that it runs correctly.** For example, you may need to add or modify test steps. For more information, see "Maintaining Tests" on page 1101.

Sample Implementation of the Keyword-Driven Methodology

As you have seen above, the process of creating a test is actually comprised of several steps.

This section walks you through the activities you might perform for each of these steps, if you were preparing a test suite for the Mercury Tours application, including:

- Define the Testing Environment for the Mercury Tours Application
- Analyze the Mercury Tours Application
- Plan and Create the Mercury Tours Test Action Repository
- Set Up the Object Repositories for the Mercury Tours Application
- Create the Function Libraries and Functions Required for Testing the Mercury Tours Application
- Create Tests and Test Steps for the Mercury Tours Business Processes

Mercury Tours is a Web-based demo application that simulates an online flight reservation application. You can view and experiment with this demo application at http://newtours.demoaut.com.

Define the Testing Environment for the Mercury Tours Application

Defining the testing environment includes determining which add-ins to load and the data required to activate the application.

Mercury Tours is a Web application that contains a few Java applets. Therefore, we need to ensure that the QuickTest Web and Java Add-ins are installed and loaded.

To activate the application, we need to run a URL in a Web browser. The URL is http://newtours.demoaut.com.

Analyze the Mercury Tours Application

When analyzing the application to determine which business processes we may want to test, we can consider both the existing business processes in the application as well as functionality that is planned for the upcoming release of the application.

The business processes that should be tested for the Mercury Tours application include:

- Registering on the site
- Reserving a flight
- Viewing the itinerary of a pending reservation
- Cancelling a reservation
- Updating user profile information
- *Reserving hotel rooms*
- *Renting a car*

Although the last two items above have not yet been implemented in the application we want to test, it is important to take them into account in the planning stage.

Now that we have determined the primary business processes, we should analyze each one to determine the break-down of these business processes into their reusable building-block elements (what will later become the test actions of our tests).

A logical breakdown of the above business processes could be:

- **Registering on the site**
  - Open the application
  - Go to the registration page
  - Enter the required information in the form
  - Submit the form
  - Verify that the form information is valid

- If a mandatory field did not have a value, an error message is displayed.
- If the password and confirm password values are not the same, an error message is displayed.
- If the username entered in the form already exists in the database, an error message is displayed.
- Otherwise, the successful registration page is displayed.

- **Reserving a flight**
  - Open the application
  - Sign on
  - Navigate to the Flight Finder page
  - Enter the flight details
  - Enter the service class and airline preferences
  - Click Next to navigate to the next page
  - Select the departure and return flights
  - Click Next to navigate to the next page
  - Enter the passenger details
  - Verify that the form information is valid
    - If the return date is earlier than the departure date, an error message is displayed.
    - If a mandatory field was not entered, an error message is displayed.
    - Otherwise, the flight confirmation page is displayed.

- **Viewing the itinerary of a pending reservation**
  - Open the application
  - Sign on
  - Navigate to the Itinerary page
- **Cancelling a reservation**
  - Open the application
  - Sign on
  - Navigate to the Itinerary page
  - Select the reservation to cancel
  - Click the **Cancel Checked Reservations** button
  - Verify
  - Successful cancellation
- **Updating user profile information**
  - Open the application
  - Sign on
  - ...

And so on for each of the remaining processes.

Comparing the sub-items in each of the business-processes helps to identify the reusable elements of each business process.

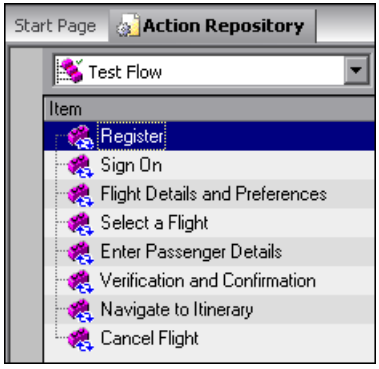
Plan and Create the Mercury Tours Test Action Repository

By analyzing the breakdown performed in the previous step, we are able to identify some logical, and reusable sub-processes. Each of these is created as a reusable action.

The required actions for the set of business processes we defined could include:

- Register
- Sign On
- Flight Details and Preference
- Select a Flight
- Enter Passenger Details
- Verification and Confirmation
- Navigate to Itinerary
- Cancel Flight

Although we are not yet ready to create the actual tests or steps yet, we can go ahead and create a single test. In the test, we can already define empty test actions for each of these. This test then acts as the **action repository**, and the tests that test each of our business processes all call actions from this action repository test.



Set Up the Object Repositories for the Mercury Tours Application

Now that we know which business processes and sub-processes we want to test, we can analyze the application in detail to determine which objects are important to test and how we want to organize the objects we will learn for these tests.

We know that it is best to create manageable-sized object repositories that are organized by areas of the application.

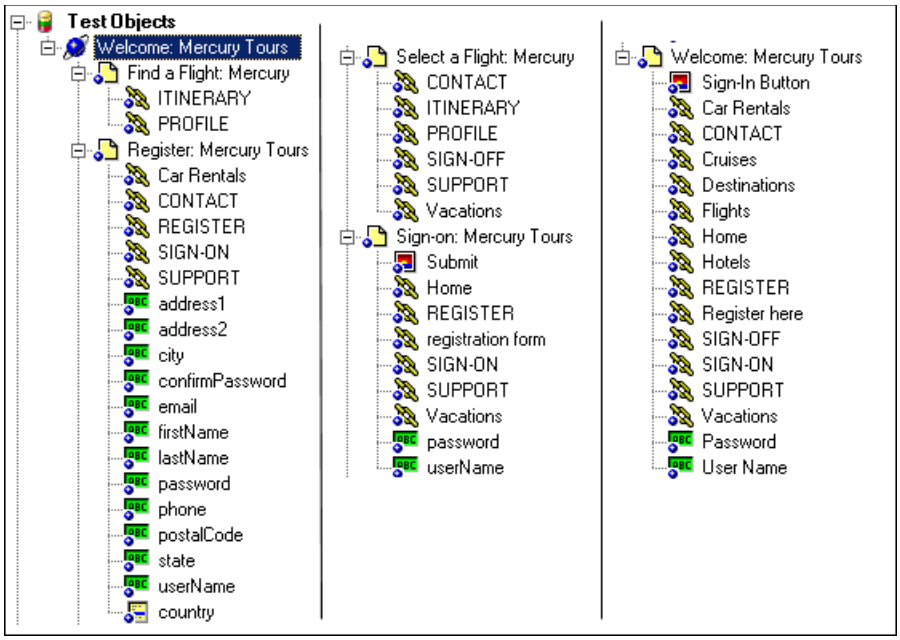
Most of the business processes we plan to test are in the central flight reservation area of the application and thus many of the same objects will be used in each of the relevant tests, but the sign on and registration processes are more standalone areas and it makes sense to store their objects separately. Thus it seems logical to create two object repository files:

- SignOn_Register
- Reservations

To create each of these repositories, we take advantage of the Navigate and Learn feature, which enables us to navigate to each page that is relevant for the object repository automatically learn all the objects in the page. By using the filter options in the Navigate and Learn feature, we can ensure that we learn only the types of objects we need. For example, we can avoid learning all the non-link image objects on every page, since these objects probably do not need to be tested and would otherwise result in a larger and less manageable object repository.

Afterwards, we should open the object repository for editing to delete specific objects that are not necessary and to rename objects that may otherwise be difficult to recognize when we later want to create steps with these objects.

Our **SignOn_Register** object repository may look something like this:



Note that each page contains only the relevant objects for the Sign on and Register business processes.

Create the Function Libraries and Functions Required for Testing the Mercury Tours Application

In some of our business processes, we want to test not only that the business processes can be performed to completion, but that certain features in the application behave as expected.

Because testing such functionality requires complex programming, and because we want to test the functionality in several different sub-processes, it makes sense to create these functionality checks in the form of functions, and to store them in function libraries, so that we can call the functions from more than one test action.

For example, we want to verify that the Mercury Tours application properly handles various invalid data in forms and we want to verify that the application properly calculates ticket prices for various types of itineraries.

We also want to make sure that we have ways to recover from certain application problems so that if such a problem occurs while a step is running, it does not prevent the action or test from completing its run or prevent other tests from running afterwards. This recovery function can be used by recovery scenarios that we will associate with our tests at later stages.

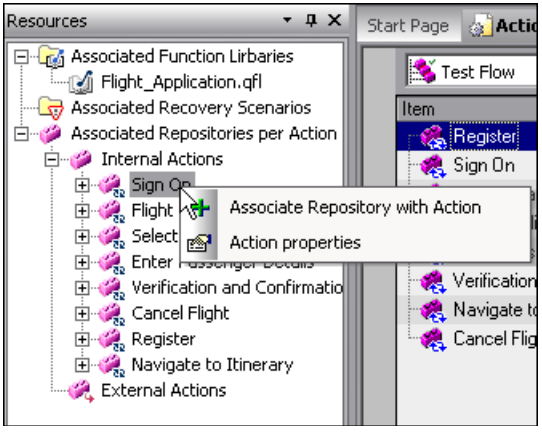
At this stage, we can create a function library containing functions such as:

- VerifyForm
- VerifyTicketPrice
- DataBaseFailureRecoveryFunction

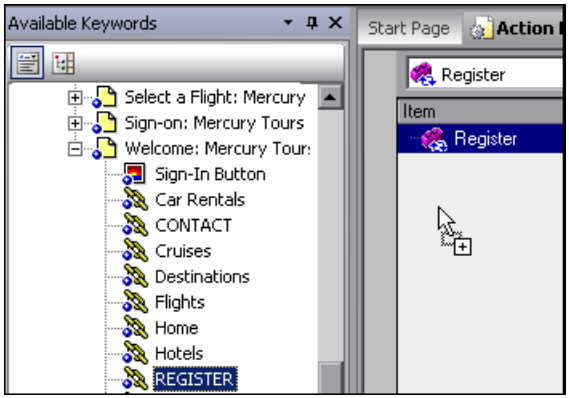
Create Tests and Test Steps for the Mercury Tours Business Processes

Now that we have planned and prepared all of the required resources for our tests, we are ready to use them to create tests and test steps that represent the steps a real user would perform on the Mercury Tours application as well as inserting functions that verify the expected functionality of various features.

We start by using the Resources pane to associate the relevant object repository with each action in the action repository test and to associate our function library with the test as well.



Then we use the Available Keywords pane to drag objects and functions into our actions to create the individual steps of each action, and add checkpoint and output value steps to verify expected behavior.



As we design our steps, we make sure to parameterize method arguments as necessary to maximize reusability of the actions in different business processes (tests).

Finally, we create new tests for each of the processes we defined in the Analyze the Mercury Tours Application step (see page 352). We use the Resources pane to associate our function library with each test and then we insert calls to the relevant actions.

13

Creating Tests Using the Recording Mechanism

You can create a test by recording the operations you perform on your application. After you create your test, you can enhance it using checkpoints and other special testing options.

Important: Before you begin recording, you need to ensure that your tests cover your testing requirements. For more information on planning your tests, see "Creating Tests — Overview" on page 309.

Tip: You can also create a test by using the keyword-driven methodology, which enables you to select keywords to indicate the operations you want to perform on your application, as described in "Creating Tests Using the Keyword-Driven Methodology" on page 335.

This chapter includes:

About Recording Tests

You record your tests while navigating through your application. As you navigate, QuickTest graphically displays each step you perform as a row in the Keyword View and a line in the Expert View. A step is anything a user does that changes the content of a page or object in your application, for example, clicking a link or typing data in an edit box. Your test steps represent the operations you perform on your application. During a run session, QuickTest uses the recorded steps to replicate the operations you performed while recording.

While you record your test steps, QuickTest creates test objects representing the objects in your application on which you perform operations. This enables QuickTest to identify the objects in your application both while creating a test and during a run session.

Recording can be useful in the following circumstances:

- You are new to QuickTest and want to learn how QuickTest interprets the operations you perform on your application and how it converts them to QuickTest objects and built-in operations.
- You need to quickly create a test that tests the basic functionality of an application or feature, and the test does not require long-term maintenance.
- You are working with a new application or with major new features of an existing application, and you want to learn how QuickTest interacts with the application.
- You are developing functions that incorporate built-in QuickTest keywords.

After creating your initial test, you can further enhance it by adding and modifying steps in the Keyword View or Expert View.

Guidelines for Recording Tests

Consider the following when recording tests:

- If you are recording steps on a Web-based application, evaluate the types of events you need to record. If you need to record more or fewer events than QuickTest generally records by default, you can configure the events you want to record. For more information, see the section on configuring Web event recording in the *HP QuickTest Professional Add-ins Guide*.
- Consider increasing the power and flexibility of your test by replacing fixed values with parameters. When you parameterize your test, you can check how it performs the same operations with multiple sets of data, or from data stored or generated by an external source. For more information, see "Parameterizing Values" on page 625.
- Consider using actions to streamline the testing process. For more information, see "Working with Actions" on page 425.
- If you have useful WinRunner assets, you can link to WinRunner tests and call WinRunner TSL functions from your QuickTest test. For more information, see "Working with WinRunner" on page 1517.
- When you record tests, you may not notice that new objects are being added to the local object repository. This may result in many testers maintaining local object repositories with copies of the same objects. When using a keyword-driven methodology, you select the objects for your steps from the existing object repository. When you need a new object, you can add it to your local object repository temporarily, but you are also aware that you need to add it to the shared object repository for future use.
- When you record a test, QuickTest enters the correct objects, methods, and argument values for you. Therefore, it is possible to create a test with little preparation or planning.

Recording a Test

You can create the main body of a test by recording the typical processes that users perform. QuickTest records the operations you perform, displays them as steps in the Keyword View, and generates them in a script (in the Expert View).

Note that by default, each test includes a single action, but can include multiple actions. This chapter describes how to record a test with a single action. For information on why and how to work with multiple actions, see Chapter 15, "Working with Actions."

By default, QuickTest records in the normal recording mode. If you are unable to record on an object in a given environment in the standard recording mode, or if you want to record mouse clicks and keyboard input with the exact x- and y-coordinates, you may want to record on those objects using analog or low-level recording. For more information, see "Choosing the Recording Mode" on page 368.

Tip: If you have objects that behave like standard objects, but are not recognized by QuickTest, you can define your objects as virtual objects. For more information, see Chapter 47, "Learning Virtual Objects."

Consider the following when recording a test:

- Before you start to record, close all applications not required for the recording session.
- If you are recording on a Web site, determine the security zone of the site. When you record on a Web browser, the browser may prompt you with security alert dialog boxes. You may choose to disable/enable these dialog boxes.
- Decide how you want to open the application when you record and run your test. You can choose to have QuickTest open one or more specified applications, or record and run on any application that is already open. The Record and Run Settings dialog box contains tabbed pages corresponding to the add-ins loaded. For more information, see the section on setting Record and Run options in the *HP QuickTest Professional Add-ins Guide*.

- Choose how you want QuickTest to record and run your test by setting global testing options in the Options dialog box and settings specific to your test in the Test Settings dialog box. For more information, see Chapter 44, "Setting Global Testing Options" and Chapter 45, "Setting Options for Individual Tests."
- If you are recording on a Web object, you must make a change to the object's value to make QuickTest record the step. For example, to record a selection in a WebList object, you must click on the list, scroll to an entry that was not originally showing, and select it. If you want to select the item in the list that is already displayed, you must first select another item in the list (click it), then return to the originally displayed item and select it (click it).

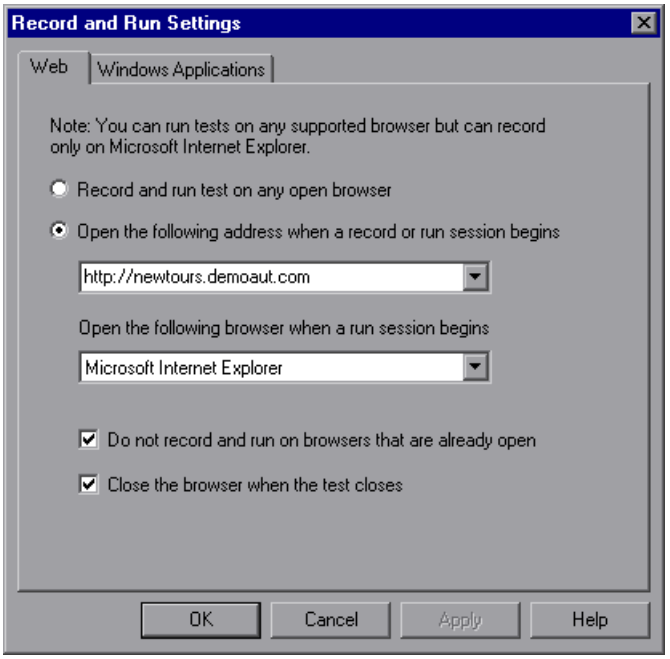
Note: If you are creating a test on Web objects, you can record your test on Microsoft Internet Explorer and run it on another supported browser (according to the guidelines specified in the *HP QuickTest Professional Readme*). QuickTest supports running tests on the following browsers—Microsoft Internet Explorer, Netscape Browser, Mozilla Firefox, and applications with embedded Web browser controls. For more information, see the *HP QuickTest Professional Add-ins Guide*.

To record a test:

1. Open QuickTest. For more information, see "Starting QuickTest" on page 20.
2. Open a test:
   - To create a new test, select **File** > **New** > **Test**, or click the down arrow next to the **New** button and select **Test**. Alternatively, click the **New** button down arrow and select **Test**.
   - To open an existing test, select **File** > **Open** > **Test** or click the down arrow next to the **Open** button and select **Test**. In the Open Test dialog box, browse to and select a test and click **Open**.

For more information, see "Managing Your Test" on page 321.

3 Click the **Record** button or select **Automation** > **Record**. If you are recording a new test and have not yet set your record and run settings in the Record and Run Settings dialog box (from **Automation** > **Record and Run Settings**), the Record and Run Settings dialog box opens.



After you set the record and run settings for a test, the Record and Run Settings dialog box will not open the next time you start a session in the same test. However, you can select **Automation** > **Record and Run Settings** to open the Record and Run Settings dialog box. You can use this option to set or modify your record and run preferences in the following scenarios:

- You have already recorded one or more steps in the test and you want to modify the settings before you continue recording.
- You want to run the test on a different application than the one you previously used.

The tabs available in the Record and Run Settings dialog box depend on the loaded add-ins.

4 Set the required options. For information on the tab to use and the options available for the environment you are testing, see the relevant add-in chapter in the *HP QuickTest Professional Add-ins Guide*.

5 To apply your changes and keep the Record and Run Settings dialog box open, click **Apply**.

6 Click **OK** to close the Record and Run Settings dialog box and begin recording your test.

7 Navigate through your application. QuickTest records each step you perform and displays it in the Keyword View and Expert View.

8 To determine if your application is functioning correctly, you can insert text checkpoints, object checkpoints, and bitmap checkpoints. For more information, see Chapter 17, "Understanding Checkpoints."

9 You can parameterize your test to check how it performs the same operations with multiple sets of data, or with data from an external source. For more information, see Chapter 24, "Parameterizing Values."

10 When you complete your recording session, 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.)

11 To save your test, click the **Save** button or select **File** > **Save**. In the Save QuickTest Test dialog box, assign a name to the test. QuickTest suggests a default folder called **Tests**. For more information, see "Saving a Test" on page 324.

Choosing the Recording Mode

Normal recording mode records the objects in your application and the operations performed on them. This mode is the default and takes full advantage of the QuickTest test object model, recognizing the objects in your application regardless of their location on the screen.

When working with specific types of objects or operations, however, you may want to choose from the following, alternative recording modes:

- **Analog Recording**. Enables you to record the exact mouse and keyboard operations you perform in relation to either the screen or the application window. In this recording mode, QuickTest records and tracks every movement of the mouse as you drag the mouse around a screen or window.

  This mode is useful for recording operations that cannot be recorded at the level of an object, for example, recording a signature produced by dragging the mouse.

Note: You cannot edit **Analog Recording** steps from within QuickTest.

- **Low Level Recording**. Enables you to record on any object in your application, whether or not QuickTest recognizes the specific object or the specific operation. This mode records at the object level and records all run-time objects as Window or WinObject test objects. Use low-level recording for recording in an environment or on an object not recognized by QuickTest. You can also use low-level recording if the exact coordinates of the object are important for your test.

Note: Steps recorded using **Low Level Recording** mode may not run correctly on all objects.

Guidelines for Analog and Low Level Recording

Consider the following guidelines when choosing **Analog Recording** or **Low Level Recording**:

- Use analog recording or low-level recording only when normal recording mode does not accurately record your operation.
- Analog recording and low-level recording require more disk space than normal recording mode.
- You can switch to either **Analog Recording** or **Low Level Recording** in the middle of a recording session for specific steps. After you record the necessary steps using analog recording or low-level recording, you can return to normal recording mode for the remainder of your recording session.

Analog Recording

- Use analog recording for applications in which the actual movement of the mouse is what you want to record. These can include drawing a mouse signature or working with drawing applications that create images by dragging the mouse.
- You can record in **Analog Recording** mode relative to the screen or relative to a specific window.
  - **Record relative to a specified window** if the operations you perform are on objects located within one window and that window does not move during the analog recording session. This ensures that during the run session, QuickTest will accurately identify the window location on which the analog steps were performed even if the window is in a different location when you run the analog steps. QuickTest does not record any click or mouse movement performed outside the specified window. When using this mode, QuickTest does not capture any Active Screen images.
  - **Record relative to the screen** if the window on which you are recording your analog steps moves during recording or if the operations you perform are on objects located within more than one window. This can include dragging and dropping an object from one window to another. When using this mode, QuickTest captures only the Active Screen image of the final state of the window on which you are recording.

- The steps recorded using analog recording are saved in a separate data file. This file is stored with the action in which the analog steps are recorded.
- When you record in **Analog Recording** mode, QuickTest adds to your test a **RunAnalog** statement that calls the recorded analog file. The corresponding Active Screen displays the results of the last analog step that was performed during the analog recording session.

Low Level Recording

- Use low-level recording for recording on environments or objects not supported by QuickTest.
- Use low-level recording for when you need to record the exact location of the operation on your application screen. While recording in normal mode, QuickTest performs the step on an object even if it has moved to a new location on the screen. If the location of the object is important to your test, switch to **Low Level Recording** to enable QuickTest to record the object in terms of its x- and y- coordinates on the screen. This way, the step will pass only if the object is in the correct position.
- While low-level recording, QuickTest records all parent level objects as Window test objects and all other objects as WinObject test objects. They are displayed in the Active Screen as standard Windows objects.
- Low-level recording supports the following methods for each test object:
  - WinObject test objects: Click, DblClick, Drag, Drop, Type
  - Window test objects: Click, DblClick, Drag, Drop, Type, Activate, Minimize, Restore, Maximize
- Each step recorded in **Low Level Recording** mode is shown in the Keyword View and Expert View. (Analog recording records only the one step that calls the external analog data file.)

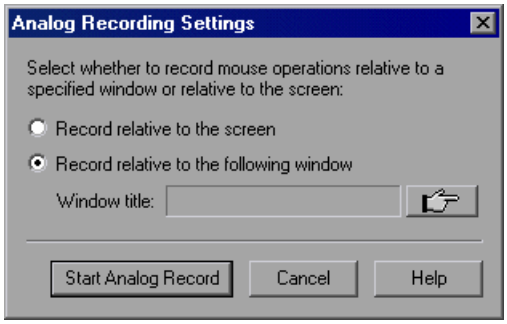
Using Analog Recording

You can switch to **Analog Recording** mode only while recording. The option is not available while editing.

To record in Analog Recording mode:

1 If you are not already recording, click the **Record** button to begin a recording session.

2 Click the **Analog Recording** button or select **Automation** > **Analog Recording**. The Analog Recording Settings dialog box opens.



3 Select from the following options:

- **Record relative to the screen**. QuickTest records any mouse movement or keyboard input relative to the coordinates of your screen, regardless of which application(s) are open or which application(s) you specified in the Record and Run Settings dialog box.

  Select **Record relative to the screen** if you perform your analog operations on objects located within more than one window or if the window itself may move while you are recording your analog operations.

Note: When you record in **Analog Recording** mode relative to the screen, the run session will fail if your screen resolution or the screen location on which you recorded your analog steps has changed from the time you recorded.

The analog tracking continues to record the movement of the mouse until the mouse reaches the QuickTest screen to turn off **Analog Recording** or to stop recording. Clicking on the QuickTest icon in the Windows taskbar is also recorded. This should not affect your test. The mouse movements and clicks on the QuickTest screen itself are not recorded.

➤ **Record relative to the following window.** QuickTest records any mouse movement or keyboard input relative to the coordinates of the specified window.

Select **Record relative to the following window** if all your operations are performed on objects within the same window and that window does not move during analog recording. This guarantees that the test will run the analog steps in the correct position within the window even if the window's screen location changes after recording.

Note: If you have selected to record in **Analog Recording** mode relative to a window, any operation performed outside the specified window is not recorded while in **Analog Recording** mode.

---

4 If you choose to **Record relative to the following window**, click the pointing hand and click anywhere in the window on which you want to record in **Analog Recording** mode. The title of the window you clicked is displayed in the window title box.

For more information on using the pointing hand feature, see "Tips for Using the Pointing Hand" on page 374.

5 Click **Start Analog Record**.

6 Perform the operations you want to record in **Analog Recording** mode.

All of your keyboard input, mouse movements, and clicks are recorded and saved in an external file. When QuickTest runs the test, the external data file is called. It tracks every movement and click of the mouse to replicate exactly the operations you recorded.

7 When you are finished and want to return to normal recording mode, click the **Analog Recording** button or select **Automation** > **Analog Recording** to turn off the option.

If you chose to **Record relative to the screen**, QuickTest inserts the RunAnalog step for a Desktop item. For example:

Item	Operation	Value
Desktop	RunAnalog	"Track1"

```
Desktop.RunAnalog "Track1"
```

If you chose to **Record relative to the following window**, QuickTest inserts the **RunAnalog** step for a Window item. For example:

Item	Operation	Value
Microsoft Internet Explorer	RunAnalog	"Track1"

```
Window("Microsoft Internet Explorer").RunAnalog "Track1"
```

The track file called by the **RunAnalog** method contains all your analog data and is stored with the current action.

You can use this track file in more than one action in your test, and also in other tests, by saving the action containing the **RunAnalog** step as a reusable action. A reusable action can be called by other tests or actions. For more information on using actions, see Chapter 15, “Working with Actions” and Chapter 16, “Working with Advanced Action Features.”

Note: When entering the RunAnalog method, you must use a valid and existing track file as the method argument.

Tip: To stop an analog step in the middle of a run session, press CTRL + ESC, then click **Stop** in the Testing toolbar.

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.

Using Low Level Recording

You can switch to **Low Level Recording** mode only while recording a test. The option is not available while editing a test.

To record in Low Level Recording mode:

1 If you are not already recording, click the **Record** button to begin a recording session.

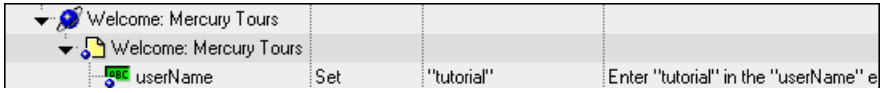
2 Click the **Low Level Recording** button or select **Automation** > **Low Level Recording**.

The record mode changes to **Low Level Recording** and all of your keyboard input and mouse clicks are recorded based on mouse coordinates. When QuickTest runs the test, the cursor retraces the recorded clicks.

3 When you are finished and want to return to normal recording mode, click the **Low Level Recording** button or select **Automation** > **Low Level Recording** to turn off the option.

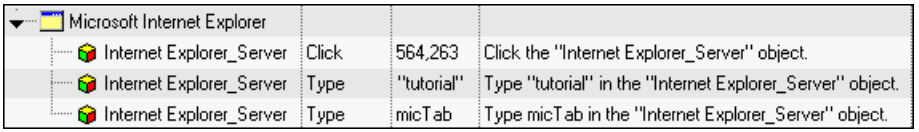
The following examples illustrate the difference between the same operations recorded using normal mode and **Low Level Recording** mode.

Suppose you type the word tutorial into a user name edit box and then press the TAB key while in normal recording mode. Your test is displayed as follows in the Keyword View and Expert View:




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

If you perform the same action while in **Low Level Recording** mode, QuickTest records the click in the user name box, followed by the keyboard input, including the TAB key. Your test is displayed as follows in the Keyword View and Expert View:



Microsoft Internet Explorer			
Internet Explorer_Server	Click	564,263	Click the "Internet Explorer_Server" object.
Internet Explorer_Server	Type	"tutorial"	Type "tutorial" in the "Internet Explorer_Server" object.
Internet Explorer_Server	Type	micTab	Type micTab in the "Internet Explorer_Server" object.

```
Window("Microsoft Internet Explorer").WinObject("Internet Explorer_Server").
    Click 564,263
Window("Microsoft Internet Explorer").WinObject("Internet Explorer_Server").
    Type "tutorial"
Window("Microsoft Internet Explorer").WinObject("Internet Explorer_Server").
    Type micTab
```

Working with the Active Screen

The Active Screen provides a snapshot of your application as it appeared when you performed the corresponding step during a recording session. An Active Screen can be captured for every step you record. Additionally, depending on the Active Screen capture options that you used while recording, the page displayed in the Active Screen can contain detailed property information on each object displayed on the page. To view the Active Screen pane, click the **Active Screen** button or select **View > Active Screen**. For information on setting Active Screen recording options, see "Enhancing Your Test" on page 315.

The Active Screen enables you to parameterize object values and insert checkpoints, methods, and output values for almost any object in the page after you finish your recording session, even if your application is not available or you do not have a step in your test corresponding to the selected object.

You can specify the level at which QuickTest captures and stores information on objects while recording tests. For example, you can instruct QuickTest to capture all properties for all test objects on the captured screen, or only the properties of the recorded objects and their parents. For more information, see "Increasing or Decreasing the Active Screen Information Saved with a Test" on page 378.

If QuickTest captured object information while recording your test, you can use the Active Screen to add these objects to the local object repository. For information on configuring the Active Screen capture settings, see "Setting Active Screen Options" on page 1240. For information on adding objects to the object repository from the Active Screen, see "Adding Test Objects to a Local or Shared Object Repository" on page 136.

When QuickTest creates an Active Screen page for a Web-based application, it stores the path to images and other resources on the page, rather than downloading and storing the images with your test. Therefore, you may need to provide login information to view password-protected resources. For information on accessing password-protected resources in the Active Screen of a Web-based application, see the section on accessing password-protected resources in the active screen in the *HP QuickTest Professional Add-ins Guide*.

When working with Web-based applications, you can specify Active Screen display criteria for captured Web pages. For example, you can specify whether QuickTest should load ActiveX controls or Java applets. For more information, see "Setting Active Screen Options" on page 1240.

Active Screen pages for non-Web-based applications are based on a single bitmap capture of the visible part of the application window (or other top-level object), with context-sensitive areas representing each object displayed in the Active Screen.

You can choose whether or not to save the content of the Active Screen with your test. Saving the content of the Active Screen with your test is especially useful if you want to be able to edit the saved test directly from the Active Screen. Later, if you need to conserve disk space after you finish editing the test, and you plan to use your test only for test runs, you can save the test without the content of the Active Screen. (Tests without Active Screen files use significantly less disk space.) For more information, see "Increasing or Decreasing the Active Screen Information Saved with a Test", below.

Increasing or Decreasing the Active Screen Information Saved with a Test

You can 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. However, more captured information also leads to slower recording and editing times. Removing or decreasing Active Screen information can be especially useful for conserving disk space after you have finished designing the test and you are using the test only for test runs.

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, you can change the amount of Active Screen information saved with your test.

To increase or decrease the Active Screen information saved with your test:

1 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. For more information, see "Setting Active Screen Options" on page 1240.

2 Perform one of the following:

- 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 step(s) containing the object(s) you want to add to the Active Screen by performing one of the following:
  - 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.
  - 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.

To stop saving Active Screen information (and reduce the disk space used by your test):

1 Open the relevant test in QuickTest.

2 Select **File** > **Save As** and clear the **Save Active Screen files** check box.

Note: If you clear this check box, your Active Screen files will not be saved, and you will not be able to edit your test using the options that are normally available from the Active Screen.

---

3 Click **Save** to apply your changes. For more information, see "Saving a Test" on page 324.

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.

Updating a Single Active Screen Capture

As the content of your application changes, you can continue to use the Active Screen from tests that you recorded previously. To do this, you update the selected Active Screen display so that you can use the Active Screen to add new steps to your test rather than re-recording steps on new or modified objects.

For example, suppose that one of the pages in your Web site now includes a new object and you want to add a checkpoint that checks this object. You can use the **Change Active Screen** command to replace the page in your Active Screen pane and then proceed to create a checkpoint for this object.

Note: It is also possible to update all Active Screen captures saved with a test using the Update Run Mode. For more information, see "Updating a Test Using the Update Run Mode Option" on page 1125.

To update a selected Active Screen capture:

1. Make sure that your application is displaying the window or page that you want to use to replace what is currently displayed in the Active Screen pane.
2. In the Keyword View, click a step that you want to change. The window or page is displayed in the Active Screen pane.
3. Select **Tools** > **Change Active Screen**. The QuickTest window is hidden and the mouse pointer becomes a pointing hand. For information about using the pointing hand feature, see "Tips for Using the Pointing Hand" on page 381.
4. Click the window or page displayed in your application.
5. When a message prompts you to change your current Active Screen display, click **Yes**.

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.

Tips for Improving Active Screen Performance

You can choose from the following Active Screen options to improve performance:

- If you are testing Windows-based 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. The less information saved, the faster your recording times will be. For more information, see "Setting Active Screen Options" on page 1240.
- If you are testing Web-based 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.
- If you are testing an application using a QuickTest add-in, see the *HP QuickTest Professional Add-ins Guide* to determine whether special Active Screen screen capture options exist for that environment.
- 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, as described in step 2 of the procedure describing how to stop saving Active Screen information on page 379. Tests without Active Screen files use significantly less disk space.

14

Working with the Keyword View

The Keyword View provides an easy way to create, view, and modify tests in a graphical easy-to-use format.

This chapter includes:

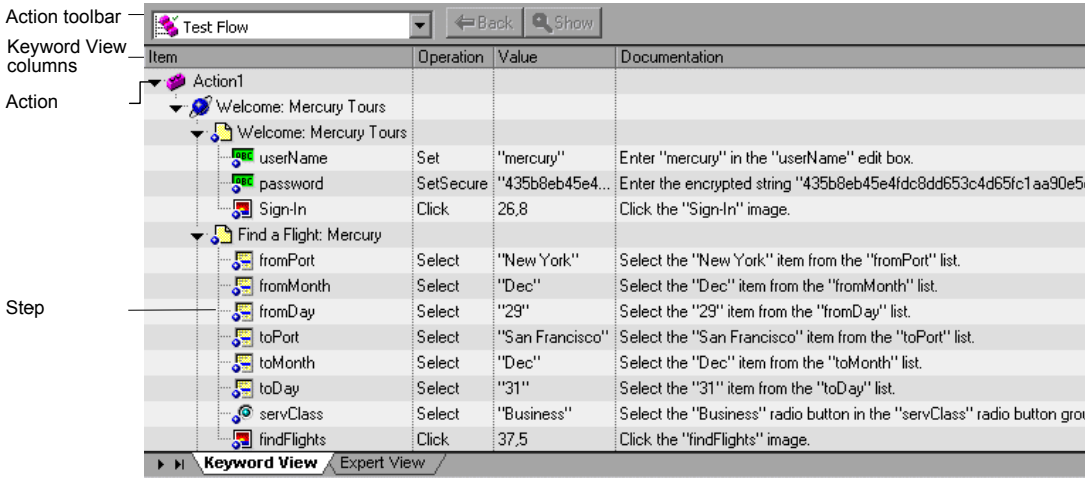
About Working with the Keyword View

The Keyword View enables you to create and view the steps of your test in a modular, table format. Each step is a row in the Keyword View that is comprised of individual, modifiable parts. You create and modify steps by selecting items and operations in the Keyword View and entering information as required. Each step is automatically documented as you complete it, enabling you to view a description of your test in understandable sentences. You can also use these descriptions as instructions for manual testing, if required.

You can use the Keyword View to add new steps to your test and to view and modify existing steps. When you add or modify a step, you select the test object or other step type you want for your step, select the method operation you want to perform, and define any necessary values for the selected operation or statement. Working in the Keyword View does not require any programming knowledge. The programming required to actually perform each test step is done automatically behind the scenes by QuickTest.

The Keyword View

The Keyword View enables you to create and view the steps of your test in a keyword-driven, modular, table format. The Keyword View is comprised of a table-like view, in which each step is a separate row in the table, and each column represents the different parts of the steps. The columns displayed vary according to your selection. For more information, see "Defining Keyword View Display Options" on page 416.

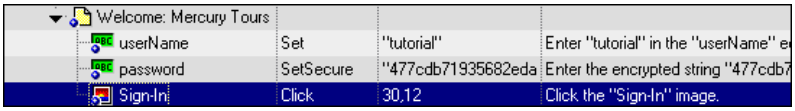


Actions are the highest level of the test hierarchy. They contain all the steps that are part of that action, and can include calls to other reusable actions. In the Keyword View, you can use the Action toolbar to view either the flow of all the top-level action calls in the test, or the content of a specific action. You can also display an action by double-clicking it in the Test Flow pane.

You can insert a new action, a call to an action, or a copy of an action, to your test. For more information on inserting and using actions in the Keyword View, see Chapter 15, "Working with Actions."

Tip: You can copy and paste or drag and drop actions to move them to a different location within a test. For more information, see "Managing Action Steps" on page 412.

Each action is comprised of steps. Each step is inserted as a row in the Keyword View. For example, the Keyword View could contain the following rows:



These rows show the following three steps that are all performed on the **Welcome: Mercury Tours** page of the Mercury Tours sample Web site:

- tutorial is entered in the **userName** edit box.
- An encrypted string is entered in the **password** edit box.
- The **Sign-In** image is clicked.
- The **Documentation** column translates each of the steps into understandable sentences.

For every step in the Keyword View, QuickTest displays a corresponding line of script in the Expert View. If you select a specific row in the Keyword View and switch to the Expert View, the cursor is located in the corresponding line of the script.

You can use the Keyword View to add steps at any point in your test. After you add steps, you can modify or delete them using standard editing commands and drag-and-drop functionality. You can print the contents of the Keyword View to your Windows default printer (and even preview the contents prior to printing). For more information, see “Printing a Test” on page 332.

In the Keyword View, you can also view properties for items such as checkpoints, output values, and actions, use conditional and loop statements, and insert breakpoints to assist you in debugging your test.

The Keyword View can contain any of the following columns: **Item**, **Operation**, **Value**, **Assignment**, **Comment**, and **Documentation**. A brief description of each column is provided below.

Item Column

The item on which you want to perform the step (test object, utility object, function call, or statement). This column displays a hierarchical icon-based tree. The highest level of the tree are actions, and all steps are contained within the relevant branch of the tree. Steps performed within the same parent object are displayed under that same object. Function calls, utility objects, and statements are placed in the tree hierarchy at the same level as the item above them (as a sibling).

You can collapse or expand an item in the item tree to change the level of detail that the tree displays.

- To collapse an item and its sub-items, click the arrow (▼) to the left of the item's icon, press the minus key (-) on your keyboard number pad, press the left arrow key on your keyboard, or right-click the item and select **Collapse Sub Tree**. The item tree hides all its sub-items and the collapse arrow changes to expand.
- To collapse all the items in the tree, select **View** > **Collapse All**.
- To expand an item one level or to its previously expanded state, select it and click the arrow (▶) to the left of the item icon, press the plus key (+) on your keyboard number pad, press the right arrow key on your keyboard, or right-click the item and select **Expand Sub Tree**. The tree displays the details for the item and all its first-level sub-items and the expand arrows change to collapse.
- To expand an item and all its sub-items, select the item and press the asterisk (*) key on your keyboard number pad. The tree displays the details for the item and all its sub-items and the expand arrows change to collapse.
- To expand all the items in the tree, select **View** > **Expand All**.

Note: When you use the +, -, and * keys to expand and collapse the Item tree, make sure that the entire row is selected (by clicking to the left of the item's icon) and that a specific column is not selected, before pressing the required key. Otherwise, the keys will not work.

Operation Column

The operation to be performed on the item. This column contains a list of all available operations (methods, functions, or properties) that can be performed on the item selected in the **Item** column, for example, **Click** and **Select**. The default operation for the item selected in the **Item** column is displayed by default.

Value Column

The argument values for the selected operation, or the content of the statement. The **Value** cell is partitioned according to the number of arguments of the selected operation.

If an argument has a predefined list of values, QuickTest provides a drop-down list of possible values. If a list of values is provided, you cannot manually type a value in this box.

Assignment Column

The assignment of a value to or from a variable. For example, **Store in cCols** would store the return value of the current step in a variable called cCols, which you could then use later in the test.

You can select either **Store in** or **Get from**, depending on whether you want to retrieve the value from a variable or store the value in a variable. A **Store in X** value in the **Assignment** column is equivalent to an **X = <*step*>** line in the Expert View. A **Get From X** value in the Assignment column is equivalent to a **<*step*> = X** line in the Expert View. For more information on storing variables, see “Storing Return Values and Action Output Parameter Values” on page 794.

Comment Column

A free text edit box for any information you want to add regarding the step. These are also displayed as inline comments in the Expert View.

Note: You can also enter a comment on a new line below the currently selected step by choosing **Insert** > **Comment**. For more information, see "Adding Comments" on page 815.

Documentation Column

Read-only auto-documentation of what the step does in an easy-to-understand sentence, for example, Click the "Sign-in" image. or Select "San Francisco" in the "toPort" list. If you want to print or view only the steps, you can choose to display only this column. For example, you may want to print or view manual testing instructions.

Tips:

- You can display only the **Documentation** column of a test by right-clicking the column header row and choosing **Documentation Only** from the displayed menu.
- You can also copy the documentation by selecting **Edit** > **Copy Documentation to Clipboard**, or right-clicking the column header row and choosing **Copy Documentation to Clipboard** from the displayed menu, and then paste it into a different application, as required.

Note: If you do not see one or more of these columns in the Keyword View, you can use the Keyword View Options dialog box to display them. For more information, see "Defining Keyword View Display Options" on page 416.

Tips for Working with the Keyword View

- You can use the left and right arrow keys to move the focus one cell to the left or right, with the following exceptions:
  - In the **Item** column, the left and right arrow keys collapse or expand the item (if possible). If not possible, the arrow keys behave as in any other column.
  - When a cell is in edit mode, for example, when modifying a value or comment, the left and right arrow keys move within the edited cell.
- When a **Value** cell is selected, press CTRL+F11 to open the Value Configuration Options dialog box.
- When the entire step is selected (by clicking to its left), use the + key (expands a specific branch), - key (collapses a specific branch), and * key (expands all branches) to expand and collapse the **Item** tree.
- When a row is selected (not a specific cell), you can type a letter to jump to the next row that starts with that letter.

Note: In addition to the above commands, you can also use QuickTest menu shortcuts. For more information, see "Performing QuickTest Commands" on page 46.

Understanding the QuickTest Object Hierarchy

The QuickTest test object hierarchy comprises one or more levels of test objects. The top level object may represent a window, dialog box, or browser type object, depending on the environment. The actual object on which you perform an operation may be learned as a top level object, a second level object, for example, Window.WinToolbar, or a third level object, for example, Browser.Page.WebButton.

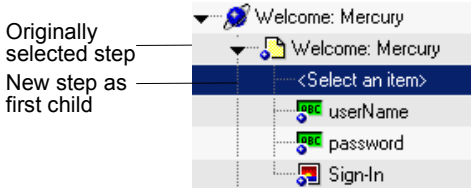
In some cases, even though the object in your application may be embedded in several levels of objects, the hierarchy does not include these objects. For example, if a WebButton object in your application is actually contained in several nested WebTable objects, which are all contained within a Browser and Page, the learned object hierarchy is only Browser.Page.WebButton.

An object that can potentially contain a lower-level object is called a container object. All top-level objects in the object hierarchy are container objects. If a second-level object contains third-level objects according to the QuickTest object hierarchy, then that object is also considered a container object. For example, in the step Browser.Page.Edit.Set "David", Browser and Page are both container objects.

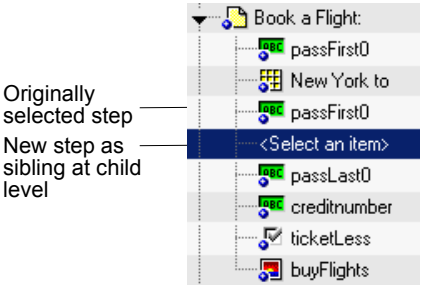
For information on the QuickTest object hierarchy for specific environments, see the relevant section in the *HP QuickTest Professional Add-ins Guide*.

When you add a step to your test in the Keyword View, the step is added as a sibling step or sub-step of the currently selected step, according to the QuickTest object hierarchy, as follows:

➤ If the selected step is a container object, the new step is inserted as the first sub-step of the container object.



➤ If the selected step is at the lowest level of the object hierarchy, the new step is inserted as a sibling step immediately after the selected step.



Adding a Standard Step to Your Test

You can use the Keyword View to add a step at any point in your test. You can add a step below the currently selected step, at the end of a test, or at the beginning of a new test. You can also add a new step immediately after a conditional or loop block, as described in "Adding a Standard Step After a Conditional or Loop Block" on page 409.

Tip: You can also add a step using the Step Generator. For more information, see "Inserting Steps Using the Step Generator" on page 777.

To add a standard step:

1 Perform one of the following:

- Click anywhere in the Keyword View (below the existing steps, if any) to add a step at the end of the test. If no steps are defined yet, this adds the first step to the test.
- Select **Insert** > **New Step** to add a new step after the existing steps (if any). If the test does not contain any steps, this adds the first step to the test.
- Select an existing step and select **Insert** > **New Step** to add a new step between existing steps. (If you select the last step, QuickTest adds a step at the end of the test.)
- Right-click an existing step and select **Insert New Step** from the context-sensitive menu.
- Drag and drop a test object from the Available Keywords pane to the Keyword or Expert view.

A new step is added to the Keyword View, either as a sibling step or a sub-step, according to the QuickTest object hierarchy, as described in "Understanding the QuickTest Object Hierarchy" on page 391.

Note: The **Select an item** list is generally expanded to display all applicable test objects, as well as the **Step Generator** and **Statement** items.

---

2 Define the step by clicking in the cell for the part of the step you want to modify and specifying its contents, as described below. Each cell in the step row represents a different part of the step. For each step, you can define the following:

- **Item.** A test object on which you perform a step. You must select an option from the **Item** column before you can add additional content to a step. For more information, see "Selecting an Item for Your Step" on page 395.
- **Operation.** The operation to be performed on the item. For more information, see "Selecting the Operation for Your Step" on page 403.

- **Value.** (If relevant.) The argument values for the selected operation. For more information, see "Defining Values for Your Step Arguments" on page 404.
- **Assignment.** (If relevant) The variable value. Double-click in the left part of the **Assignment** cell if you want to create or edit an assignment to or from a variable. Click the arrow button to select either **Get from** or **Store in**, depending on whether you want to retrieve the value from a variable or store the value in a variable. Click in the right part of the **Assignment** cell to specify or modify the name of the variable.

Note: The **Documentation** cell is read-only. This cell displays an explanation of what the step does in an easy-to-understand sentence, for example, Click the "Sign-in" image. or Select "San Francisco" in the "toPort" list. In most cases, QuickTest can generate the description displayed in this cell.

If you created a function library and associated it with the test, QuickTest can display documentation for it only if you defined the relevant text in the function library. For more information, see "Documenting the Function" on page 934 and "Working with User-Defined Functions and Function Libraries" on page 905.

Tip: You can use the standard editing commands (**Cut**, **Copy**, **Paste**, and **Delete**) in the **Edit** menu or in the context menu to make it easier to define or modify your steps. You can also drag and drop steps to move them to a different location within your action. For more information, see "Managing Action Steps" on page 412 and "Using Keyboard Commands in the Keyword View" on page 415.

---

3 After you make your changes, save the test. For more information, see "Saving a Test" on page 324.

Selecting an Item for Your Step

An item can be any of the following:

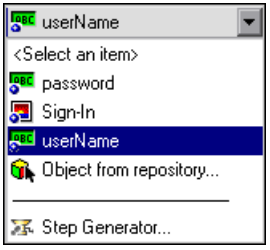
- A test object in the object repository.
  - You can either choose a test object from the list, or select **Object from repository** to open the Select Object for Step dialog box in which you can select a test object from the object repository or an object from your application. The test objects available in the list are the sibling and child test objects of the previous step's test object. The Select Object for Step dialog box contains all test objects in the object repository. You can select whether you want the operation for the step to be a test object operation or a run-time object operation. If you select a run-time object, an Object statement is added to the Keyword View.
  - You can drag and drop an object from the Available Keywords pane to your test. For more information, see "Understanding the Available Keywords Pane" on page 1165.
  - You can select an object directly from your application and add it to the object repository so that you can use it in the step.
- A statement, for example, a **Dim** statement.
- A step generated by the Step Generator. For more information, see "Inserting Steps Using the Step Generator" on page 777.

To select an item:

Click in the **Item** cell. Then click the down arrow and select the item on which you want to perform the step from the displayed list. When you insert a new step, the list is displayed automatically.

Selecting a Test Object from the Item List

The test objects available in the **Item** list are the sibling and child test objects of the previous step's test object, as defined in the shared object repository. The example below shows the objects available for the step following a **userName** test object.



To select a test object from the displayed Item list:

1. Click in the **Item** cell, then click the arrow button to display the **Item** list. If you have just created a new step, the list is displayed automatically as soon as you create the new step.
2. In the **Item** list, select the test object on which you want to perform the step. The item you select is displayed in the **Item** cell. You now need to specify an operation for the step. For more information, see "Selecting the Operation for Your Step" on page 403.

Selecting a Test Object from the Shared Object Repository

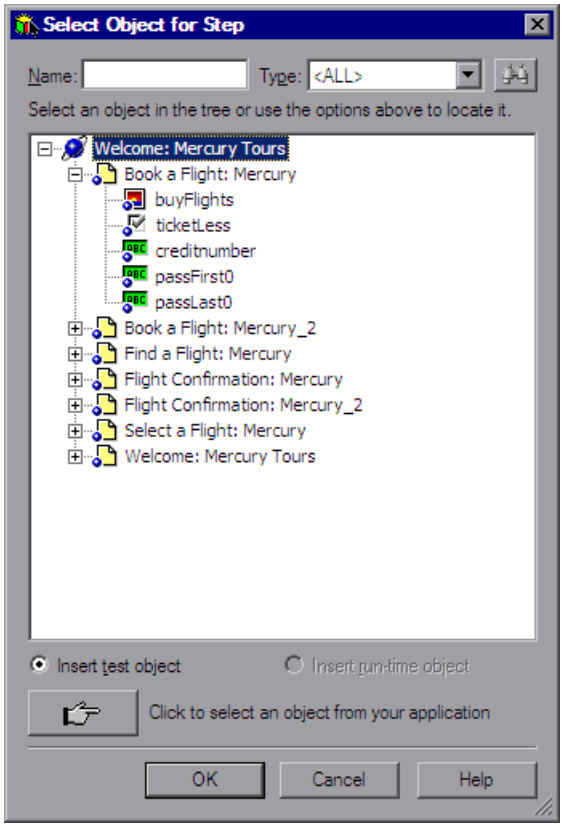
You can select any object in the object repository tree for your new step, or you can select the **Insert run-time object** option to enter an Object statement for the selected test object in your test. If the object repository is very large, you can search for the object. For example, you may want to add a password object that you know is an Edit box. You can search all the **Edit** type objects for one called password, or even one containing the letter p.

For more information on the object repository, see Chapter 5, "Managing Test Objects in Object Repositories." For more information on Object statements, see "Accessing Native Properties and Operations" on page 887.

To select a test object from the shared object repository:

1 Click in the **Item** cell, then click the arrow button to display the **Item** list. If you have just created a new step, the list is displayed automatically as soon as you create the new step.

2 In the **Item** list, select **Object from repository**. The Select Object for Step dialog box opens.



3 Select an object from the object repository tree. If the object repository is very large, you can search for the object, as described below. If a search is not required, proceed to step 8.

4 In the **Name** box, enter the name of the object, or any part of the name. For example, you can enter p to search for all object names containing the letter p.

Note: If the **Name** box is left empty, all objects of the selected object type are considered matching criteria.

---

5 In the **Type** box, select the type of object for which to search, or select **<All>** to search for the object in all the object types.

Note: The object types in this list are a generic grouping of objects according to the general object characteristics. For example, the **List** type contains list and list view objects, as well as combo boxes; the **Table** type contains both tables and grids; and the **Miscellaneous** type contains a variety of other objects, such as WebElement and WinObject.

6 Click the **Find Next** button. The search starts at the currently selected node, and the number of objects that match your criteria is displayed. The first object in the list that matches your criteria is highlighted.

7 If required, click the **Find Next** button to navigate through all the objects that match your search criteria. The search continues to the end of the tree, then wraps to the beginning of the tree, and continues.

Tip: Press F3 to find the next object that matches your search criteria, or SHIFT+F3 to find the previous match.

---

8 Click **OK**. The object is displayed in the **Item** column of the Keyword View. and is also added to the **Item** list. You can now specify the operation for the selected object. For more information, see “Selecting the Operation for Your Step” on page 403.

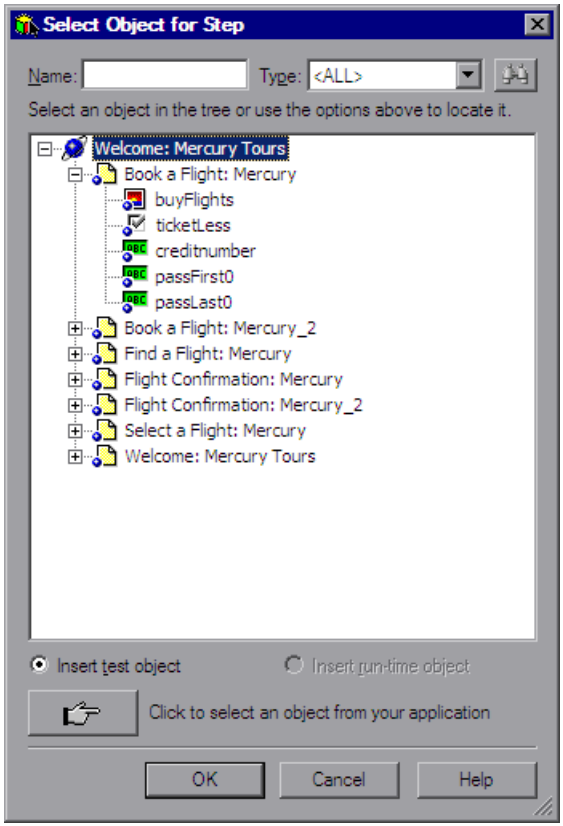
Selecting a Test Object from Your Application

If the shared object repository does not include the test object that you need for this step, you can select it directly from your application and add it to the shared object repository so that you can use it in this and other steps.

To add a test object from your application:

1 Click in the **Item** cell, then click the arrow button to display the **Item** list. If you have just created a new step, the list is displayed automatically as soon as you create the new step.

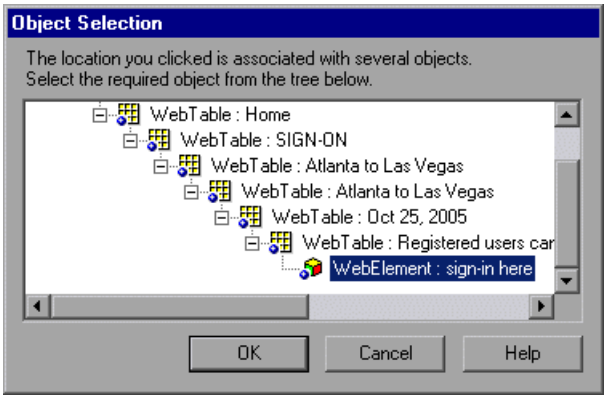
2 In the **Item** list, select **Object from repository**. The Select Object for Step dialog box opens.



3 Click the pointing hand button. QuickTest is hidden.

4 Use the pointing hand to click on the required object in your application. For more information about using the pointing hand feature, see "Tips for Using the Pointing Hand" on page 402.

If the location you clicked is associated with more than one object, the Object Selection dialog box opens.



5 Select the object for the new step and click **OK**. The object is displayed in the shared object repository tree in the Select Object for Step dialog box.

6 Click **OK**. The object is displayed in the **Item** column in the Keyword View. You can now specify the operation for the selected object. For more information, see "Selecting the Operation for Your Step" on page 403.

Tips:

- If you select an object in your application that is not in the shared object repository, a test object is added to the local object repository when you insert the new step. After you add a new test object to the local object repository, it is recommended to rename it, if its name does not clearly indicate its use. For example, you may want to rename a test object named Edit (that is used for entering a username) to UserName. This will enable other users to select the appropriate test object when adding steps using test objects located in this shared object repository.
- After you add the required objects to the local object repository, you can use the Object Repository Merge Tool to update the shared object repository and make the new objects available to other tests. For more information, see "Updating a Shared Object Repository from Local Object Repositories" on page 269.
- If you are adding a container test object, it is also recommended to specify its context, for example, if you are adding a confirmation message box from a Login page, you may want to name it Login > Confirm. For more information, see "Renaming Test Objects" on page 169.

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.

Selecting the Operation for Your Step

The **Operation** cell specifies the operation to be performed on the item listed in the **Item** column. The available operations vary according to the item selected in the **Item** column. When you select an item, all operations associated with that item are listed.

For example, if you selected a browser test object, such as a WebButton object, the list contains all of the available methods, such as Click or Exist.

To select an operation for the step:

Click in the **Operation** cell. Then click the down arrow button and select the operation to be performed on the item. The available operations vary according to the item selected in the **Item** column. For example, if you selected a browser test object, the list contains all of the methods and properties available for the browser object. If you selected a test object in the **Item** column, the default operation (most commonly-used operation) for the test object is automatically displayed in the **Operation** column. This cell is not applicable if you chose to insert a statement in the **Item** column.

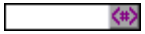
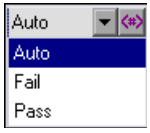
Note: Even if the **Item** column in the Keyword View is displayed to the right of the **Operation** column, you must still first select an item to view the list of available operations in the **Operation** column.

Defining Values for Your Step Arguments

The **Value** cell lists the values for each of the operation arguments. You can insert a constant value or a parameter for each argument.

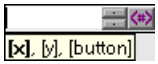
You can also encode password values. For more information, see "Inserting Encoded Passwords into Method Arguments and Data Table Cells" on page 406.

The **Value** cell is partitioned according to the number of possible arguments of the selected operation. Each partition contains different options, depending on the type of argument that can be entered in the partition, as follows:

Argument Partition	Argument Type	Instructions
	String	Enables you to enter a string containing English letters and numbers, enclosed by quotes. If you do not enter the quotes, QuickTest adds them automatically. If you modify a cell that contains a string enclosed by quotes by removing the quotes, QuickTest will not restore the quotes and the value will be treated as a variable name.
	Integer	Enables you to enter any number, or use the up and down arrows to select a number.
	Boolean	Enables you to select a **True** or **False** value from the list.
	Predefined Constant	Enables you to select a predefined value from the list. If a list of values is provided, you cannot manually type a value in this box.

To define or modify a value:

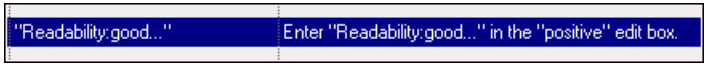
Click in each partition of the **Value** cell and enter the argument values for the selected operation. Note that when you click in the **Value** cell, a tooltip displays information for each argument. In the tooltip, the argument for the partition that is currently highlighted is displayed in bold, and any optional arguments are enclosed in square brackets.



Note: After you enter the initial value, you can edit the value at any time in the Keyword View for a test object, utility object, function call, conditional statement, or loop statement. You cannot edit the value of a regular statement, such as `x=10`, in the Keyword View after you define its initial value. You can edit the previously defined value of a regular statement only in the Expert View.

To add multi-line arguments:

Press SHIFT+ENTER to add line breaks to your argument value. After you enter a multi-line argument value, QuickTest automatically converts it to a string, and displays only the first line of the argument, followed by an ellipsis (...). This format for multi-line argument values is also displayed in the Documentation column of the Keyword View.



Tip: Select the cell to display the entire argument value to be used in the step. Note that the argument value is used during the run session exactly as it appears in the step. For example, if you enter quotation marks as part of the argument value, they are included in the argument value used during the run session. QuickTest automatically interprets a multi-line value as a string, so you do not need to add quotation marks for this purpose.

To parameterize the value for an argument:

Click the  button in the required **Value** cell. For more information, see "Parameterizing Values" on page 625.

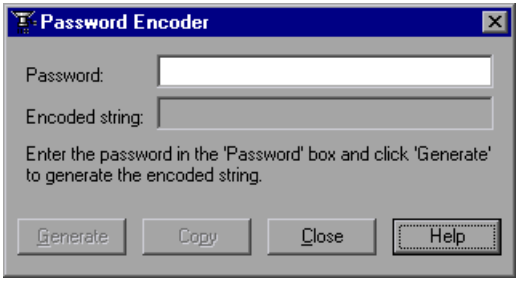
Inserting Encoded Passwords into Method Arguments and Data Table Cells

You can encode passwords to use the resulting strings as method arguments or Data Table parameter values. For example, your Web site may include a form in which the user must supply a password. You may want to test how your site responds to different passwords, but you also want to ensure the integrity of the passwords. The **Password Encoder** enables you to encode your passwords and place secure values into the Data Table.

Tip: 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.

To encode a password:

1 From the **Windows** menu, select **Start** > **Programs** > **QuickTest Professional** > **Tools** > **Password Encoder**. The Password Encoder dialog box opens.



2 Enter the password in the **Password** box.

3 Click **Generate**. The Password Encoder encrypts the password and displays it in the **Encoded String** box.

4 Use the **Copy** button to copy and paste the encoded value into the Data Table.

5 Repeat the process for each password you want to encode.

6 Click **Close** to close the Password Encoder.

Adding Other Types of Steps to Your Test

In addition to adding standard statement steps to your test using the Keyword View, you can also insert the following special types of steps using the relevant options from the **Insert** menu. Each step is entered as a row in the Keyword View, and you can then modify it as described in "Modifying the Parts of a Step" on page 410.

- You can insert a checkpoint step. For more information, see "Understanding Checkpoints" on page 495.
- You can insert an output value step. For more information, see "Outputting Values" on page 669.
- You can insert comments in steps to separate parts of an action or a test and to add details about a specific part. For more information, see "Adding Comments" on page 815.
- You can insert a step that sends information to the results, a step that puts a comment line in your test, a step that synchronizes your test with your application, or a step that measures a transaction in your test. For more information, see "Adding Steps Containing Programming Logic" on page 775.
- You can insert a step that calls a WinRunner test or function. For more information, see "Working with WinRunner" on page 1517.
- You can use conditional statements and loop statements in your test. For more information, see "Using Conditional and Loop Statements in the Keyword View" on page 408.

  For information on adding a new step immediately after a conditional or loop block, see "Adding a Standard Step After a Conditional or Loop Block" on page 409.

Using Conditional and Loop Statements in the Keyword View

Using conditional statements, you can incorporate decision making into your tests. Using loop statements, you can run a group of steps repeatedly, either while or until a condition is true. You can also use loop statements to repeat a group of steps a specific number of times. Each statement type is indicated by one of the following icons in the Keyword View:

Icon	Type
	If...Then statement
	ElseIf...Then statement
	Else statement
	While...Wend statement
	For...Next statement
	Do...While statement
	Do...Until statement

After you insert a conditional or loop statement in the Keyword View, you can insert or record steps after the statement to include them in the conditional or loop block.

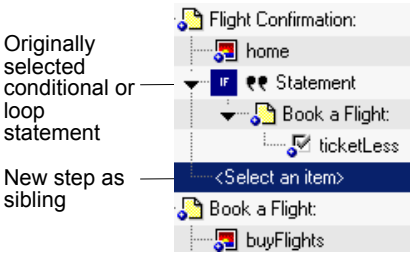
For information on including conditional and loop statements in your test, see Chapter 28, “Adding Steps Containing Programming Logic.”

Adding a Standard Step After a Conditional or Loop Block

After you add a conditional or loop statement to your test, all steps that you add or record are automatically inserted within the conditional or loop statement block. After you have finished adding steps to the block, you can add a step outside of the block, at a sibling level to the conditional or loop statement step, as described below. For more information on conditional and loop statements, see Chapter 28, "Adding Steps Containing Programming Logic."

To add a standard step after a conditional or loop block:

1 Select the conditional or loop statement step after and outside of which you want to add the new step, and select **Insert** > **New Step After Block** or press SHIFT+F8. A new step is added to the Keyword View, at the end of the conditional or loop block, outside of the conditional or loop statement (as a sibling).



2 Specify the content of the step by modifying it, as described in "Adding a Standard Step to Your Test" on page 392.

Modifying the Parts of a Step

You can modify any part of a step in the Keyword View. For example, you can change the test object on which the step is performed, change the operation to be performed in the step, or add information regarding a step in the **Comment** column.

When working in the Keyword View, you can use the standard editing commands (**Cut**, **Copy**, **Paste** and **Delete**) in the **Edit** menu or in the context menu to make it easier to modify your steps.

Tip: You can copy and paste, or drag and drop steps to move them to a different location in an action. For more information, see "Managing Action Steps" on page 412.

To modify a step, click in the cell containing the part of the step you want to modify and specify the content of the cell. Each cell in the step row represents a different part of the step. For more information, see "Adding a Standard Step to Your Test" on page 392.

Working with Comments

A **Comment** is free text entry. You can insert a comment in the **Comment** cell of a step, or you can add a comment in a separate step. Using comments can help improve readability and make a test easier to update. For example, you may want to add a comment step at the beginning of each action to specify what that section includes.

After you add a comment, it is always visible as long as one or more columns are displayed. For information on selecting columns to display, see "Defining Keyword View Display Options" on page 416. QuickTest does not process comments when it runs a test.

To add a comment to an existing step:

Select the step and type your comment in the **Comment** column.

Note: You can also insert a comment step. For more information, see "Adding Comments" on page 815.

To modify an existing comment:

Double-click the comment in the **Comment** column. The cell becomes a free text field.

Managing Action Steps

You can move an action step before or after any other step in an action. You can also delete it if it is no longer required.

Note: You can also change the run order of actions in the test from the Test Flow pane. For more information, see "Using the Test Flow Pane" on page 431.

Moving an Action Step

You can move an action to a different location within a test, as needed, and you can move a step to a different location within an action.

To move an action or a step in the Keyword View:

➤ In the **Item** column, drag the step up or down and drop it at the required location within the action. When you drag a selected step, a line is displayed, enabling you to see the location to which the step will be moved. If you drag a step within its parent object, the step is displayed in the new position under its parent. If you move the step to a different parent object, the parent is duplicated, and the step is moved below it.

To move a top-level action to a different location in the test, use the Action toolbar to display the Test Flow and then drag the action up or down to the required location.

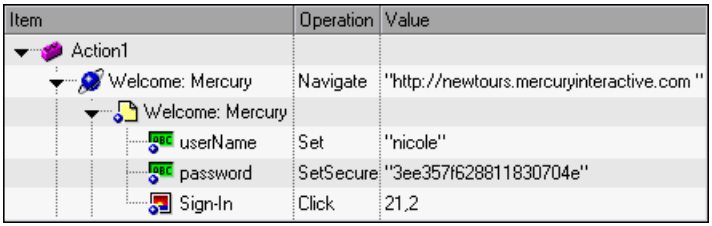
➤ Copy or cut the step to the Clipboard and then paste it in the required location. You can use **Edit** > **Copy** or CTRL + C to copy the step, **Edit** > **Cut** or CTRL + X to cut the step, and **Edit** > **Paste** or CTRL + V to paste the step. When you move, copy, or cut an action or step, you also move, copy, or cut all of its sub-steps, if any.

Notes:

- Conditional and loop blocks can only be copied or cut in their entirety. QuickTest does not enable you to copy or cut only the child nodes of conditional or loop blocks. After you copy or cut conditional or loop blocks to the Clipboard, QuickTest enables you to paste them only in valid locations.
- You cannot copy or cut a parent object together with only some of its child objects. You must either select only the parent (which automatically includes all its child objects) or the parent object together with all of its children.
- If you copy an action (**Insert** > **Call to Copy of Action**, right-click an action icon and select **Insert Call to Copy of Action**, or right-click any step and select **Action** > **Insert Call to Copy**), the Select Action dialog opens, which enables you to insert a call to a copy of an action. For more information on inserting a call to a copy of an action, see "Inserting Calls to Copies of Actions" on page 466.

Deleting an Action Step

You can delete an action step, if required. Before you delete a step, make sure that removing it will not prevent the action from running correctly. When an item has both an operation and sub-steps defined for it, as in the example below, you can choose whether to delete only the operation of the item, or to delete the item and all of its sub-steps.



Note: You cannot delete a step if one of its cells is in edit mode.

To delete a step:

1. Select the row for the item you want to delete.
2. Select **Edit** > **Delete** or press the DELETE key. One of the following messages is displayed, depending on the type of step you select:
   - ➤ If you select an item with either an operation (or checkpoint or output value) or sub-steps (but not both), a message opens asking if you want to delete the selected item and all of its sub-steps (if any).
   - ➤ If you select an item with both an operation (or checkpoint or output value) and sub-steps, a message opens asking whether you want to delete the selected item and all of its sub-steps, or delete only the item's operation (and leave the item and sub-steps).
3. Click **Delete Item** to delete the selected item (and any sub-steps), or click **Delete Operation** to delete only the operation for the selected item (and not delete the item).

Using Keyboard Commands in the Keyword View

If you prefer to use your keyboard, you can use the following keyboard commands to navigate within the Keyword View:

- Press F8 to add a new step below the currently selected step.
- Press SHIFT+F8 to add a new step after a conditional or loop block.
- Press F7 to use the Step Generator to add a new step below the selected step.
- The TAB and SHIFT+TAB keys move the focus left or right within a single row, unless you are in a cell that is in edit mode. If so, press ENTER to exit edit mode, and then you can use the TAB keys.
- When a cell containing a list is selected:
  - You can press SHIFT+F4 to open the list for that cell.
  - You can change the selected item by using the up and down arrow keys. In the **Item** column, the list must be open before you can use the arrow keys.
  - You can type a letter or sequence of letters to move to a value that starts with the typed letters. The typed sequence is highlighted in white.

Defining Keyword View Display Options

You can choose how you want to display the information in the Keyword View using the Keyword View Options dialog box. You can customize the display of the Keyword View columns, fonts, and colors. The options you set remain in effect for all tests in all subsequent sessions on your computer.

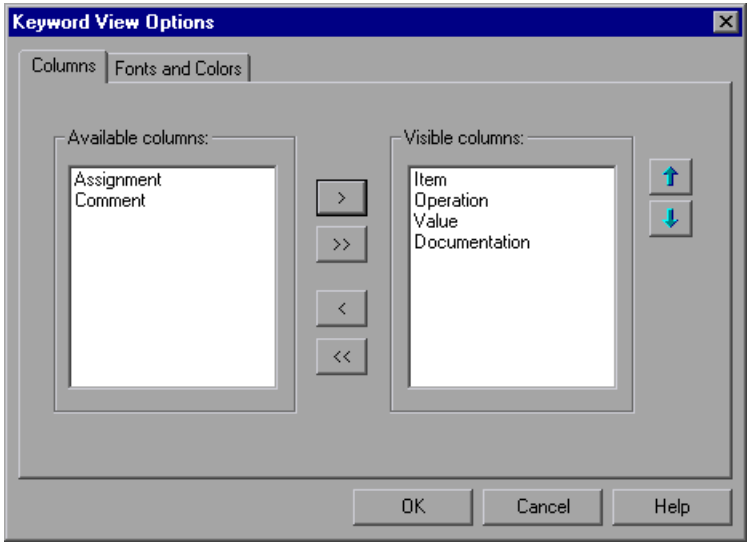
Displaying Keyword View Columns

You can use the Columns tab of the Keyword View Options dialog box to specify which columns you want to display in the Keyword View. You can also specify the order in which the columns are displayed.

Tip: You can display only the **Documentation** column by right-clicking the column header row and choosing **Documentation Only** from the displayed menu. You can then print the Keyword View for use as instructions for manual testing. For more information on printing from the Keyword View, see “Printing a Test” on page 332.

To specify the Keyword View columns to display:

1 Select **Tools** > **View Options**. The Keyword View Options dialog box opens.



The **Available columns** list shows columns not currently displayed in the Keyword View. The **Visible columns** list shows columns currently displayed in the Keyword View.

2 Double-click column names or choose column names and click the arrow buttons (> and <) to move them between the **Available columns** and **Visible columns** lists.

Tip: Click the double arrow buttons (>> and <<) to move all the column names from one list to the other. Select multiple column names (using the SHIFT and/or CONTROL keys) and click the arrow buttons (> and <) to move only the selected column names from one list to the other.

3 In the **Visible columns** list, set the order in which columns appear in the Keyword View by selecting one or more columns and then using the up and down arrow buttons.

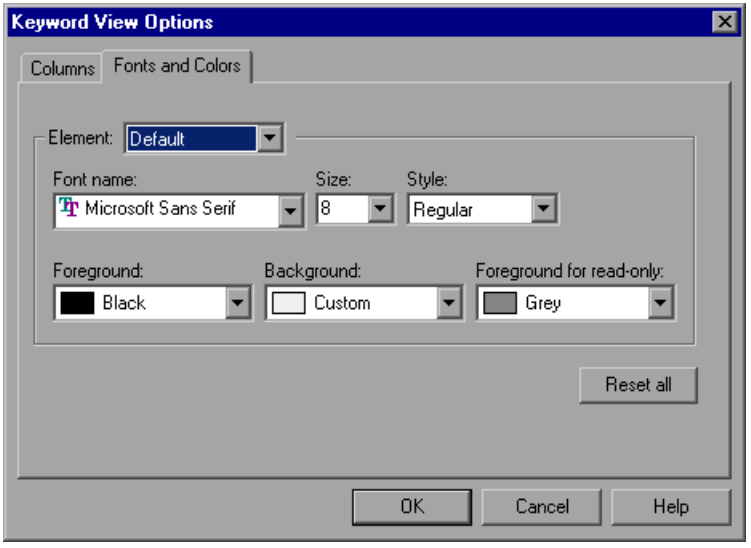
Note: The order of the columns in the Keyword View does not affect the order in which the cells need to be completed for each step. For example, if you choose to display the **Operation** column to the left of the **Item** column, you still need to select the item first, and only then is the **Operation** column list refreshed to match the selection you made in the **Item** column.

---

4 Click **OK** to close the dialog box and apply the new column display.

Setting Keyword View Fonts and Colors

You can use the Fonts and Colors tab of the Keyword View Options dialog box to specify different text and color display options for different elements in the Keyword View.



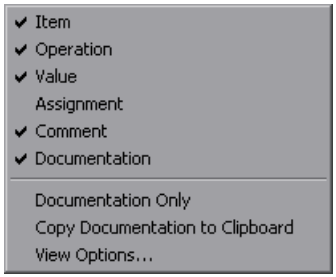
The Fonts and Colors tab includes the following options:

Option	Description
Element	You can specify different font and color options for each of these Keyword View elements. Select one of the following elements to see the current definitions and modify them: ➤ **Alternate Rows**. The background color of every other row. The font and text color for the alternate rows is the same as the font and text color defined for the **Default** element. ➤ **Comment**. The row and text of comment lines. Note that all of the available formatting options apply to entire comment rows, not to comments within a regular step row. For comments within a step row, only the specified **Foreground** color applies (all other settings are taken from the **Alternate Rows**, **Default**, or **Selected Row** settings, as appropriate). ➤ **Default**. All rows and text in the Keyword View (except for the elements listed below). ➤ **Selected Row**. The row and text currently selected (highlighted).
Font Name	Enables you to modify the font used for text in the selected element. You cannot change the font for **Alternate Rows** or **Selected Row** elements. **Note:** When testing in a Unicode environment, you must select a Unicode-compatible font. Otherwise, elements in your test may not be correctly displayed in the Keyword View. However, the test will still run in the same way, regardless of the font you choose.
Size	Enables you to modify the font size used for text in the selected element. You cannot change the font size for **Alternate Rows** or **Selected Row** elements.

Option	Description
Style	Enables you to modify the font style used for text in the selected element. You can select **Regular**, **Bold**, **Italic**, or **Underline** font styles. You cannot change the font style for **Alternate Rows** or **Selected Row** elements.
Foreground	Enables you to modify the text color for the selected element. You cannot change the foreground color for **Alternate Rows**.
Background	Enables you to modify the row color for the selected element.
Foreground for read-only	Enables you to modify the text color for rows that are read-only. This option cannot be changed for **Alternate Rows**.
Reset all	Resets all Fonts and Colors tab options to the default settings.

Tips for Working with the Keyword View

- You can display or hide specific columns by right-clicking the column header row in the Keyword View and then selecting or deselecting the required column name from the displayed menu.



  For example, you can display only the **Documentation** column if you want to print the steps for use as instructions for manual testing, by selecting **Documentation Only**.

- You can rearrange columns by dragging a column header to its new location in the Keyword View. Red arrows are displayed when the column header is dragged to an available location.

Viewing Properties of Step Elements in the Keyword View

You can view properties for different parts of a step in the Keyword View. For example, you can view object properties, action properties, action call properties, checkpoint properties, and output value properties. Right-click the item whose properties you want to view, and select the relevant option from the displayed menu.

The property options available in the **Step** menu or the context (right-click) menu change according to the currently selected step. For example, if you right-click a step that contains a checkpoint or output value on a test object, you can view object properties and checkpoint or output value properties for the current object and checkpoint or output value. If you right-click an action, you can choose to view action properties or action call properties for the current action.

Working with Breakpoints in the Keyword View

You can insert and remove breakpoints in the Keyword View. When you place a breakpoint in a step in the Keyword View, it is also displayed in the Expert View, and vice versa.

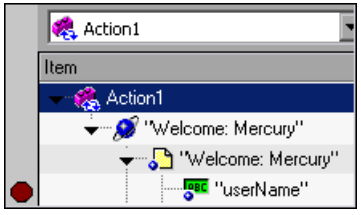
To insert a breakpoint in the Keyword View:

- Click in the left margin at the point where you want to insert the breakpoint.
- Select a step and press F9.
- Select **Debug** > **Insert/Remove Breakpoint**.

A red breakpoint icon is displayed.

To remove a breakpoint from the Keyword View:

- Click the breakpoint icon.
- Select a step and press F9.
- Select **Debug** > **Insert/Remove Breakpoint**.



Note: QuickTest automatically places the breakpoint next to the appropriate item for the step. In the example shown above, even if you click next to the **Welcome: Mercury** browser or page item, the breakpoint is automatically inserted next to the **userName** edit item, on which the step is actually performed. When you collapse items, the breakpoint icons remain in the left margin next to the closest visible item, so you can see that the test contains breakpoints.

For more information on breakpoints, see “Using Breakpoints” on page 1078.

15

Working with Actions

You can divide your test into actions to streamline the process of testing your application. This chapter covers the basic use of actions in your test. Using advanced action-related features is described in Chapter 16, "Working with Advanced Action Features."

This chapter includes:

About Working with Actions

Actions help divide your test into logical units, such as the main sections of a Web site, or specific activities that you perform in your application.

A test comprises calls to actions. When you create a new test, it contains a call to a single action. By creating tests that call multiple actions, you can design tests that are more modular and efficient.

An action consists of its own test script, including all of the steps in that action, and any objects in its local object repository.

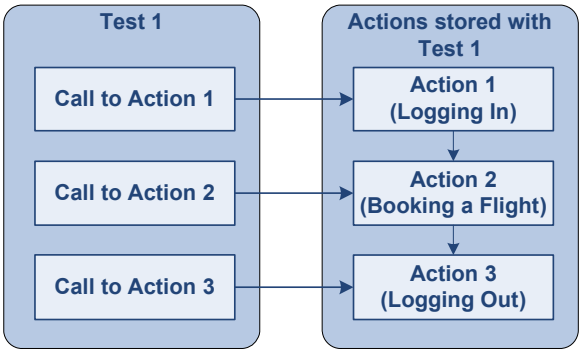
Each action is stored together with the test in which you created it. You can insert a call to an action that is stored with the test and, depending on the properties of the action, you may also be able to call an action stored with another test.

When you open a test, you can choose to view the test flow (calls to actions) or you can view and edit the individual actions stored with your test.

If you work with tests that include many steps or lines of script, it is recommended that you use actions to divide your test steps. Actions should ideally contain no more than a few dozen test steps.

For example, suppose you want to test several features of a flight reservation system. You plan several tests to test various business processes, but each one requires the same login and logout steps. You can create one action that contains the steps required for the login process, another for the logout steps, and other actions for the main steps in your test. After you create the login and logout actions, you can insert those actions into other tests.

If you create a test in which you log into the system, book one flight, and then log out of the system, your test might be structured as shown—one test calling three separate actions:



Actions enable you to parameterize and iterate over specific elements of a test. They can also make it easier to modify steps in one action when part of your application changes.

For every action called in your test, QuickTest creates a corresponding action sheet in the Data Table so that you can enter Data Table parameters that are specific to that action only. For more information on global and action data sheets, see "Using Global and Action Data Sheets" on page 429. For information on parameterizing tests, see Chapter 24, "Parameterizing Values," and Chapter 25, "Outputting Values."

Using Multiple Actions in a Test

When you create a test, it includes one action. All the steps you add and all the modifications you make while editing your test are part of a single action.

You can divide your test into multiple actions by creating new actions and inserting calls to them, by inserting calls to existing actions, or by splitting existing actions. The actions used in the test, and the order in which they are run, are displayed in the Test Flow pane.

There are three kinds of actions:

- **Reusable action.** An action that can be called multiple times by the test with which it is stored (the local test), as well as by other tests.
- **Non-reusable action.** An action that can be called only in the test with which it is stored, and can be called only once.
- **External action.** A reusable action stored with another test. External actions are read-only in the calling test, but you can choose to use a local, editable copy of the Data Table information for the external action.

For more information on creating and calling new actions, see "Creating New Actions" on page 436. For more information on inserting calls to existing actions, see "Nesting Actions" on page 453.

By default, new actions are reusable. You can mark each action you create in a test as reusable or non-reusable. Only reusable actions can be called multiple times from the current test or from another test. You can store a copy of a non-reusable action with your test and then insert a call to the copy, but you cannot directly insert a call to a non-reusable action saved with another test. Inserting calls to reusable actions makes it easier to maintain your tests, because when an object or procedure in your application changes, it needs to be updated only one time, in the original action.

Two or more tests can call the same action and one action can call another action (this is known as nesting an action, described in "Nesting Actions" on page 453). Complex tests may have many actions and may share actions with other tests.

When you run a test with multiple actions, the test results are divided by actions within each test iteration so that you can see the outcome of each action, and you can view the detailed results for each action individually. For more information on the Test Results window, see Chapter 33, "Viewing Run Session Results."

Using Global and Action Data Sheets

When you output a value to the Data Table or add a Data Table parameter to your test, you can specify whether to store the data in the **Global** data sheet or in the **action** data sheet.

- Choosing **Global sheet** enables you to create a new column or select an existing column in the **Global** sheet in the Data Table. When you run your test, QuickTest inserts or outputs a value from or to the current row of the Global data sheet during each global iteration. You can use the columns in the Global data sheet for Data Table output values or Data Table parameters in any action. This enables you to pass information between actions.
- Each action also has its own sheet in the Data Table so that you can insert data that applies only to that action. Choosing **Current action sheet (local)** enables you to create a new column or select an existing column in the corresponding action sheet in the Data Table. Note that the name of the action sheet is the same as the name of the relevant action. When you run your test, QuickTest inserts or outputs a value from or to the current row of the current action (local) data sheet during each action iteration.

  When there are parameters or output value steps in the current action's sheet, you can set QuickTest to run one or more iterations on that action before continuing with the current global iteration of the test. When you set your action call properties to run iterations on all rows, QuickTest inserts the next value from or to the corresponding action parameter or output value during each action iteration, while the values of the global parameters stay constant.

Note: If you create Data Table parameters or output value steps in your action and select to use the **Current action sheet (local)** option, be sure that the run settings for your action are set correctly in the Run tab of the Action Call Properties dialog box. You can set your action to run without iterations, to run iterations on all rows in the action's data sheet, or to run iterations only on the rows you specify. For more information on setting action iteration preferences, see "Inserting a Call to an Existing Action" on page 468.

For example, suppose you want to test how a flight reservation system handles multiple bookings. You may want to parameterize the test to check how your site responds to multiple sets of customer flight itineraries. When you plan your test, you plan the following procedures:

1. The travel agent logs into the flight reservation system.
2. The travel agent books five sets of customer flight itineraries.
3. The travel agent logs out of the flight reservation site.

When you consider these procedures, you realize that it is necessary to parameterize only the second step—the travel agent logs into the flight reservation system only once, at the beginning, and logs out of the system only once, at the end. Therefore, it is not necessary to parameterize the login and logout procedures in your test.

By creating three separate actions within your test—one for logging in, another for booking a flight, and a third for logging out—you can parameterize the second action in your test without parameterizing the others.

For more information on the Data Table, see Chapter 42, "Working with Data Tables." For more information on parameterization, see Chapter 24, "Parameterizing Values." For more information on output values, see Chapter 25, "Outputting Values."

Using the Test Flow Pane

The Test Flow pane enables you to view all the calls to actions in the current test and the order in which they are run. From the Test Flow pane, you can display test, action, and action call properties, manage actions and change their order in the test, work with the object repository, and run specific actions.

For more information, see:

- "Understanding the Test Flow Pane" on page 432
- "Working with Actions in the Test Flow Pane" on page 433

Note: The Test Flow pane is displayed by default when you start QuickTest Professional. To hide or show the pane, select **View** > **Test Flow** or click the **Test Flow Pane** toolbar button.

When you double-click an action in the Test Flow pane, the Keyword View and Expert View show only the selected action. In the Keyword View and Expert View, you can view and edit the individual steps of an action stored in this test, and view the steps for each selected external action.

- The Keyword View displays the steps of your test in a modular, table format. For more information on the Keyword View, see Chapter 14, "Working with the Keyword View."
- The Expert View displays the script for the selected action. For more information on the Expert View, see Chapter 29, "Working in the Expert View and Function Library Windows."

Understanding the Test Flow Pane

The Test Flow pane uses the following icons to indicate the different types of item in the hierarchy:

Icon	Description
	A test
	A call to a non-reusable action
	A call to an external action
	A call to a conditional, external action
	A call to a reusable action
	A call to a conditional, reusable action
	A call to a missing action (an action whose path is not saved with the test)
	A call to a conditional, missing action
	A call to a looped, reusable action
	A call to a conditional, looped, reusable action
	A call to an external, looped action
	A call to a conditional, external, looped action

Tips:

- You can right-click in the Test Flow pane title bar to view available display options and decide how to display the Test Flow pane. For example, you can auto hide the pane, dock it, or close it.
-  You can click the **Test Flow Pane** toolbar button to hide or show the Test Flow pane view.

Working with Actions in the Test Flow Pane

You can perform the following operations in the Test Flow pane:

- **Display an action in the Keyword View and Expert View.** Double-click an action in the Test Flow pane to show only that action in the Keyword View and Expert View.
- **View or hide the sub-nodes in the test.** Right-click the **Test** node in the tree and select **Expand All** or **Collapse All** to view or hide the sub-nodes in the tree. You can also select the **Test** node and press + or * on the keyboard to expand all the nodes in the test, and - to collapse the nodes.
- **Display the test properties.** Right-click the **Test** node in the tree and then select **Settings** to display the Test Settings dialog box. Details of the test and its path are displayed. For more information on the Test Settings dialog box, see "Using the Test Settings Dialog Box" on page 1262.
- **View or hide the sub-nodes of an action.** Right-click an action in the tree and then select **Expand Sub Tree** or **Collapse Sub Tree** to view or hide the sub-nodes in the action. You can also select a sub-node and press + or * on the keyboard to expand the node and - to collapse the node.
- **Display the action properties.** Right-click an action in the tree and then select **Action Properties** to display the Action Properties dialog box. The name of the action and its path are displayed. For more information on the Action Properties dialog box, see "Setting Action Properties" on page 441.
- **Display the action call properties.** Right-click an action in the tree and then select **Action Call Properties** to display the Action Call Properties dialog box. For more information on the Action Call Properties dialog box, see "Setting Action Call Properties" on page 481.
- **Work with the Object Repository.** Right-click an action in the tree and then select **Object Repository** to open the Object Repository window, which displays a tree containing all objects in the current test. For more information, see Chapter 7, "Managing Object Repositories."

- **Manage Actions.** Right-click an action in the tree and then select **Copy** or **Delete**.
  - Select **Copy** to open the Select Action dialog box and create a copy of the action in your test. For more information, see "Inserting Calls to Copies of Actions" on page 466.
  - Select **Delete** to remove the action from your test. For more information, see "Removing Actions from a Test" on page 460.
- **Run the test.** Right-click an action in the tree and then select **Run from Action** or **Run to Action** to start a run session from the beginning of the selected action, or to run the test until the beginning of the selected action and then pause the run session.
- **Debug your test.** Right-click an action in the tree and then select **Debug from Action** to begin (and pause) a debug session at the beginning of the selected action.
- **Change the run order of actions.** You can perform either of the following steps to move a top-level action (a direct child of the test) in the Test Flow Pane tree, and change the run order of the test accordingly. The action and any sub-actions are moved.
  - Right-click a top-level action in the tree and then select **Move Up** or **Move Down**. You can also press CTRL+UP arrow or CTRL+DOWN arrow to move an action and its sub-actions.
  - Drag a top-level action in the tree up or down to the required location. When you drag a selected action, a line is displayed, enabling you to see the location in the tree to which the action will be moved. You can only drag top-level actions. Selecting the parent action automatically includes all its sub-actions. You cannot drag a sub-action, nor can you drag a parent action together with only some of its sub-actions.

  For more information on moving actions, see "Managing Action Steps" on page 412.

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 Test Flow pane, and QuickTest lists the action in the Missing Resources pane.

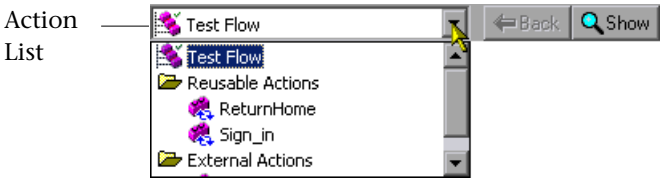
Using the Action Toolbar in the Keyword View

The Action toolbar contains options that enable you to view the top-level actions in the test flow or to view any action stored with your test (whether or not the action is actually called in the test). The Action toolbar is automatically displayed above the Keyword View when a reusable or external action is included in test.

Tip: You can display or hide the Action toolbar in the Keyword View by choosing **View** > **Toolbars** > **Action**. For more information, see Chapter 2, "QuickTest at a Glance."

In the Expert View, the Action List is always visible and the Expert View always displays the steps for the selected action. For more information on the Expert View, see Chapter 29, "Working in the Expert View and Function Library Windows."

The **Action List** enables you to view either the test flow (the calls to the top-level actions in the test) or you can view the steps for a selected reusable or external action. Selecting **Test Flow** in the Action List displays the overall flow of your test with all the calls to the top-level actions in your test. The test flow also enables you to view and edit the individual steps of non-reusable actions. An action view displays all the details of the selected reusable or external action.



In the test flow, reusable actions are not expandable. You can view the expanded steps of a reusable action by selecting the action from the Action List. For more information on reusable actions, see "Setting General Action Properties" on page 443.

There are several ways to open the action view for a reusable or external action in the Keyword View:

- In the Test Flow pane, double-click the action you want to view.
- Use the Action toolbar to display the top-level Test Flow and then double-click the call to the action you want to view.
- Use the Action toolbar to display the top-level Test Flow and then highlight the call to the action you want to view and click the **Show** button.
- Select the name of the action from the Action List.

You may have actions that are stored with your test, but are not currently called from your test. (They may be called by other tests, and you can insert calls to these actions from within your test, if needed).

Actions that are not called in your test are not displayed in the Test Flow pane, but they are displayed in the Action List. You can select these actions to view or edit their contents.

If an action is stored with your test but is not called by the test, and you are sure that you do not need the action for this test or any other test, you can delete the action from the test. For more information, see "Removing Actions from a Test" on page 460.

Creating New Actions

You can create new actions and add calls to them, as needed.

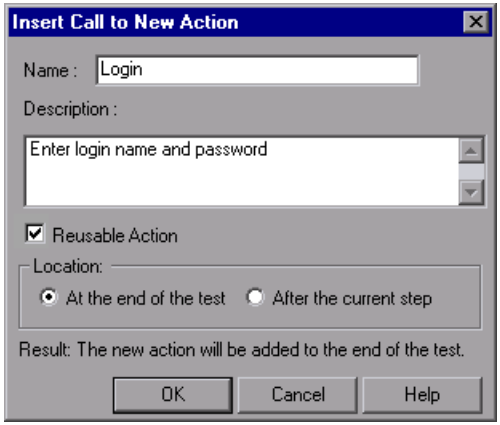
You can call the new action from your test flow as a top-level action, or you can call the new action from within another action in your test as a sub-action (or nested action). For more information, see "Nesting Actions" on page 453.

You can also split an existing action into two actions. For more information on splitting actions, see "Splitting Actions" on page 455.

To create a new action in your test:

1 If you want to insert a call to the new action from an existing action in your test, click the step after which you want to insert the new action. To insert a call to the new action from the test flow as a top-level action, click any step.

2 Select **Insert** > **Call to New Action** or click the **Insert Call to New Action** button on the **Insert** toolbar. The Insert Call to New Action dialog box opens.



3 In the **Name** box, type a new action name or accept the default name. If you rename the action, make sure that the action name is unique (within the test), does not exceed 1023 characters, does not begin or end with a space, and does not contain the following characters:
\ / : * ? " < > | % ' ! { }

4 In the **Description** box, add a description of the action. You can also add an action description at a later time using the Action Properties dialog box.

Tip: Descriptions of actions are displayed in the Select Action dialog box. The description makes it easier for you to choose an existing action you want to call. For more information, see "Setting General Action Properties" on page 443.

---

5 Ensure **Reusable Action** is selected if you want to be able to call the action from other tests or multiple times from within this test. By default, this option is selected. You can also set or modify this setting at a later time using the Action Properties dialog box.

For more information on reusable actions, see "Using Multiple Actions in a Test" on page 427. For more information on the Action Properties dialog box, see "Setting Action Properties" on page 441.

6 Decide where to insert the call to the action by selecting **At the end of the test** or **After the current step**. Choosing **At the end of the test** creates a call from the test flow to a top-level action. Choosing **After the current step** inserts the call to the action from within the current action (nests the action).

Note: If the currently selected step is a reusable action from another test, the new action is added automatically to the end of the test (the location options are disabled).

For more information on inserting action calls within actions, see "Nesting Actions" on page 453.

7 Click **OK**. A new action is stored with your test and the call to it is displayed at the bottom of the test or after the current step. You can move your action call to another location at a parallel (sibling) level within your test by dragging it to the desired location. For more information on moving actions, see "Using the Test Flow Pane" on page 431 and "Managing Action Steps" on page 412.

8 If you inserted the call to the new action while editing your test, make sure your new action is selected before adding steps to it.

Guidelines for Working with Actions

Consider the following guidelines when working with actions:

- If your action runs more than one iteration, the action must end at the same point in your application as it started, so that it can run another iteration without interruption. For example, suppose you are testing a sample flight reservation site. If the action starts with a blank flight reservation form, it should conclude with a blank flight reservation form.
- A single test may include both global Data Table parameters and action (local) Data Table parameters. For example, you can create a test in which a travel agent logs into the flight reservation system, books three flights, and logs out; the next travel agent logs into the flight reservation system, books three flights, logs out, and so forth.

  To parameterize the 'book a flight' action, you select **Current action sheet (local)** in the parameterization dialog box and enter the three flights into the relevant **Action** tab in the Data Table. To parameterize the entire test, you select **Global** in the parameterization dialog box and enter the login names and passwords for the different agents into the **Global** tab in the Data Table.

  Your entire test runs one time for each row in the Global data sheet. Within each test, each parameterized action is repeated according to the number of rows in its data sheet and the run settings selected in the Run tab of the Action Properties dialog box.
- You may want to rename the actions in your test with descriptive names to help you identify them. It is also a good idea to add detailed action descriptions. This facilitates inserting actions from one test to another. You can rename an action by choosing **Edit** > **Action** > **Rename Action**. (Make sure you follow the naming conventions for actions. For more information, see "Creating New Actions" on page 436.)

- If you plan to use an identical or virtually identical procedure in more than one test, you should consider inserting a call to an action from another test.
  - If you want to make slight modifications to the action in only one test, you should use the **Insert Call to Copy of Action** option to create a copy of the action.
  - If you want modifications to affect all tests containing the action, you should use the **Insert Call to Existing Action** option to insert a link to the action from the original test.
  - If you want modifications to the action to affect all tests containing the action, but you want to edit data in a specific test's Data Table, use the **Insert Call to Existing Action** option and, in the External tab of the Action Properties dialog box, select **Use a local, editable copy**.
- Reusable actions help you to maintain your tests, but it is important to consider the effects of including reusable actions in tests. Be sure to consider how changes to an action could potentially affect other tests that call that action.
- If you expect other users to open your tests and all actions in your tests are stored in the same drive, you should use relative paths for your reusable actions so that other users will be able to open your tests even if they have mapped their network drives differently.

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.

- If you expect certain elements of your application to change regularly, it is a good idea to divide the steps related to changeable elements into a separate action so that it will be easy to change the required steps, if necessary, after the application is modified.
- If you decide to remove an action, consider how that might affect your test or another test that contains a call to that action. For example, will it prevent a later action in the same test from running correctly? Will it cause the test containing a call to that action to fail?

➤ When you insert a call to an external action, the action is inserted in read-only format, and the **Record** button is disabled. If you want to record, you first need to insert a call to a reusable or non-reusable action into your test, or select a step from a reusable or non-reusable action that already exists in your test.

Setting Action Properties

The Action Properties dialog box enables you to define options for the stored action. These settings apply each time the action is called. 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.

The Action Properties dialog box also enables you to define input and output parameters to be used by the action, and specify the object repositories that are associated with the action. For more information, see "Setting Action Parameters" on page 472 and "Associating Object Repositories with Actions" on page 446.

Note: The following sections describe how to define action properties using the Action Properties dialog box. You can also define actions and action parameters in the Expert View. For more information, see "Understanding Action Syntax in the Expert View" on page 488.

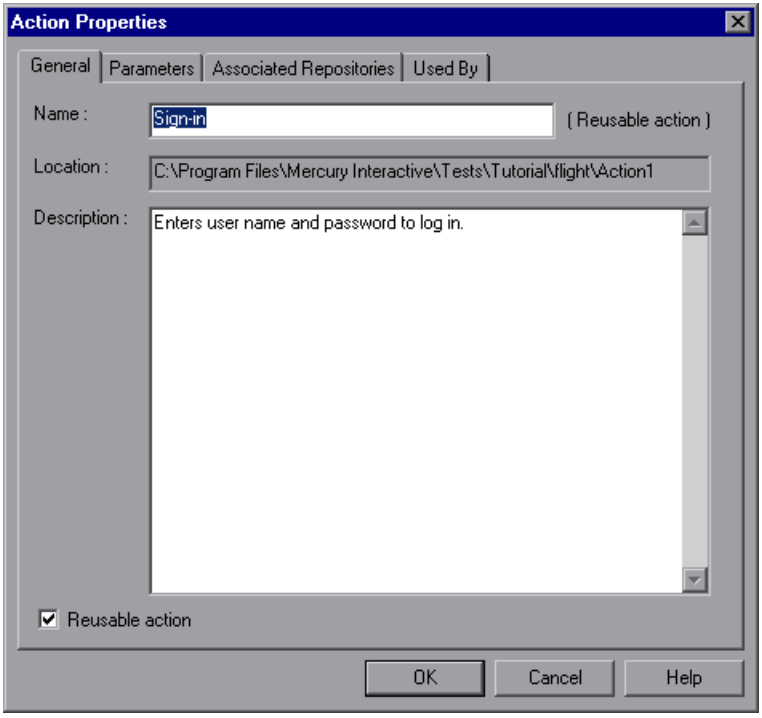
You can open the Action Properties dialog box while working with your test by:

➤ Right-clicking an action node in the Test Flow pane and selecting **Action Properties**.

➤ Choosing **Edit** > **Action** > **Action Properties** when an action node is highlighted in the Keyword View or displayed in the Expert View.

➤ Right-clicking an action node in the Keyword View and selecting **Action Properties**.

The Action Properties dialog box always contains the General tab, the Parameters tab (described in “Setting Action Parameters” on page 472), the Associated Repositories tab, and the Used By tab:



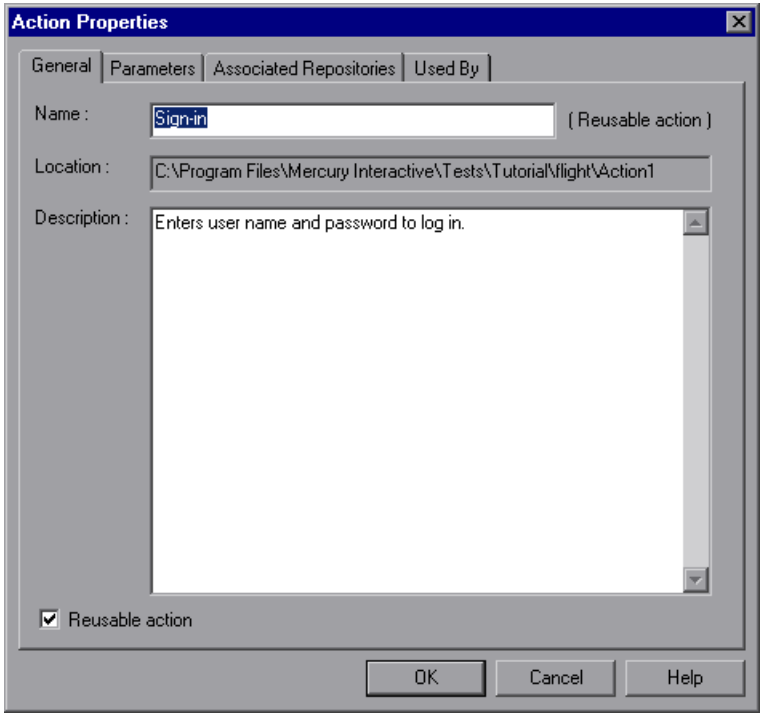
Note: In addition to the tabs shown above, the Action Properties dialog box for a called external action also contains an External Action tab, and the other tabs are read-only. For more information, see “Setting Properties for an External Action” on page 450.

For more information, see:

- "Setting General Action Properties" on page 443
- "Setting Action Parameters" on page 472
- "Associating Object Repositories with Actions" on page 446
- "Viewing a List of the Tests and Actions Using this Action" on page 452
- "Setting Properties for an External Action" on page 450

Setting General Action Properties

You can use the General tab of the Action Properties dialog box to modify the name of an action, add or edit an action's description, or change the reusability status of the action.



Note: The name of the action and its path are displayed in the tab. If it was defined with a relative path in QuickTest, then the path is displayed as <test name>\<action name>. If the action is a reusable action or an external action, then **Reusable action** or **External Action** is displayed next to the action name.

The General tab includes the following options:

Option	Description
Name	The name of the action. By default, the action name is the internal name provided by QuickTest, such as Action 1. This number is incremented by 1 for each new action that is added to the test. You can rename the action, as needed. The action name must be unique (within the test), cannot begin or end with a space, cannot exceed 1023 characters, and cannot contain the following characters: \ / : * ? " < > | % ' ! { }
Location	The folder or Quality Center path where the action is stored.
Description	You can insert comments about the action. An action description helps you and other testers know what a specific action does without reviewing all the steps in the action. The description is also displayed in the description area of the Select Action dialog box. This enables you and other testers to determine which action you want to call or copy from another test without having to open it. For more information on inserting copies and calls to actions, see "Nesting Actions" on page 453. **Note:** You can also add a description when inserting a call to a new action. For more information, see "Creating New Actions" on page 436.

Option	Description
Reusable action	Indicates whether the action is a reusable action. By default, this check box is selected. A reusable action can be called multiple times within a test and can be called from other tests. Non-reusable actions can be copied and inserted as independent actions, but cannot be inserted as calls to the original action. When you change this setting, the action icon changes to a non-reusable action icon  or reusable action icon  as appropriate. If the steps of the action were expanded, they collapse after changing a non-reusable action to a reusable action. You can view the steps of the reusable action by selecting the action name in the Test Flow pane.

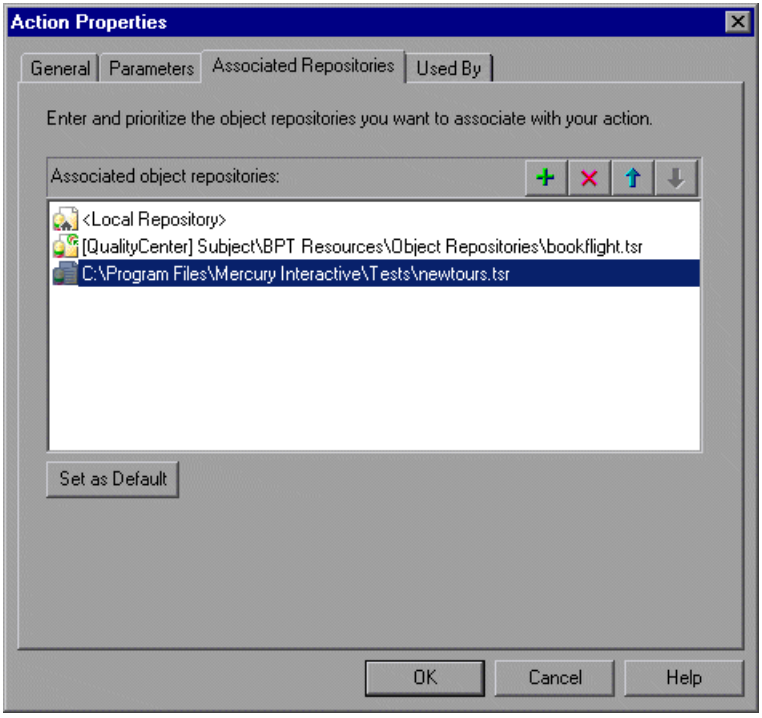
Notes:

- If the action is called more than once within the test flow or if the action is called by a reusable action, the **Reusable action** option is read-only. If you want to make the action non-reusable, remove the additional calls to the action from the test.
- You cannot expand reusable actions from the test flow view. You can view details of a reusable action by double-clicking the action in the Keyword View, or selecting the action from the Action List. For more information on the test flow and action views, see "Using the Action Toolbar in the Keyword View" on page 435.

Associating Object Repositories with Actions

You can associate object repositories with actions in several ways:

- You can associate a single action with an object repository by right-clicking the action in the Resources pane and choosing **Associate repository with action** from the context menu. This opens the Open Shared Object Repository dialog box, enabling you to associate an object repository with the selected action.
- You can use the Associated Repositories tab of the Action Properties dialog box to associate one or more object repositories with the current action. (Right-click an action in the Test Flow pane and select **Action Properties**, or select **Edit** > **Action** > **Action Properties** to open the Action Properties dialog box.)



Tip: You can associate shared object repositories with multiple actions simultaneously, using the Associate Repositories dialog box. For more information, see "Managing Shared Object Repository Associations" on page 199.

QuickTest searches these files to locate test object descriptions when identifying objects in your application. You can associate object repositories that are saved in your file system or in a Quality Center project.

Note: QuickTest uses associated object repositories from Quality Center project folders only when you are connected to the corresponding Quality Center project. If you are not connected to the relevant Quality Center project, all associated object repositories that are stored in your Quality Center project are listed as missing in the Missing Resources pane. (QuickTest always lists any associated object repository that cannot be found in the Missing Resources pane.)

In addition, if an object repository cannot be found, QuickTest displays a warning message when you click the Associated Repositories tab in the Action Properties dialog box. QuickTest also adds a question mark to the missing object repository icon  to the left of the missing object repository in the **Associated object repositories** list.

For more information on missing resources, see Chapter 41, "Handling Missing Resources."

You can associate as many object repositories as needed with an action, and the same object repository can be associated with different actions as needed. You can also set the default object repositories to be associated with all new actions in all tests.

The order of the object repositories in the list determines the order in which QuickTest searches for a test object description. If there are test objects in different object repositories with the same name, object class, and parent hierarchy, QuickTest uses the first one it finds based on the priority order defined in the Associated Repositories tab. The local object repository is always listed first and cannot be moved down the priority list or deleted.

You can enter an associated object repository as a relative path. During the run session, QuickTest searches for the file in the folders listed in the Folders pane of the Options dialog box, in the order in which the folders are listed. For more information, see "Setting Folder Testing Options" on page 1237.

Note: If you want other users or HP products to be able to run an action on other computers, and the action's associated object repositories are stored in the file system, 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 action 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.

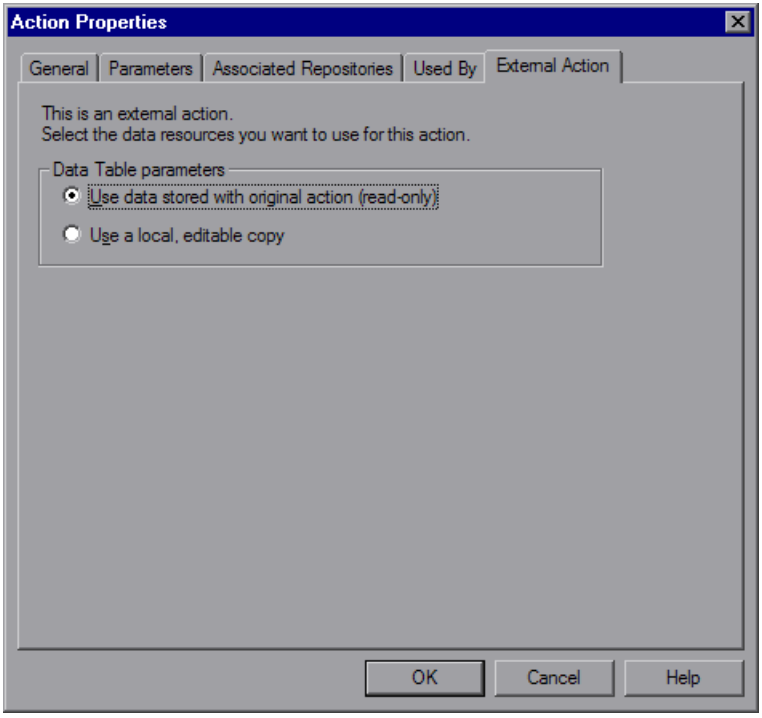
Important: If you are working with the Resources and Dependencies model with Quality Center 10.00, you should store the action's associated object repositories in the Quality Center Test Resources module and specify an absolute Quality Center path in the Folders pane. For more information, see "Considerations for Working with Relative Paths in Quality Center" on page 1450.

You can add, delete and prioritize the object repositories associated with the action using the following buttons:

Option	Description
	Associates an object repository with the action. You can enter the absolute or relative path and filename of the object repository, or use the browse button to locate the required file. You can associate object repositories that are saved in your file system or in a Quality Center project. **Note:** To use 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. **Tips:** ➤ To add a Quality Center path when connected to Quality Center, click this button. QuickTest adds [Quality Center], 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 [Quality Center], and you enter the path. You can also type the entire Quality Center path manually. If you do, you must add a space after [Quality Center]. For example: [Quality Center] Subject\ObjectRepositories\flight.tsr.
	Removes an associated object repository from the list.
	Assigns a higher priority to the selected object repository.
	Assigns a lower priority to the selected object repository.
Set as Default	Sets the current list of object repositories as the default list to be associated with all new actions. **Note:** The **Set as Default** option is enabled when the setting for this action is different from the default for all actions. **Caution:** If the default object repository is moved or renamed, QuickTest will not be able to locate it. The object repository 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."

Setting Properties for an External Action

When you insert a call to an external action, you can choose where you want QuickTest to store the Data Table data. You can specify this in the External Action tab of the Action Properties dialog box.



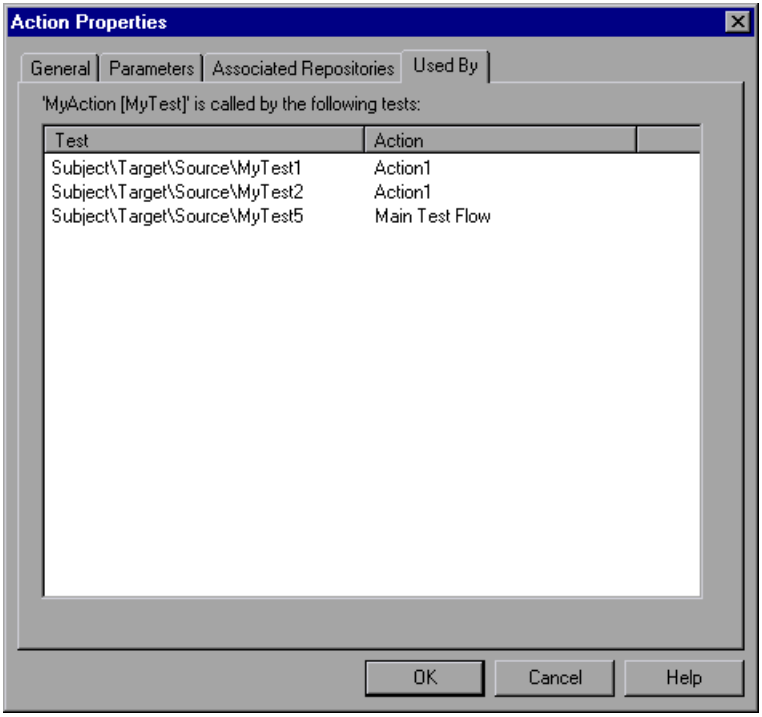
The External Action tab includes the following options:

Option	Description
Data Table parameters	Indicates where to store the action's Data Table data: ➤ To use the original action's data, select **Use data stored with the original action (read-only)**. If you select this option, the data is read-only when viewed from the calling test, and all changes to the original action's data sheet apply when the action runs in the calling test. ➤ To use an editable copy of the data in the test's Data Table, select **Use a local, editable copy**. If you select this option, a copy of the called action's data sheet is added to the calling test's Data Table and is independent of the original action. Changes to the original action's data sheet do not affect the calling test even if you insert another call to this action after the action's data sheet is modified. If the called action has parameterized steps that rely on new information in the original action's data sheet, enter the relevant column names and required data to the action sheet in the calling test manually. **Note**: When you call an external action, the global data sheet columns and data from the called action's test are always imported as a local, editable copy in the calling test's global data sheet. Changes to the original action's global data sheet do not affect the calling test even if you insert another call to this action after the called action's global data sheet is modified. If the called action has parameterized steps that rely on new information in the global data sheet, enter the relevant column names and required data to the calling test's global data sheet manually.

For more information on calls to external actions, see "Inserting a Call to an Existing Action" on page 468.

Viewing a List of the Tests and Actions Using this Action

If your tests are stored in Quality Center and are using the resources and dependencies model, the Action Properties dialog box displays the Used By tab. This enables you to view a list of the tests and actions that contain calls to this particular action. This is the same list that is displayed in the Dependencies tab of the Test Plan module in Quality Center. For more information, see “Using the Resources and Dependencies Model” on page 1447.



The Used By tab includes the following options:

Option	Description
Test	Indicates the Quality Center path of the test containing a call to this action.
Action	Indicates the internal name of the action containing a call to this action. The internal name is the name that QuickTest applies to an action by default when the action is created, for example, Action 1. The internal name of the action calling this action is displayed even if the calling action was renamed.

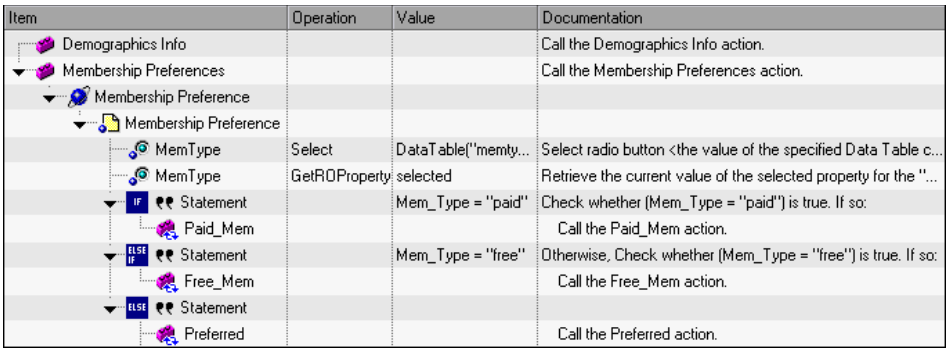
Nesting Actions

Sometimes you may want to call an action from within an action. This is called **nesting**. By nesting actions, you can:

- Maintain the modularity of your test.
- Run one or more actions based on the results of a conditional statement.

  For example, suppose you have parameterized a step where a user selects one of three membership types as part of a registration process. When the user selects a membership type, the page that opens depends on the membership type selected in the previous page. You can create one action for each type of membership. Then you can use If statements to determine which membership type was selected in a particular iteration of the test and run the appropriate action for that selection.

In the Keyword View, your test might look something like this:



Item	Operation	Value	Documentation
Demographics Info			Call the Demographics Info action.
Membership Preferences			Call the Membership Preferences action.
Membership Preference			
Membership Preference			
MemType	Select	DataTable("memty...	Select radio button <the value of the specified Data Table c...
MemType	GetROProperty	selected	Retrieve the current value of the selected property for the "...
IF Statement		Mem_Type = "paid"	Check whether (Mem_Type = "paid") is true. If so:
Paid_Mem			Call the Paid_Mem action.
ELSE IF Statement		Mem_Type = "free"	Otherwise, Check whether (Mem_Type = "free") is true. If so:
Free_Mem			Call the Free_Mem action.
ELSE Statement			
Preferred			Call the Preferred action.

In the Expert View, your test might look something like this:

```
Browser("Membership Preference").Page("Membership Preference").
    WebRadioGroup("MemType").Select DataTable("memtype", dtGlobalSheet)
Mem_Type=Browser("Membership Preference").
    Page("Membership Preference").WebRadioGroup("MemType").
        GetROProperty ("value")
If Mem_Type="paid" Then
    RunAction "Paid_Mem", oneIteration
ElseIf  Mem_Type = "free"  Then
    RunAction "Free_Mem", oneIteration
Else
    RunAction "Preferred", oneIteration
End If
```

For more information on inserting conditional statements, see “Using Conditional Statements” on page 797.

To nest an action within an existing action:

1. Highlight the step after which you would like to insert the call to the action.
2. Follow the instructions for inserting a call to a new action as described in “Creating New Actions” on page 436, or for inserting a call to a copy of an action or a call to an existing action as described in “Inserting Calls to Existing Actions” on page 464.

Splitting Actions

You can split an action that is stored with your test into two sibling actions or into parent-child nested actions. When you split an action, the second action starts with the step that is selected when you perform the split action operation.

You cannot split an action, and the option is disabled when:

- an external action is selected
- the first step of an action is selected
- you are working with a read-only test
- recording a test
- running a test

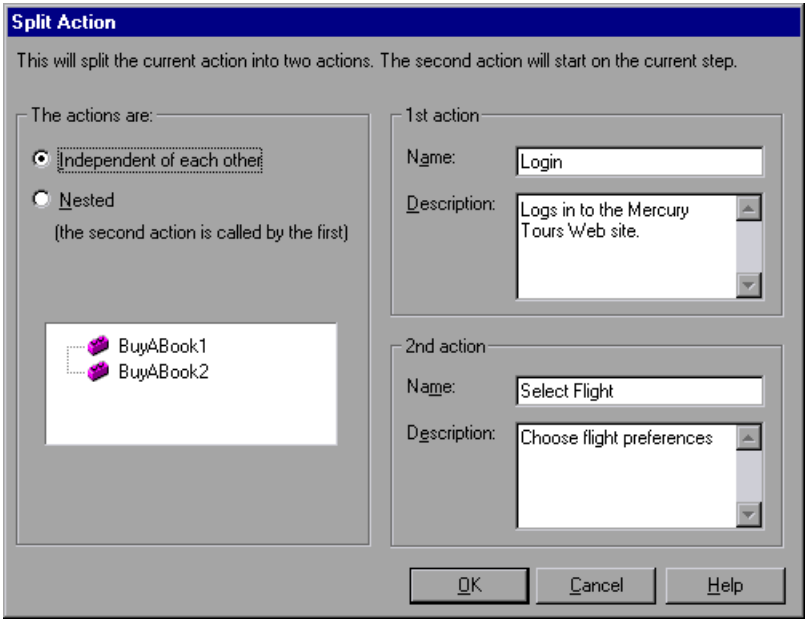
When you split an action in your test that uses a local object repository:

- QuickTest makes a copy of the local object repository.
- The two actions have identical local object repositories containing all of the objects that were in the original local object repository.
- If you add objects to one of the split actions, the new objects are added only to the corresponding local object repository.

To split an action:

1 Select the step before which you want the new (second) action to begin.

2 Select **Edit** > **Action** > **Split Action**, click the **Split Action** button, or right-click the step and select **Action** > **Split**. The Split Action dialog box opens.



3 Select one of the following options:

- **Independent of each other.** Splits the selected action into two sibling actions.
- **Nested (the second action is called by the first).** Splits the selected action into a parent action whose last step calls the second, child action.

4 If you want, modify the name and description of the two actions in the **Name** and **Description** boxes.

Note: If a reusable action is called more than once in a test and you split the action into two independent actions, each call to the action within the test will be followed by a call to the new (reusable) action. If a reusable action is called from another test, however, splitting it may cause the calling test to fail.

Renaming Actions

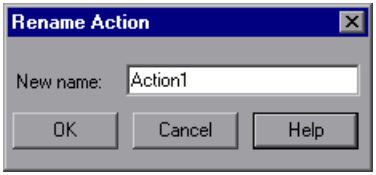
You can rename an action from the Keyword View or Expert View using the Action Properties dialog box or the Rename Action dialog box. When you rename an action, consider how it will affect your test and any tests that call this action. For example, if you rename an action that is used by another test, future run sessions may fail because the test cannot locate the specified action.

Note: If you are working with the Resources and Dependencies model, and the test containing the action you are renaming is stored the Test Plan module in Quality Center, the internal (default) action name is always displayed in the Used By tab in the Action Properties dialog box. This is true even if you rename the action. For more information, see "Viewing a List of the Tests and Actions Using this Action" on page 452.

Important: You must use the **Rename Action** option in QuickTest if you want to save an action under another name. You cannot change the name of an action directly in the file system or in Quality Center.

To rename an action in the Rename Action dialog box:

1 In the Keyword View, select the call to the action you want to rename and select **Edit** > **Action** > **Rename Action**. In the Expert View, display the action that you want to rename and select **Edit** > **Action** > **Rename Action**. The Rename Action dialog box opens.



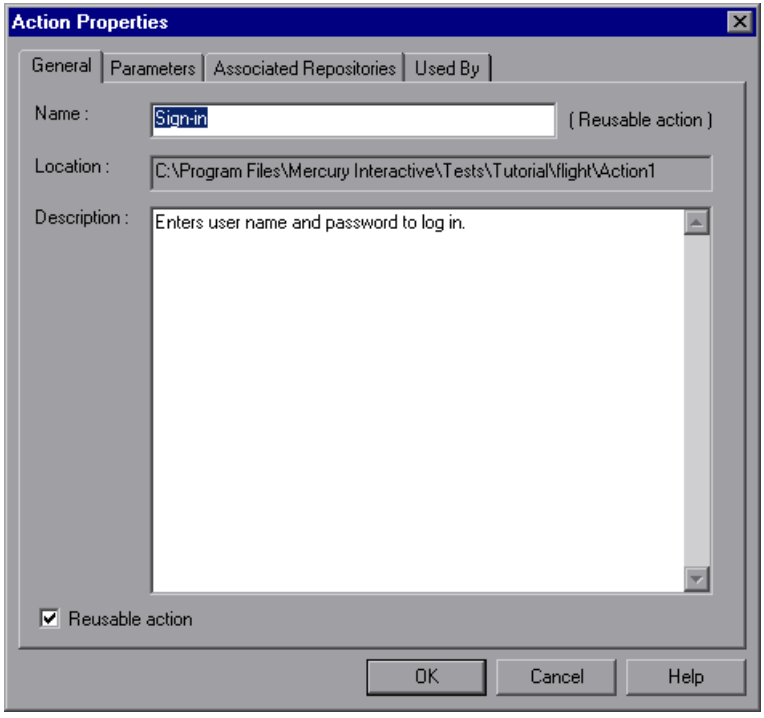
2 Enter a new name for the action in the **New name** box. Make sure that action name is unique within the test, does not begin or end with a space, does not exceed 1023 characters, and does not contain the following characters:
\ / : * ? " < > | % ' ! { }

3 Click **OK** to save the change.

Tip: You can also press SHIFT + F2 to open the Rename Action dialog box.

To rename an action in the Action Properties dialog box:

1 In the Test Flow pane or in the Keyword View, right-click the action and select **Action Properties**. Alternatively, in the Keyword View or in the Expert View, select an action and select **Edit** > **Action** > **Action Properties**. The Action Properties dialog box opens.



2 Enter a new action name in the **Name** box of the General tab. Each action name within a test must be unique. Make sure that action name is unique within the test, does not begin or end with a space, does not exceed 1023 characters, and does not contain the following characters:
\ / : * ? " < > | % ' ! { }

3 Click **OK** to save the change.

Removing Actions from a Test

If an action is no longer needed, you can remove it from your test. If the action is stored with your test (reusable or non-reusable action) and is called only once in the test, then removing the action deletes it entirely. Alternatively, if the action is stored in another test (external action), or is called more than once in this test (reusable action), removing the action deletes the selected call to the action, without affecting the source action.

The following table illustrates what happens when you delete an action:

Action Type	How deleting the action affects the test:
Reusable action (action stored in the current test)	➤ If multiple action calls exist in the current test, QuickTest removes only the call to this action. Additional calls to the action in this test remain unchanged. The corresponding action sheet in the Data Table remains unchanged. ➤ If this is only call to this action in the current test, QuickTest deletes the action in its entirety, including its corresponding action sheet in the Data Table. **Important:** Be careful when deleting a local reusable action. If the action is called by other tests, deleting the action may cause the other tests to fail.
Non-reusable action (action stored in the current test)	Deletes the action in its entirety, including its corresponding action sheet in the Data Table.
External action (action stored in a different test)	Removes the call to the action from the current test without affecting the action in the source test. The original action remains stored with the test in which it was created.

Tips for Removing Action Calls and Deleting Actions

➤ QuickTest provides several locations from which you can remove calls to actions:

Resources pane. Use to simultaneously remove all calls to a specific action.

- ➤ If you remove a reusable or non-reusable local action, QuickTest removes all calls to the action in this test and deletes the action in its entirety.
- ➤ If you remove an external action, QuickTest removes all calls to the action from the test, but does not affect the source action in any way.

Test Flow pane or the **Keyword View.** Use to remove specific calls to an action.

- ➤ If a test contains multiple calls to a single reusable action, and you remove some—but not all—of the calls, QuickTest removes the calls to the action in the specified locations, but does not delete the action itself. This means that the action can continue to be called by this test and by other tests, as needed.
- ➤ If you remove all calls to an action, the result is the same as removing the action from the Resources pane. For reusable and non-reusable actions, QuickTest removes all calls to the action in this test and deletes the action in its entirety. For external actions, QuickTest removes all calls to the action from the test, but does not affect the source action in any way.

➤ When QuickTest deletes an action in its entirety, the corresponding action sheet is removed from the Data Table, but columns related to this action that are located in the Global sheet are not removed.

➤ If you open a test containing a call to an action you removed, QuickTest informs you that the action is missing. For more information, see "Handling Missing Resources" on page 1179.

To remove a call to an action or delete an entire action:

1 In the Resources pane, the Test Flow pane, or the Keyword View:

- ➤ Right-click the action you want to remove and select **Delete**.
- ➤ Select the action you want to remove and press the **Delete** key on your keyboard.
- ➤ Select the action you want to remove and select **Edit** > **Delete**.

2 Click **Yes** in the confirmation message box.

Note: If an action stored in this test is called by other tests, deleting the action in this test may cause other tests to fail.

Creating an Action Template

If you want to include one or more statements in every new action in your test, you can create an action template. For example, if you always enter your name as the author of an action, you can add this comment line to your action template. An action template applies only to actions created on your computer.

To create an action template:

1 Create a text file containing the comments, function calls, and other statements that you want to include in your action template. The text file must be in the structure and format used in the Expert View.

2 Save the text file as **ActionTemplate.mst** in your **<QuickTest Installation Folder>\dat** folder. All new actions you create contain the script lines from the action template.

Note: Only the file name **ActionTemplate.mst** is recognized as an action template.

16

Working with Advanced Action Features

You can divide your test into actions to streamline the process of testing your application. This chapter covers the advanced use of actions in your test. Using basic action-related features is described in Chapter 15, "Working with Actions."

This chapter includes:

About Working with Advanced Action Features

Actions help divide your test into logical units, such as the main sections of a Web site, or specific activities that you perform in your application.

A test is comprised of calls to actions. When you create a new test, it contains a call to a single action. By creating tests that call multiple actions, you can design tests that are more modular and efficient.

You can pass information between actions in several ways. You can also specify input parameters for actions, so that steps in an action can use values supplied from elsewhere in the test. You can also output values from actions to be used in steps later in the test, or to be passed back to the application that ran the test. For more information, see "Using Action Parameters" on page 476.

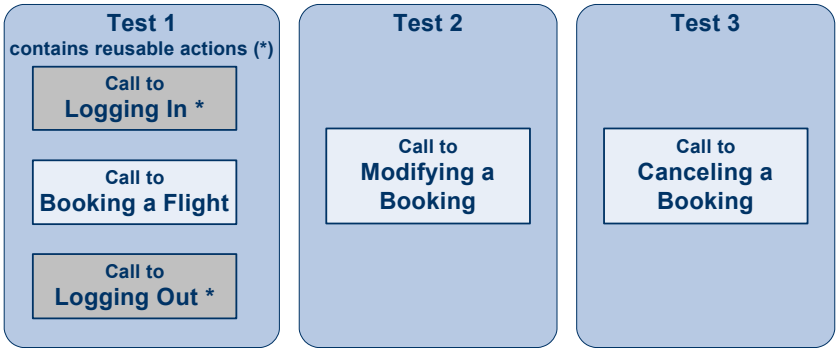
Inserting Calls to Existing Actions

When you plan a suite of tests, you may realize that each test requires some identical activities, such as logging in. Rather than inserting all of the login steps three times in three separate tests and enhancing this part of the script (with checkpoints, parameterization, and programming statements) separately for each test, you can create an action that logs into a flight reservation system and store it with one test. After you are satisfied with the action you created, you can insert calls to the existing action into other tests.

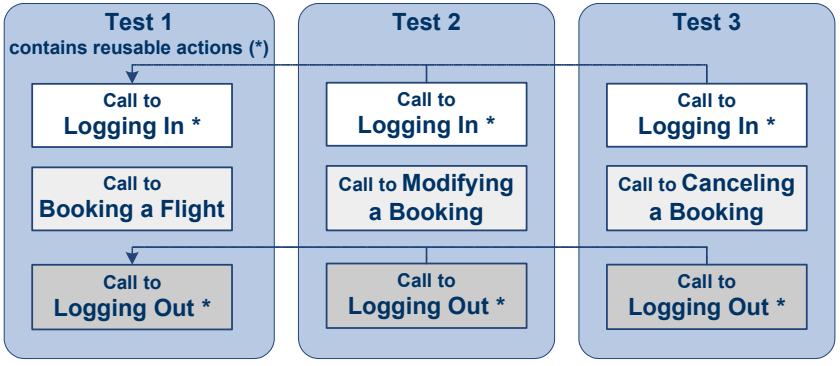
You can insert calls to an existing action by inserting a call to a copy of the action, or by inserting a call to the original action.

For example, suppose you want to create the following three tests for the Mercury Tours site—booking a flight, modifying a reservation, and deleting a reservation. While planning your tests, you realize that for each test, you need to log in and log out of the site, giving a total of five actions for all three tests.

You would initially create three tests with five actions. Test 1 would contain two reusable actions (Logging In and Logging Out). These actions can later be called by Test 2 and Test 3.



You would then finish creating Test 2 and Test 3 by inserting calls to the reusable actions you created in Test 1.



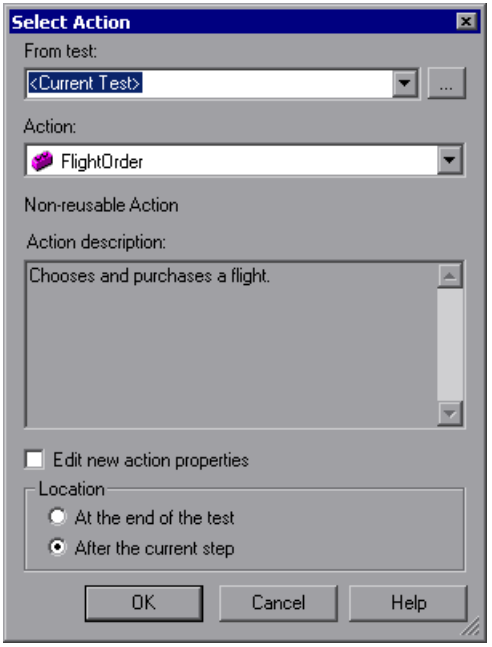
Inserting Calls to Copies of Actions

When you insert a call to a copy of an action into a test, the original action is copied in its entirety, including checkpoints, parameterization, the corresponding action tab in the Data Table, plus any defined action parameters. If the test you are copying has objects in the local object repository, the copied action's local object repository is also copied together with the action.

The action is inserted into the test as an independent, non-reusable action (even if the original action was reusable). After the action is copied into your test, you can add to, delete from, or modify the action just as you would with any other non-reusable action. Any changes you make to this action after you insert it affect only this action, and changes you make to the original action do not affect the copied action.

To create a copy of an action and call the copy in your test:

1 In your test, select **Insert** > **Call to Copy of Action**, right-click an action icon and select **Insert Call to Copy of Action**, or right-click any step and select **Action** > **Insert Call to Copy**. The Select Action dialog box opens.



2 Use the **From test** browse button to find the test containing the action you want to copy. The **Action** box displays all local actions (actions that are stored with the test you selected).

Note: You can enter a Quality Center folder or a relative path in the **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 insert. When you select an action, its type (**Non-reusable Action** or **Reusable Action**) and description, if one exists, are displayed. This helps you identify the action you want to copy. For more information on action descriptions see "Setting General Action Properties" on page 443.

4 If you want to modify the copied action's properties, select the **Edit new action properties** check box. If you select this option, the Action Properties dialog box is displayed when you click **OK**. You can then modify the action properties as described in "Setting Action Call Properties" on page 481.

Note: If you do not select this option, you can modify the action's properties later by right-clicking the action icon in the Keyword View and selecting **Action Properties**.

---

5 Decide where to insert the call to the copy of the action and select **At the end of the test** or **After the current step**.

For more information on inserting actions within actions, see "Using Action Parameters" on page 476.

Note: If the currently selected step is a reusable action from another test, the call to the copy of the action is added automatically to the end of the test (the **After the current step** option is disabled).

---

6 Click **OK**. The action is inserted into the test as a call to an independent, non-reusable action. You can move your action call to another location in your test by dragging it to the desired location. For more information on moving actions, see "Managing Action Steps" on page 412.

Inserting a Call to an Existing Action

You can insert a call to a reusable action that is stored in your current test (local action), or in any other test (external action). Inserting a call to an existing action is similar to linking to it. You can view the steps of the action in the action view, but you cannot modify them. The called action's local object repository (if it has one) is also read-only.

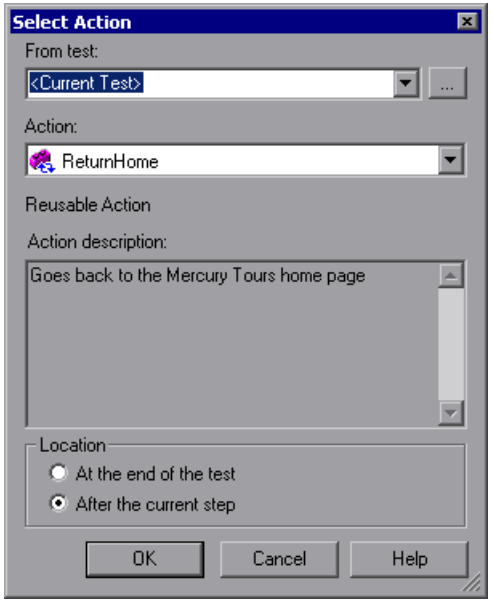
If the called external action has data in the Data Table, however, you can choose whether you want the data from the action's data sheet to be imported as a local, editable copy, or whether you want to use the (read-only) data from the original action. (Columns and data from the called action's global data sheet is always imported into the calling test as a local, editable copy.) For more information, see "Setting Properties for an External Action" on page 450.

To modify a called, external action, you must open the test with which the action is stored and make your modifications there. The modifications apply to all tests that call that action. If you chose to use the original action's data when you call an external action, then changes to the original action's data are applied as well.

Tip: You can view the location of the original action in the General tab of the Action Properties dialog box.

To insert a call to an existing action:

1 Select **Insert** > **Call to Existing Action**, right-click an action icon and select **Insert Call to Existing Action**, or right-click any step and select **Action** > **Insert Call to Existing**. The Select Action dialog box opens.



2 Use the **From test** browse button to find the test that contains the action you want to call. The **Action** box displays all reusable actions in the test you selected.

Note: You can enter a Quality Center folder or a relative path in the **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.

Tip: External actions that the test calls are also displayed in the list. If the action you want to call is already called from within the selected test, you can select it from the list of actions. This creates another call to the original action.

Note: QuickTest disables the **Action** list if the selected test does not contain any reusable or external actions.

---

4 Decide where to insert the call to the action and select **At the end of the test** or **After the current step**.

Note: If the currently selected step is a reusable action from another test, the call to the action is added automatically to the end of the test (the **After the current step** is disabled).

For more information on inserting actions within actions, see "Using Action Parameters" on page 476.

5 Click **OK**. A call to the action  is inserted into the test flow. You can move your action call to another location in your test by dragging it to the desired location. For more information on moving actions, see "Managing Action Steps" on page 412.

Tip: You can create an additional call to any reusable or external action in your test by pressing CTRL while you drag and drop the action to another location at a parallel (sibling) level within your test.

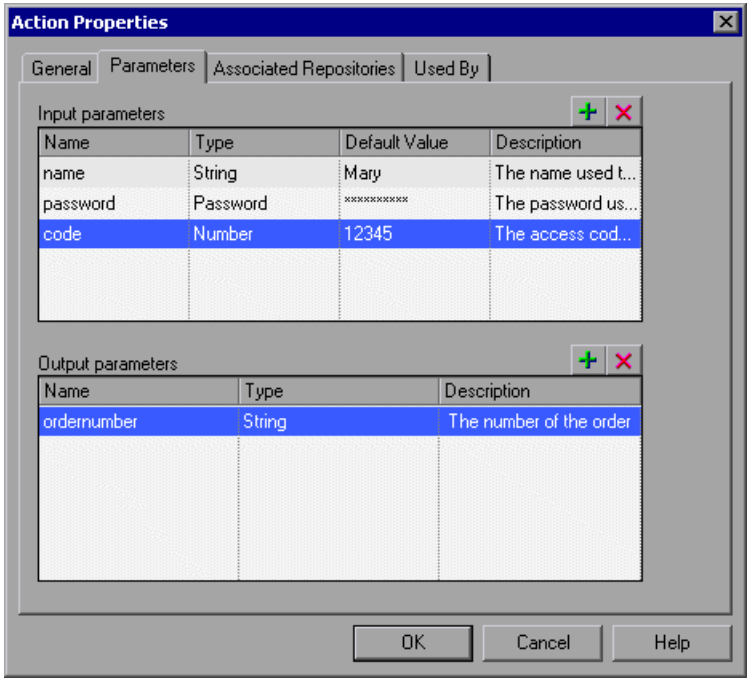
Setting Action Parameters

You can specify input parameters for an action so that steps in the action can use values supplied from elsewhere in the test. Input values for an action parameter can be retrieved from the test (for a top-level action) or from the parameters of the parent action that calls it (for a nested action), or from the output of a previous action call (for a sibling action).

You can specify output parameters for an action, so that it can return values for use later in the test. For example, you can output a parameter value to a parent action so that a later nested action can use the value.

For each input or output action parameter, you define a name (case sensitive), a type, and optionally, a description. You can also specify a default value for each action input parameter, or you can use the default value that QuickTest provides for the parameter value type that you choose. The default value is saved with the action and is used by the action if a value is not defined for a parameter in the action call. You can define, modify, and delete input and output parameters in the Parameters tab of the Action Properties dialog box (**Edit** > **Action** > **Action Properties** or right-click an action and select **Action Properties**).

For more information on using action parameters, see "Using Action Parameters" on page 476 and "Guidelines for Working with Action Parameters" on page 479.



To add a new input or output action parameter:

1. Click the **Add** button above the **Input parameters** or **Output parameters** lists to add a new parameter to the appropriate list. A row for the new parameter is added to the relevant list.
2. Click in the **Name** box and enter a name for the parameter. (Action parameter names are case sensitive.)

3 Select the value type for the parameter in the **Type** box. You can select one of the following types:

- **String.** A character string enclosed within a pair of quotation marks, for example, "New York". If you enter a value and do not include the quotation marks, QuickTest adds them automatically when the value is inserted in the script during the test run. The default value is an empty string.
- **Boolean.** A true or false value. If you select a **Boolean** value type, you can click in the **Default Value** column and click the arrow to select a **True** or **False** value. The default value is **True**.
- **Date.** A date string, for example, 3/2/2005. If you select a **Date** value type, you can click in the **Default Value** column and click the arrow to open a calendar from which you can select a date. The default value is today's date.
- **Number.** Any number. The default value is 0.
- **Password.** An encrypted password value. If you select a **Password** value type, the password characters are masked when you enter the password in the **Default Value** field. In the action, however, the value appears encrypted. The default value is an empty string, which also appears as an encrypted value in the actual action.
- **Any.** A variant value type, which accepts any of the above value types. Note that if you select the **Any** value type, you must specify the value in the format that is required in the location where you intend to use the value. For example, if you intend to use the value later as a string, you must enclose it in quotation marks. When you specify a value of **Any** type, QuickTest checks whether it is a number. If the value is not a number, QuickTest automatically encloses it in quotation marks. If you are editing an existing value, QuickTest automatically encloses it in quotation marks if the previous value had quotation marks. The default value is an empty string.

4 If you are defining an input action parameter, click in the **Default Value** box and enter a default value for the parameter. Alternatively, you can leave the default value provided by QuickTest for the parameter value type. The default value is required so that you can run the action without receiving parameter values from elsewhere in the test.

5 (Optional) Click in the **Description** box and enter a description of the parameter, for example, the purpose of the parameter in the action. QuickTest displays this description together with the name of the parameter in any dialog box in which you can choose an action parameter, including the Output Options, Parameter Options, and Value Configuration Options dialog boxes.

To modify an existing action parameter:

1 Select the parameter you want to modify from the **Input parameters** or **Output parameters** list.

2 Modify the values as necessary in the edit boxes of the parameter row.

To delete an existing action parameter:

1 Select the parameter you want to delete from the **Input parameters** or **Output parameters** list.

2 Click the **Delete** button. The parameter is removed from the list.

Note: When you delete an action parameter, make sure that you also delete any steps that use the action parameter.

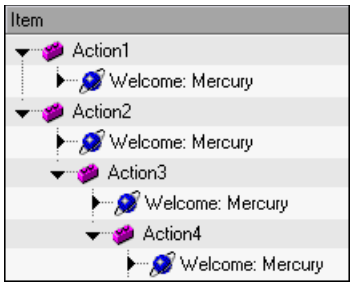
Using Action Parameters

Action parameters enable you to transfer input values from your test to a top-level action, from a parent action to a nested action, or from an action to a sibling action that occurs later in the test. Action parameters also enable you to transfer output values from a step in an action to its parent action, or from a top-level action back to the script or application that ran (called) your test. For example, you can output a value from a step in a nested action and store it in an output action parameter, and then use that value as input in a later step in the calling parent action.

You can use action parameters in any step in your action (including function calls). You define the parameters that an action can receive and the output values that it can return in the Parameters tab of the Action Properties dialog box (**Edit > Action > Action Properties** or right-click an action and select **Action Properties**). You specify the actual values that are provided to these parameters and the locations in which the output values are stored using the Parameter Values tab in the Action Call Properties dialog box (opened by right-clicking an action and choosing **Action Call Properties**).

You can specify input parameters for an action so it can receive input values from elsewhere in the test. Input values for an action parameter can be retrieved from the test (for a top-level action), from the parameters of the parent action that calls it (for a nested action), or from the output of a previous action call (for a sibling action). You can also specify output parameters for an action, so that it can output values for use later in the test, or pass values back to the application that ran (called) the test.

For example, suppose you want to take a value from the external application that runs (calls) your test and use it in an action within your test. In the test below, you would need to pass the input test parameter from the external application through Action2 and Action3 to the required step in Action4.



You would do this as follows:

1. Define the input test parameter (**File** > **Settings** > **Parameters** node) with the value that you want to use later in the test.
2. Define an input action parameter for Action2 (**Edit** > **Action** > **Action Properties** > **Parameters** tab) with the same value type as the input test parameter.
3. Parameterize the input action parameter value (**Edit** > **Action** > **Action Call Properties** > **Parameter Values** tab) using the input test parameter value you specified above.
4. Define an input action parameter for Action3 (**Edit** > **Action** > **Action Properties** > **Parameters** tab) with the same value type as the input test parameter.
5. Parameterize the input action parameter value.
   - ➤ Select **Edit** > **Action** > **Action Call Properties** > **Parameter Values** tab and select the input action parameter value you specified for Action2.
   - ➤ Use the Parameter utility object to specify the action parameter as the *Parameters* argument for the RunAction statement in the Expert View. For more information, see "Calling Actions with Parameters" on page 489.
6. Define an input action parameter for Action4 (**Edit** > **Action** > **Action Properties** > **Parameters** tab) with the same value type as the input test parameter.

7 Parameterize the input action parameter value.

- ➤ Select **Edit** > **Action** > **Action Call Properties** > **Parameter Values** tab and select the input action parameter value you specified for Action3.
- ➤ Use the Parameter utility object to specify the action parameter as the *Parameters* argument for the RunAction statement in the Expert View. For more information, see "Calling Actions with Parameters" on page 489.

8 Parameterize the value in the required step in Action4.

- ➤ Click the parameterization icon  and specify the parameter in the Value Configuration Options dialog box using the input action parameter you specified for Action 4.
- ➤ Use the Parameter utility object in the Expert View to specify the value to use for the step. For more information, see "Using Action Parameters in Steps in the Expert View" on page 638.

An action's parameters are stored with the action and are the same for all calls to that action. If you modify an action parameter's name, type, or description, and then view the action properties for a call to that same action in a different part of the test, you will see that the action parameter has changed.

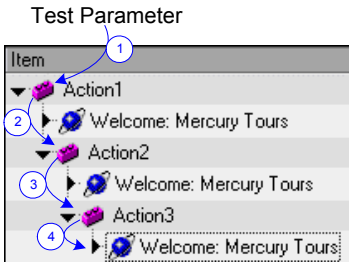
The actual value specified for an input action parameter and the location specified for action output parameter can be different for each call to the action. When you insert a call to a copy of an action, the copy of the action is inserted with the action parameters and action call parameter values that were defined for the action you copied. When you split an action, the action parameters are copied to both actions. The action call values for the second action are taken from the default values of that action's parameters.

For information on defining action parameters and the values used in action calls, see "Setting Action Parameters" on page 472, and "Setting Action Call Parameter Values" on page 483.

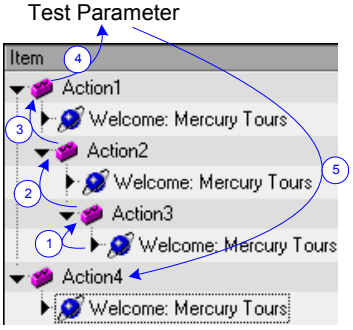
Guidelines for Working with Action Parameters

Consider the following guidelines when working with action parameters:

- Input action parameter values can be used only within the steps of the current action. You can use an action input value from another action (or from the test) only if you pass the value from action to action down the test hierarchy to the action in which you want to use it. For example:
Test -> Action1 -> Action2 -> Action3 -> (Action3) Step 1

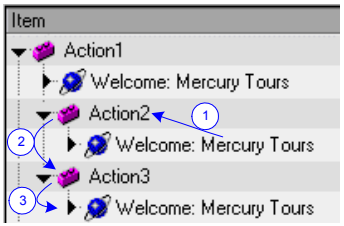


- Output action parameter values can be retrieved from a previous action at the same hierarchical level, from a parent action, or from the current action. You can use an action output value from one action within the step of another action if:
  - You pass the value from action to action up the test hierarchy to the action in which you want to use it. For example:
  (Action3) Step 1-> Action3 -> Action2 -> Action1 -> Test -> Action4



In this example, any step in Action 1, Action 2, or Action 3 can potentially use the output value from (Action3) Step 1, even though the example shows that the output value is used by steps in Action4.

- You pass the value from a previous action to the sibling action in which you want to use it. For example:
  (Action2) Step 1 -> Action2 -> Action3 -> (Action3) Step 1



In this example, any step in Action 2 or Action 3 can potentially use the output value from (Action2) Step 1, even though the example shows that the output value is used by (Action3) Step 1.

- In subsequent steps of a calling action, you can use any type of action output value as a variable, if the value was retrieved from the called action. For example, if ActionA calls ActionB and specifies MyBVar as the variable in which to store ActionB's output parameter, then steps in ActionA after the call to ActionB can use the MyBVar as a value (just as you would use any other variable).

Setting Action Call Properties

The Action Call Properties dialog box controls the way the action behaves in a specific call to the action. It enables you to specify how many times QuickTest should run the called action (according to the number of rows in the Data Table), and also to specify the initial value for any input action parameters and the location in which you want to store the values of any output action parameters.

Note: The following sections describe how to define action call properties using the Action Call Properties dialog box. You can also define action calls and action call parameters in the Expert View. For more information, see "Understanding Action Syntax in the Expert View" on page 488.

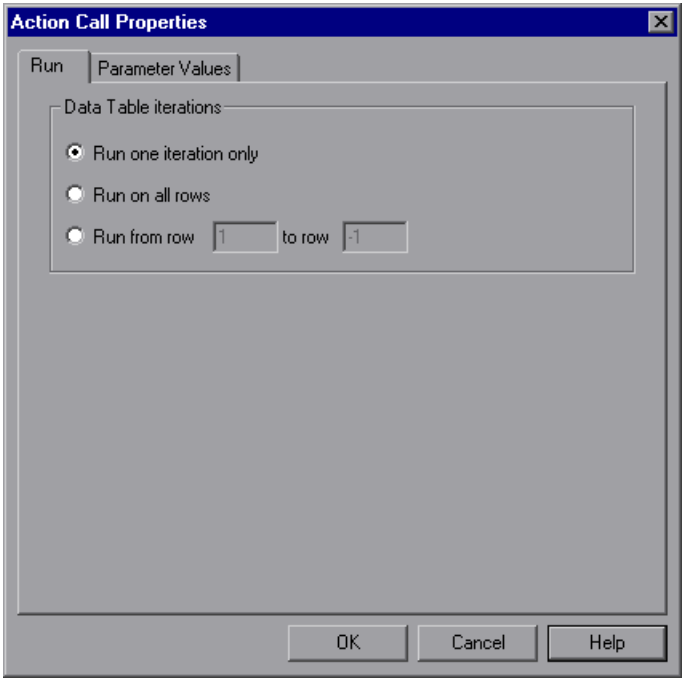
You can open the Action Call Properties dialog box by:

- Right-clicking an action node in the Test Flow pane and selecting **Action Call Properties**.
- Right-clicking an action node in the Keyword View and selecting **Action Call Properties**.
- Choosing **Edit** > **Action** > **Action Call Properties** from the Keyword View when an action node is highlighted.

The Action Call Properties dialog box enables you to set options that apply only to a specific action call. The dialog box contains both the Run tab and the Parameter Values tab.

Setting the Run Properties for an Action

You can use the Run tab of the Action Call Properties dialog box to instruct QuickTest to run only one iteration on the called action, to run iterations on all rows in the Data Table, or to run iterations only for a certain row range in the Data Table.



The Run tab includes the following options:

Option	Description
Run one iteration only	Runs the called action only once, using the first row in the action's data sheet.
Run on all rows	Runs the called action with the number of iterations according to the number of rows in the action's Data Table.
Run from row __ to row __	Runs the called action with the number of iterations according to the specified row range.

Notes:

- If you run multiple iterations on an action, the action must begin and end at the same point in the application, so that the application is in the proper location and state to run the next iteration of the action.
- The Run tab of the Action Call Properties dialog box applies to individual action calls and refers to the rows in the action's data sheet. You can set the Run properties for an entire test (setting iterations for rows on the Global data sheet) from the Run pane in the Test Settings dialog box. For more information, see Chapter 45, "Setting Options for Individual Tests."

Setting Action Call Parameter Values

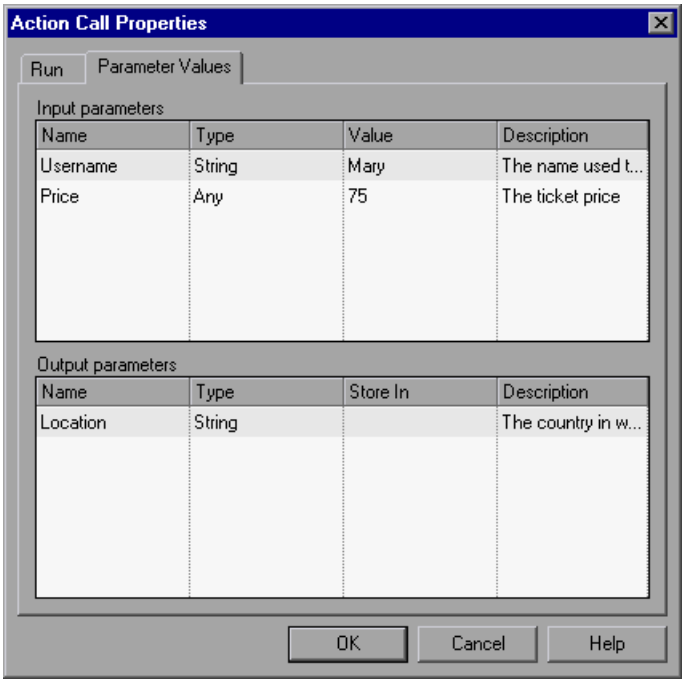
You use the Parameter Values tab of the Action Call Properties dialog box to specify the values of input action parameters used by the called action and to specify the locations in which you want to store output action parameter values. You can also parameterize the value used for a particular input action parameter using any available parameter type.

Note: Specifying input and output parameter values in action calls is optional.

If you do not set a value for an input action parameter, the default value that is specified in the Action Properties dialog box is used.

If you do not define a storage location for an output parameter value, the calling action still has access to the output parameter data generated by the actions it calls. However, specifying a storage location can make your action call statements more readable.

The actual input and output action parameters that an action can receive or return, and their types, are defined in the Action Properties dialog box.



For more information on defining input and output action parameters, see “Setting Action Call Properties” on page 481. For general information on using action parameters, see “Using Action Parameters” on page 476.

To specify the value for an input action parameter:

1 In the **Input parameters** area, click in the **Value** box for the parameter and enter a value. For a description of the different options available for each value type, see the definitions included in "Setting Action Parameters" on page 472.

Alternatively, you can click the parameterization button  in the **Value** box to open the Value Configuration Options dialog box in which you can parameterize the value. You can parameterize the value using a test or action parameter (test parameter for a top-level action, or action parameter for a nested or sibling action), Data Table parameter, environment parameter, or random number parameter. For more information, see Chapter 24, "Parameterizing Values."

2 Repeat this procedure for any additional input action parameter values you want to set.

To specify a location in which to store an output action parameter value:

1 In the **Output parameters** area, click in the **Store In** box for the parameter and enter a variable name.

Alternatively, you can click the output storage button  in the **Store In** box to open the Storage Location Options dialog box in which you can specify a location for storing the output value. You can select to store the value in a test parameter, the calling action parameter, a Data Table parameter, or an environment parameter. For more information, see "Sharing Action Information" on page 486 and "Storing Return Values and Action Output Parameter Values" on page 794.

2 Repeat this procedure for each output action parameter value in the list.

Sharing Action Information

There are several ways to share or pass values from one action to other actions:

- Store values in the output action parameters of a called action and use those values in steps that are performed after the action call within the calling action, or in steps within sibling actions. For more information, see "Storing Values in Test and Action Parameters" on page 673.
- Store values from one action in the global Data Table and use these values as Data Table parameters in other actions. For more information, see "Sharing Values via the Global Data Table" on page 486.
- Set a value from one action as a user-defined environment variable and then use the environment variable in other actions. For more information, see "Sharing Values Using Environment Variables" on page 487.
- Add values to a Dictionary object in one action and retrieve the values in other actions. For more information, see "Sharing Values Using the Dictionary Object" on page 487.

Sharing Values via the Global Data Table

You can share a value that is generated in one action with other actions in your test by storing the value in the global Data Table. Other actions can then use the value in the Data Table as an input parameter. You can store a value in the Data Table by outputting the value to the global Data Table or by using DataTable, Sheet and Parameter objects and methods in the Expert View to add or modify a value.

For example, suppose you are testing a flight reservation application. When a user logs into the application, his or her full name is displayed on the top of the page. Later, when the user chooses to purchase the tickets, the user must enter the name that is listed on his or her credit card. Suppose your test contains three actions—Login, SelectFlight, and PurchaseTickets and the test is set to run multiple iterations with a different login name for each iteration. In the Login action, you can create a text output value to store the displayed name of the user. In the PurchaseTickets action, you can parameterize the value that is set in the Credit Card Owner edit box using the Data Table column containing the user's full name.

For more information on output values, see Chapter 25, "Outputting Values." For more information on parameterization, see Chapter 24, "Parameterizing Values." For more information on DataTable objects and methods, see Chapter 42, "Working with Data Tables," and the *HP QuickTest Professional Object Model Reference*.

Sharing Values Using Environment Variables

If you don't need to run multiple iterations of your test or you want the value you are sharing to stay constant for all iterations, you can use an internal, user-defined environment variable that can be accessed by all local actions in your test.

For example, suppose you want to test that your flight reservation application correctly checks the credit card expiration date that the user enters. The application should request a different credit card if the expiration date that was entered is earlier than the scheduled flight departure date. In the SelectFlight action, you can store the value entered in the departure date edit box in an environment variable. In the PurchaseTickets action, you can compare the value of the expiration date edit box with the value stored in your environment variable.

For more information on environment variables, see Chapter 24, "Parameterizing Values." For information on the Environment object, see the *HP QuickTest Professional Object Model Reference*.

Sharing Values Using the Dictionary Object

As an alternative to using environment variables to share values between actions as described above, you can use the Dictionary object. The Dictionary object enables you to assign values to variables that are accessible from all actions (local and external) called in the test in which the Dictionary object is created.

To use the Dictionary object, you must first add a reserved object to the registry (in **HKEY_CURRENT_USER\Software\Mercury Interactive\QuickTest Professional\MicTest\ReservedObjects**) with ProgID = "Scripting.Dictionary". For example:

```
HKEY_CURRENT_USER\Software\Mercury
Interactive\QuickTest Professional\MicTest\ReservedObjects\GlobalDictionary
```

After you have added the reserved Dictionary object to the registry and restarted QuickTest, you can add and remove values to the Dictionary in one action and retrieve the values in another action from the same test.

For example, if you want to access the departure date set in the SelectFlight action from the PurchaseTickets action, you can add the value of the DepartDate WebEdit object to the dictionary in the SelectFlight action as follows:

```
GlobalDictionary.RemoveAll
GlobalDictionary.Add "DateCheck", DepartDate
```

Then you can retrieve the date from the PurchaseTickets action as follows:

```
Dim CompareDate
CompareDate=GlobalDictionary("DateCheck")
```

For more information on the Dictionary object, see the VBScript Reference documentation (**Help** > **QuickTest Professional Help** > **VBScript Reference** > **Script Runtime**).

Understanding Action Syntax in the Expert View

An action call in the Expert View can define the action iterations, input parameter values, output parameter storage locations, and an action return values.

Calling Actions Using Basic Syntax

In the Expert View, a call to an action with no parameters is displayed within the calling action with the following basic syntax:

RunAction *ActionName*, *IterationQuantity*

For example, to call the **Select Flight** action and run it one iteration:

```
RunAction "Select Flight", oneIteration
```

For example, to call the **Select Flight** action and run it as many iterations as there are rows in the Data Table:

```
RunAction "Select Flight", allIterations
```

For example, to call the **Select Flight** action and run it four iterations (for the first four rows of the Data Table):

```
RunAction "Select Flight", "1 - 4"
```

Calling Actions with Parameters

If the action you are calling has input and/or output parameters defined, you can also supply the values for the input parameters and the storage location of the output parameters as arguments of the RunAction statement. Input parameters are listed before output parameters.

For an input parameter, you can specify either a fixed value or you can specify the name of another defined parameter (Data Table parameter, environment parameter, or an action input parameter of the calling action) from which the argument should take its value.

For an output parameter, you can specify either a variable in which you want to store the value or the name of a defined parameter (Data Table parameter, environment parameter, or an action output parameter of the calling action).

An action call with parameters has the following syntax:

RunAction *ActionName, IterationQuantity, Parameters*

For example, suppose you call Action2 from Action1, and Action2 has one input and one output parameter defined.

The following statement supplies a string value of MyValue for the input parameter and stores the resulting value of the output parameter in a variable called MyVariable.

```
RunAction "Action2", oneIteration, "MyValue", MyVariable
```

The following statement uses the value defined for Action1's Axn1_In input action parameter as the value for the input parameter, and stores the resulting value of the output parameter in Action1's Data Table sheet in a column called Column1_out.

```
RunAction "Action2", oneIteration, Parameter("Axn1_In"),
    DataTable("Column1_out", dtLocalSheet)
```

In the following example, the first statement calls Action2 using its default input parameter value. The second statement uses the value defined for Action2's Axn2_out output action parameter as the value for the call to Action 3's input parameter, and stores the resulting value of the output parameter in Action1's Axn1_out so that the output value is available at the parent action level.

```
RunAction "Action2", oneIteration
RunAction "Action3", oneIteration, Parameter("Action2","Axn2_out"),
     Parameter("Axn1_out")
```

Note that the Action2 output parameter is available for use in the call to Action3, even though no storage location is specified in the call to Action2.

Storing Action Return Values

If the action called by the RunAction statement includes an ExitAction statement, the RunAction statement can return the value of the ExitAction's *RetVal* argument. Note that this return value is a return value of the action call itself and is independent of any values returned by specific output parameters of the action call.

To store the return value of an action call, use the syntax:

MyRetVal=**RunAction (***ActionName, IterationQuantity, Parameters***)**

For more information on the Expert View, see Chapter 29, "Working in the Expert View and Function Library Windows." For more information on the RunAction statement, see the *HP QuickTest Professional Object Model Reference*.

Exiting an Action

You can add a line in your script in the Expert View to exit an action before it runs in its entirety. You may want to use this option to return the current value of the action to the value at a specific point in the run or based on the result of a conditional statement. There are four types of exit action statements you can use:

- **ExitAction.** Exits the current action, regardless of its iteration attributes.
- **ExitActionIteration.** Exits the current iteration of the action.
- **ExitRun.** Exits the test, regardless of its iteration attributes.
- **ExitGlobalIteration.** Exits the current global iteration.

You can view the exit action node in the Test Results tree. If your exit action statement returns a value, the value is displayed in the action, iteration, or test summary, as applicable.

For more information on these functions, see the *HP QuickTest Professional Object Model Reference*. For more information on the Test Results, see Chapter 33, "Viewing Run Session Results."

Part IV

Enhancing Tests

17

Understanding Checkpoints

You can check objects in your application to ensure that they function properly.

This chapter includes:

About Understanding Checkpoints

QuickTest enables you to add checks to your test. A **checkpoint** is a verification point that compares the current value for specified properties with the expected value for those properties. This enables you to identify whether your application is functioning correctly.

When you add a checkpoint, QuickTest inserts a checkpoint step to the current row in the Keyword View and adds a **Check CheckPoint** statement in the Expert View. By default, QuickTest names the checkpoint using the name of the test object on which the checkpoint was created. You can choose to specify a different name for the checkpoint or accept the default name.

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. You can view the results of the checkpoint in the Test Results window.

Tip: You can also use the **CheckProperty** method and the **CheckItemProperty** method to check specific property or item property values. For more information, see the *HP QuickTest Professional Object Model Reference*.

Note: If you want to retrieve the return value of a checkpoint (a boolean value that indicates whether the checkpoint passed or failed), you must add parentheses around the checkpoint argument in the statement in the Expert View. For example:

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

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

Adding New Checkpoints to a Test

You can add checkpoints while creating or editing your test. It is generally more convenient to define checkpoints after creating the initial test.

Note: You can also add an existing checkpoint to your test. For more information, see "Adding Existing Checkpoints to a Test" on page 498.

To add new checkpoints while editing or recording your test:

Use the commands in the **Insert** > **Checkpoint** menu, or click the **Insert Checkpoint** button in the toolbar. This displays a menu of checkpoint options that are relevant to the selected step.

To add new checkpoints while editing only:

- Right-click the step where you want to add the checkpoint and select the relevant checkpoint option.
- Select the step where you want to add the checkpoint, select **Insert** > **Checkpoint**, and then select the relevant checkpoint option.
- Right-click any object in the Active Screen and select the relevant checkpoint option. These options can be used to create checkpoints for any object in the Active Screen (even if the object is not part of any step in the Keyword View).

Notes:

- If you use the Active Screen option, ensure that the Active Screen contains sufficient data for the object you want to check. For more information, see "Setting Active Screen Options" on page 1240.
- Throughout this guide, procedures for creating checkpoints may be described using only one of the above methods. However, you can choose any of the methods described above.

Adding Existing Checkpoints to a Test

QuickTest enables you to reuse the existing checkpoints in your test. When you insert checkpoints into your test, consider which checkpoints can be reused in multiple locations in your test. For example:

- Checkpoints that check generic content or the state of your application may be useful in multiple locations.
- Checkpoints that check the content of a specific area of your application are generally useful in only one particular place in your test.

The following examples illustrate situations in which inserting an existing checkpoint may be useful:

- If each page of your application contains your organization's logo, you can reuse a bitmap checkpoint to verify each occurrence in the application.
- If your application contains multiple edit boxes, you can reuse a checkpoint to confirm the enabled status of these edit boxes throughout your test.

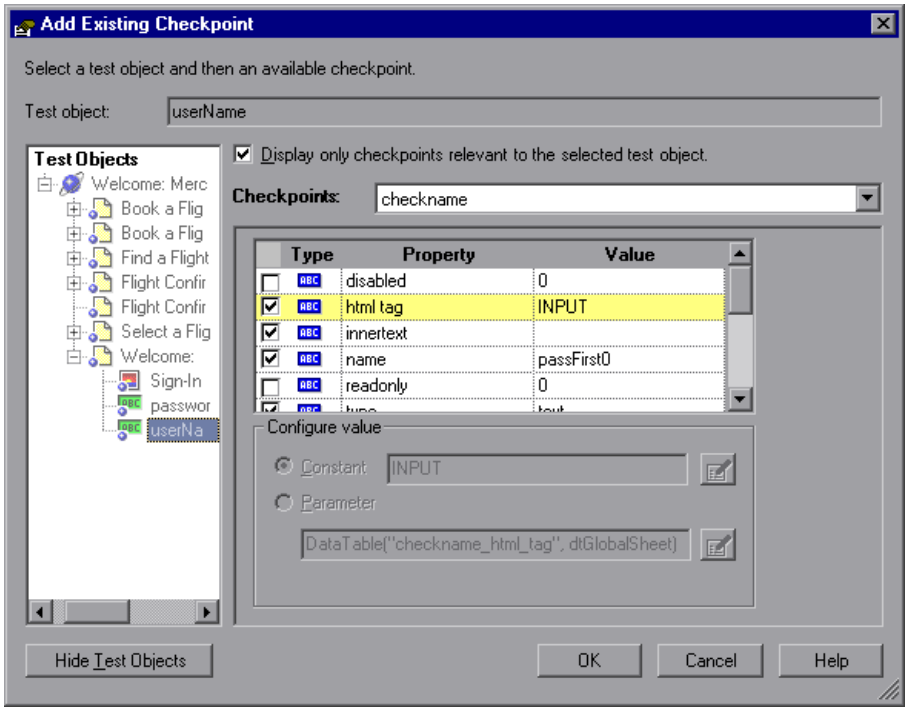
Understanding the Add Existing Checkpoint Dialog Box

You open the Add Existing Checkpoint dialog box by selecting **Insert** > **Checkpoint** > **Existing Checkpoint**. This option is available only if at least one of the object repositories associated with the current action (including the local object repository) contains at least one checkpoint.

If a test object step is highlighted in the Keyword View or the cursor is located in a step in the Expert View, the Add Existing Checkpoint dialog box opens with the **TestObjects** tree hidden.

The test object displayed in the **Test object** box is the object from the highlighted step in the Keyword View or the specific object where the cursor is located in the Expert View.

You can display or hide the **TestObjects** tree by clicking the **Show/Hide Test Objects** button.



The Add Existing Checkpoint dialog box contains the following items:

Item	Description
Test object	Specifies the test object for which you are adding a checkpoint.
TestObjects tree	Displays every object in the current test.
Show/Hide Test Objects	Shows or hides the **TestObjects** tree.
Display only checkpoints relevant to the selected test object	When selected, QuickTest determines which checkpoints from the current action's object repositories are relevant for the selected object (based on the checkpoint type and the properties selected in the checkpoint) and displays only those checkpoints in the **Checkpoints** list. When using this option, it is recommended to open your application and display the selected object so that QuickTest can accurately determine all of the checkpoints that can apply to that object.
Checkpoints	Lists the checkpoints available for insertion. If the **Display only checkpoints relevant to the selected test object** option is cleared, this list includes all checkpoints from all object repositories associated with the current action. If the **Display only checkpoints relevant to the selected test object** option is selected, this list displays only the relevant checkpoints as described above.
Properties Area	Displays the checkpoint properties for the selected checkpoint in read-only format.

To insert an existing checkpoint in your test:

1. Display the action in which you want to insert the checkpoint and select the step after which you want to insert the checkpoint.
2. Select **Insert** > **Checkpoint** > **Existing Checkpoint**. The Add Existing Checkpoint dialog box opens.
3. If the **TestObjects** tree is displayed, select the object for which you want to insert a checkpoint. Otherwise proceed to step 4.
4. From the **Checkpoints** list, select the checkpoint that you want to insert for the object displayed in the **Test object** box.
5. Click **OK**. The checkpoint is inserted after the current step.

Understanding Types of Checkpoints

You can insert the following checkpoint types to check various objects in an application.

- **Standard Checkpoint** checks the property value of an object in your application. The standard checkpoint checks a variety of objects such as buttons, radio buttons, combo boxes, lists, and so forth. For example, you can check that a radio button is activated after it is selected or you can check the value of an edit box.

  Standard checkpoints are supported for all add-in environments (see "Supported Checkpoints" on page 504).

  For more information on standard checkpoints, see Chapter 18, "Checking Object Property Values Using Standard Checkpoints."

- **Image Checkpoint** checks the value of an image in your application. For example, you can check that a selected image's source file is correct.

Note: You create an image checkpoint by inserting a standard checkpoint on an image object.

Image checkpoints are supported for the Web add-in environment (see "Supported Checkpoints" on page 504).

For more information on image checkpoints, see Chapter 18, "Checking Object Property Values Using Standard Checkpoints."

- **Bitmap Checkpoint** checks an area of your application as a bitmap. For example, suppose you have a Web site that can display a map of a city the user specifies. The map has control keys for zooming. You can record the new map that is displayed after one click on the control key that zooms in the map. Using the bitmap checkpoint, you can check that the map zooms in correctly.

  You can create a bitmap checkpoint for any area in your application, including buttons, text boxes, and tables.

  Bitmap checkpoints are supported for all add-in environments (see "Supported Checkpoints" on page 504).

  For more information on bitmap checkpoints, see Chapter 19, "Checking Bitmaps."

- **Table Checkpoint** checks information within a table. For example, suppose your application contains a table listing all available flights from New York to San Francisco. You can add a table checkpoint to check that the time of the first flight in the table is correct.

Note: You create a table checkpoint by inserting a standard checkpoint on a table object.

Table checkpoints are supported for Web, ActiveX, Java, Oracle, and .NET Windows Forms environments, as well as other add-in environments (see "Supported Checkpoints" on page 504). Table checkpoints are also supported for some list view objects, such as WinListView and VbListView, as well as other list view objects in add-in environments.

For more information on table checkpoints, see "Checking Tables" on page 529.

- **Text Checkpoint** checks that a text string is displayed in the appropriate place on a Web page or application. For example, suppose a Web page displays the sentence Flight departing from New York to San Francisco. You can create a text checkpoint that checks that the words "New York" are displayed between "Flight departing from" and "to San Francisco".

  Text checkpoints are supported for most add-in environments (see "Supported Checkpoints" on page 504).

  For more information on text checkpoints, see Chapter 21, "Checking Text."

- **Text Area Checkpoint** checks that a text string is displayed within a defined area in a Windows-based application, according to specified criteria. For example, suppose your Visual Basic application has a button that says View Doc <Num>, where <Num> is replaced by the four digit code entered in a form elsewhere in the application. You can create a text area checkpoint to confirm that the number displayed on the button is the same as the number entered in the form.

  Text area checkpoints are supported for all Windows-based environments, such as Standard Windows, Visual Basic, and ActiveX add-in environments (see "Supported Checkpoints" on page 504). Text area checkpoints are also supported for some other add-in environments, such as Java.

  For more information on text area checkpoints, see Chapter 21, "Checking Text."

- **Accessibility Checkpoint** identifies areas of your Web site that may not conform to the World Wide Web Consortium (W3C) Web Content Accessibility Guidelines. For example, guideline 1.1 of the W3C Web Content Accessibility Guidelines requires you to provide a text equivalent for every non-text element. You can add an **Alt** property check to check whether objects that require the **Alt** property under this guideline, do in fact have this tag.

  Accessibility checkpoints are supported for the Web add-in environment (see "Supported Checkpoints" on page 504).

  For more information on accessibility checkpoints, see the section on testing Web objects in the *HP QuickTest Professional Add-ins Guide*.

➤ **Page Checkpoint** checks the characteristics of a Web page. For example, you can check how long a Web page takes to load or whether a Web page contains broken links.

Note: You create a page checkpoint by inserting a standard checkpoint on a page object.

Page checkpoints are supported for the Web add-in environment (see "Supported Checkpoints" on page 504).

For more information on page checkpoints, see the section on testing Web objects in the *HP QuickTest Professional Add-ins Guide*.

➤ **Database Checkpoint** checks the contents of a database accessed by your application. For example, you can use a database checkpoint to check the contents of a database containing flight information for your Web site.

Database checkpoints are supported for all add-in environments (see "Supported Checkpoints" on page 504).

For more information on database checkpoints, see Chapter 22, "Checking Databases."

➤ **XML Checkpoint** checks the data content of XML documents in XML files or XML documents in Web pages and frames. For more information on XML checkpoints, see Chapter 23, "Checking XML."

The **XML Checkpoint (Web Page/Frame)** option is supported for the Web add-in environment. The **XML Checkpoint** option is supported for all add-in environments (see "Supported Checkpoints" on page 504).

Supported Checkpoints

QuickTest add-ins help you to create and run tests and components on applications in a variety of development environments. For information about using checkpoints for each add-in environment installed with QuickTest Professional, see "Supported Checkpoints" on page 1546.

18

Checking Object Property Values Using Standard Checkpoints

By adding standard checkpoints to your tests, you can compare object property values in your application with the expected values.

This chapter includes:

About Checking Object Property Values

You can check the object property values in your application using standard checkpoints. Standard checkpoints compare the expected values of object properties to the object's current values during a run session. You can create standard checkpoints for all supported testing environments (as long as the appropriate add-ins are loaded).

You use standard checkpoints to perform checks on images, tables, Web page properties, and other objects within your application.

Creating Standard Checkpoints

You can check that a specified object in your application has the property values you expect, by adding a standard checkpoint step to your test while recording or editing the test. To set the options for a standard checkpoint, you use the Checkpoint Properties dialog box.

To add a standard checkpoint while recording:

1 While in a recording session, select **Insert** > **Checkpoint** > **Standard Checkpoint**, or click the **Insert Checkpoint or Output Value** toolbar button.

The QuickTest window is hidden, and the pointer changes into a pointing hand.

Tips:

- If the window on which you want to spy 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 hold the left CTRL key to change the pointing hand to a standard pointer, and then change the window focus or perform operations, such as right-clicking or moving the pointer over an object to display a context menu.
- 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.

---

2 Click the object you want to check. The Object Selection - Checkpoint Properties dialog box opens.

3 Select the item you want to check from the displayed object tree. The tree item name corresponds to the object's class.

4 Click **OK**. The Checkpoint Properties dialog box opens.

5 Specify the settings for the checkpoint. For more information, see "Understanding the Checkpoint Properties Dialog Box" on page 508.

6 Click **OK** to close the dialog box. A checkpoint statement is added for the selected object in the Keyword View and Expert View.

To add a standard checkpoint while editing:

1 Perform one of the following:

- Right-click the step on which you want to perform a checkpoint and select **Insert Standard Checkpoint**.
- Select the step where you want to add the checkpoint and select **Insert** > **Checkpoint** > **Standard Checkpoint**.
- Right-click any object in the Active Screen and select **Insert Standard Checkpoint**.

The Checkpoint Properties dialog box opens.

Note: To add a standard checkpoint while editing, one of the following must be true:

- Active Screen information exists for the step. For more information, see "Working with the Active Screen" on page 376.
- The object for which you want to create a checkpoint is currently displayed in the application.

Depending on the object and environment, it may be necessary for both of these conditions to be true. For more information, see the chapter for your environment in the *HP QuickTest Professional Add-ins Guide*.

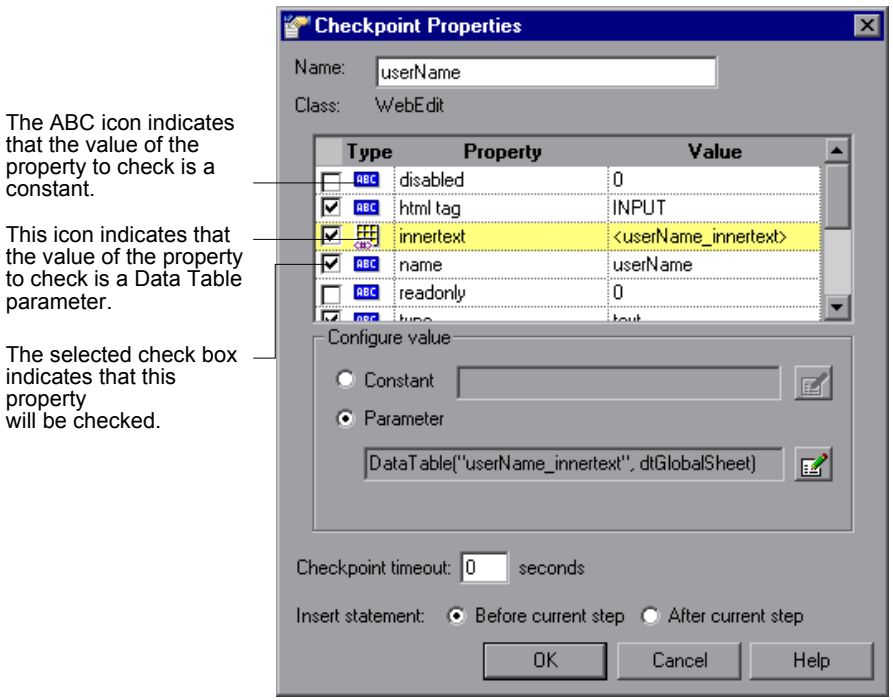
---

2 Specify the settings for the checkpoint. For more information, see "Understanding the Checkpoint Properties Dialog Box" on page 508.

3 Click **OK** to close the dialog box. A checkpoint statement is added for the selected object in the Keyword View and Expert View.

Understanding the Checkpoint Properties Dialog Box

In the Checkpoint Properties dialog box, you can specify which properties of the object to check, and edit the values of these properties. While the specific elements vary slightly depending on the type of object you are checking, the Checkpoint Properties dialog box generally includes the following basic elements:



The dialog box described above is used to configure most standard checkpoints. Certain standard checkpoint types, however, employ different dialog boxes, as follows:

For information on the Dialog Box for:	See:
Image checkpoint properties	"Understanding the Image Checkpoint Properties Dialog Box" on page 512
Page checkpoint properties	The section on checking Web pages in the *HP QuickTest Professional Add-ins Guide*
Table checkpoint properties	"Understanding the Table Checkpoint Properties Dialog Box" on page 535

Identifying the Checkpoint

The top part of the dialog box displays information on the checkpoint:

Information	Description
Name	The name of the checkpoint. By default, the checkpoint name is the same as the name of the test object on which the checkpoint was created. You can specify a different name for the checkpoint or accept the default name. If you rename the checkpoint, make sure that the name: ➤ is unique ➤ does not begin or end with a space ➤ does not contain " (double quotation mark) ➤ does not contain the following character combinations: := @@
Class	The type of object. In this example, the **WebEdit** class indicates that the object is an edit box.

Information	Description
Find in Repository button 	Displays the checkpoint in its object repository. **Note:** This option is available only when editing an existing checkpoint. It is not available when creating a new checkpoint. When available, it is located to the right of the **Name** box.

Selecting the Object Property to Check

The properties for the object are listed in the Properties pane of the dialog box. The pane includes the properties, their values, and their types:

Pane Element	Description
Check box	For each object class, QuickTest recommends default property checks. You can accept the default checks or modify them accordingly. To check a property, select the corresponding check box. To exclude a property check, clear the corresponding check box.
Type	The  icon indicates that the value of the property is currently a constant. The  icon indicates that the value of the property is currently a test or action parameter. The  icon indicates that the value of the property is currently a Data Table parameter. The  icon indicates that the value of the property is currently an environment variable parameter. The  icon indicates that the value of the property is currently a random number parameter.
Property	The name of the property.
Value	The expected value of the property. For more information on modifying the value of a property, see “Setting Values in the Configure Value Area” on page 757.

Editing the Expected Value of an Object Property

In the **Configure value** area, you can define the expected value of the property to check as a **Constant** or **Parameter**. For information on modifying property values, see "Setting Values in the Configure Value Area" on page 757.

Setting the Checkpoint Timeout Option

Checkpoint timeout. Specifies the time interval (in seconds) during which QuickTest attempts to perform the checkpoint successfully. QuickTest continues to perform the checkpoint until it passes or until the timeout occurs. If the checkpoint does not pass before the timeout occurs, the checkpoint fails.

For example, suppose it takes some time for an object to achieve an expected state. Increasing the checkpoint timeout value in this case can help ensure that the object has sufficient time to achieve that state, enabling the checkpoint to pass (if the data matches) before the maximum timeout is reached.

If you specify a checkpoint timeout other than **0**, and the checkpoint fails, the Test Results window displays information on the checkpoint timeout.

Inserting the Checkpoint in Your Test

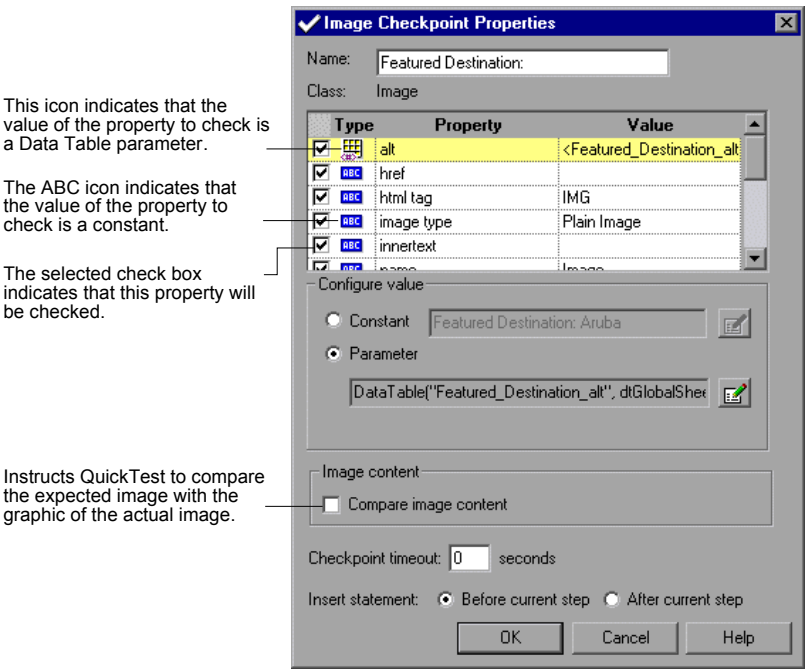
The **Insert statement** option specifies when to perform the checkpoint in the test.

- Select **Before current step** if you want to check the value of the object property before the highlighted step is performed.
- Select **After current step** if you want to check the value of the property after the highlighted step is performed.

Note: The **Insert statement** option is not available when adding a checkpoint during recording or when modifying an existing object checkpoint. It is available only when adding a new checkpoint to an existing test while editing it.

Understanding the Image Checkpoint Properties Dialog Box

Image checkpoints enable you to check the properties of a Web image. In the Image Checkpoint Properties dialog box, you can specify which properties of the image to check and edit the values of those properties. This dialog box is similar to the standard Checkpoint Properties dialog box, except that it contains the **Compare image content** option. This option enables you to compare the expected image source file with the actual image source file.



Identifying the Checkpoint

The top part of the dialog box displays information on the checkpoint:

Information	Description
Name	The name that QuickTest assigns to the checkpoint. By default, the checkpoint name is the name of the test object on which the checkpoint is being performed. You can specify a different name for the checkpoint or accept the default name. If you rename the checkpoint, make sure that the name: ➤ is unique ➤ does not begin or end with a space ➤ does not contain " (double quotation mark) ➤ does not contain the following character combinations: := @@
Class	The type of object. This is always **Image**.
Find in Repository button  (Located to the right of the **Name** box)	Displays the checkpoint in its object repository. **Note:** This option is available only when editing an existing checkpoint. It is not available when creating a new checkpoint.

Selecting the Image Property to Check

The default properties for the image are listed in the Properties pane of the dialog box. This pane includes the properties, their values, and their types. It is identical to the Properties pane in the Checkpoint Properties dialog box for standard checkpoints. For more information, see “Selecting the Object Property to Check” on page 510.

Editing the Expected Value of an Image Property

The middle part of the Image Checkpoint Properties dialog box contains the following:

- **Configure value.** Enables you to define the expected value of the property as a **Constant** or **Parameter**. For information on modifying property values, see "Setting Values in the Configure Value Area" on page 757.
- **Compare image content.** Compares the expected image source file with the graphic of the actual image source file. If the expected and actual images are different, QuickTest displays them both in the Test Results. If the images are identical, only one graphic is displayed.

Setting General Image Checkpoint Options

The bottom part of the Image Checkpoint Properties dialog box contains the **Checkpoint timeout** and **Insert statement** options. These options are identical to those in the Checkpoint Properties dialog box for standard checkpoints. For more information, see "Setting the Checkpoint Timeout Option" on page 511 and "Inserting the Checkpoint in Your Test" on page 511.

Modifying Checkpoints

You can modify the settings of existing checkpoints.

To modify a checkpoint:

1. In the Keyword View or Expert View, right-click the checkpoint that you want to modify and select **Checkpoint Properties**. Alternatively, select the step containing the checkpoint and select **Edit** > **Step Properties** > **Checkpoint Properties**. The relevant checkpoint dialog box opens.

2. Modify the properties and click **OK**. For more information, see "Understanding the Checkpoint Properties Dialog Box" on page 508.

19

Checking Bitmaps

QuickTest enables you to check that the visible parts of your application are displayed correctly by comparing bitmaps of objects in your application to bitmaps captured previously and stored with the test.

This chapter includes:

About Checking Bitmaps

You can check an area of an application as a bitmap. You can check an entire object or any area within an object. For example, suppose you have a Web site that can display a map of a city the user specifies. The map has control keys for zooming. You can record the new map that is displayed after one click on the control key that zooms in the map. Using the bitmap checkpoint, you can check that the map zooms in correctly.

You can create bitmap checkpoints for all supported testing environments (as long as the appropriate add-ins are loaded).

The results of bitmap checkpoints may be affected by factors such as operating system, screen resolution, and color settings.

When you create a bitmap checkpoint, QuickTest captures the **visible** part of the specified object as a bitmap (QuickTest does not capture any part that is scrolled off the screen, or hidden by another object, for example), and inserts a checkpoint in the test.

When you run the test, QuickTest captures a bitmap of the actual object in the application and compares this bitmap (or a selected area within it) with the bitmap stored in the checkpoint.

If there are differences, QuickTest saves the bitmap of the actual object and displays it next to the expected bitmap in the details pane of the Test Results window. In the Test Results window you can also view a bitmap that reflects the difference between the two bitmaps, to assist you in identifying the nature of the discrepancy. You can configure QuickTest not to save the bitmaps in the test results, or to save them even if the checkpoint passes (**Tools > Options > Run > Screen Capture** pane). For more information on test results of a checkpoint, see "Viewing Checkpoint Results" on page 1028.

Fine-Tuning the Bitmap Comparison

When running a bitmap checkpoint, QuickTest compares the area that you are checking in the application with the bitmap stored in the checkpoint, pixel by pixel. By default, if any pixels are different, the checkpoint fails. The Bitmap Checkpoint Properties dialog box (described on page 522) provides options for fine-tuning the bitmap comparison.

You can adjust the comparison to enable the checkpoint to pass even if the bitmaps are not identical by setting the **RGB tolerance** and **Pixel tolerance** options described below.

In addition, QuickTest enables you to use **custom comparers** for bitmap checkpoints. A custom comparer is a COM object that you or a third party developed to run the bitmap comparison in the checkpoint according to a more specific algorithm. If one or more custom comparers are installed and registered on the QuickTest computer, the Bitmap Checkpoint Properties dialog box includes a **Comparer** option.

This option enables you to select the QuickTest default comparer or a custom comparer that performs the bitmap comparison according to your testing requirements. For an example on when it can be useful to create a custom comparer, see "Use-Case Scenario: Handling Images Whose Location in the Application Changes" on page 1577. For more information on developing custom comparers, see Appendix D, "Bitmap Checkpoint Customization."

If you select a custom comparer, some of the options in the Bitmap Checkpoint Properties dialog box are different. For more information, see "The Bitmap Checkpoint Properties Dialog Box" on page 522.

Bitmap Checkpoint Tolerance Options

➤ **RGB tolerance.** The RGB (Red, Green, Blue) tolerance determines the percent by which the RGB values of the pixels in the actual bitmap can differ from those of the expected bitmap and allow the checkpoint to pass. (The RGB tolerance option is limited to bitmaps with a color depth of 24 bits.)

For example, a bitmap checkpoint on identical bitmaps could fail if different display drivers are used when you create your checkpoint and when you run your test. Suppose one display driver displays the color white as RGB (255, 255, 255) and another driver displays the color white as RGB (231, 231, 231). The difference between these two values is about 9.4%. By setting the **RGB tolerance** to 10%, your checkpoint will pass when running your test with either of these drivers.

Note: QuickTest applies the RGB tolerance settings when comparing each pixel in the actual and expected bitmaps. The Red, Green, and Blue values for each pixel are compared separately. If any of the values differs more than the tolerance allows, the pixel fails the comparison.

➤ **Pixel tolerance.** The pixel tolerance determines the number or percentage of pixels in the actual bitmap that can differ from those in the expected bitmap and allow the checkpoint to pass.

For example, suppose the expected bitmap has 4000 pixels. If you define the pixel tolerance to be 50 and select the **Pixels** radio button, up to 50 pixels in the actual bitmap can be different from those in the expected bitmap and the checkpoint passes. If you define the pixel tolerance to be 5 and select the **Percent** radio button, up to 200 pixels (5 percent of 4000) in the actual bitmap can be different from those in the expected bitmap and the checkpoint passes.

Using both RGB and Pixel Tolerances

If you define both RGB and pixel tolerances, the RGB tolerance is calculated first. The pixel tolerance then defines the maximum number of pixels that can fail the RGB criteria and allow the checkpoint to pass.

For example, suppose you define an RGB tolerance of 10 percent and a pixel tolerance of 5 percent for a bitmap that has 4000 pixels.

For the checkpoint to pass, each pixel in the actual bitmap must have RGB values that are no greater than or no less than 10 percent of the RGB values of the expected bitmap. If that criterion fails, QuickTest checks that the number of pixels that failed are less than 200. If that criterion passes, the checkpoint passes.

Creating and Modifying Bitmap Checkpoints

You insert a bitmap checkpoint while recording or editing a test. You can also modify an existing bitmap checkpoint.

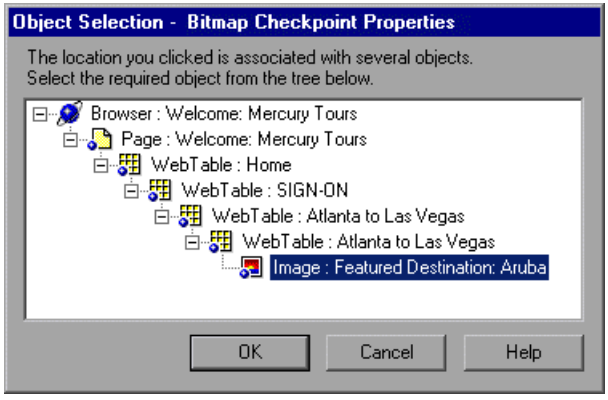
Bitmap checkpoints can capture only the visible part of an object. Therefore, confirm that the object to capture is always fully visible on the screen before a bitmap checkpoint step is performed. One way to do this is to insert a MakeVisible statement (for relevant environments) prior to your bitmap checkpoint step. For more information on the MakeVisible method, see the *QuickTest Object Model Reference*.

To create a bitmap checkpoint while recording:

1 Select **Insert** > **Checkpoint** > **Bitmap Checkpoint**, or click the **Insert Checkpoint or Output Value** button and select **Bitmap Checkpoint**.

The QuickTest window is hidden, and the pointer turns into a pointing hand. For more information about using the pointing hand feature, see "Tips for Using the Pointing Hand" on page 521.

2 Click an object to check in your application. If the location you click is associated with more than one object, the Object Selection - Bitmap Checkpoint Properties dialog box opens.



3 Select an object from the tree on which to create the bitmap checkpoint.

Tip: If you want to create a bitmap checkpoint that contains multiple objects, you should select the highest level object that includes all the objects to include in the bitmap checkpoint.

---

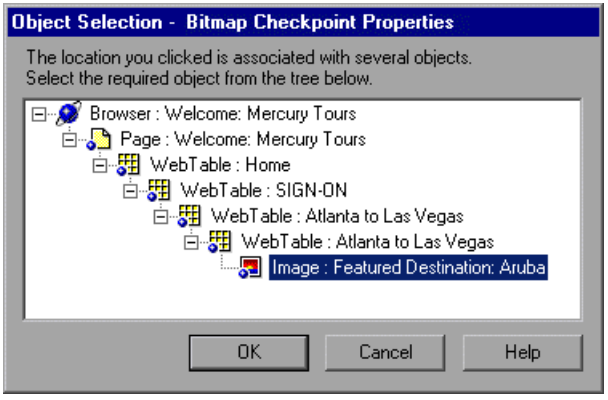
4 Click **OK**. The Bitmap Checkpoint Properties dialog box opens. Create the Bitmap checkpoint using the options in the dialog box. For more information, see "The Bitmap Checkpoint Properties Dialog Box" on page 522.

To create a bitmap checkpoint while editing:

1 Make sure the **Active Screen** button is selected.

2 Click the step in the Keyword View for which you want to add a checkpoint. The Active Screen displays the area of the application corresponding to the highlighted step.

3 Right-click an object in the Active Screen and select **Insert Bitmap Checkpoint**. If the location you click is associated with more than one object, the Object Selection - Bitmap Checkpoint Properties dialog box opens.



4 Select an object from the tree on which to create a bitmap checkpoint.

Tips:

- Ensure that the object you select is completely visible. If another application is overlapping the object, it is also captured.
- To create a bitmap checkpoint that contains multiple objects, select the highest level object that includes all the objects to include in the bitmap checkpoint.

---

5 Click **OK**. The Bitmap Checkpoint Properties dialog box opens. Create the Bitmap checkpoint using the options in the dialog box. For more information, see "The Bitmap Checkpoint Properties Dialog Box" on page 522.

To modify a bitmap checkpoint:

1 Select the step containing the checkpoint and select **Edit** > **Step Properties** > **Checkpoint Properties**, or select the **Value** cell in the step and click the **Checkpoint Properties** button. Alternatively, in the Keyword View or Expert View, right-click the checkpoint that you want to modify and select **Checkpoint Properties**.

2 The Bitmap Checkpoint Properties dialog box opens and displays the object or area you saved with the checkpoint. Modify the Bitmap checkpoint using the options in the dialog box. For more information, see "The Bitmap Checkpoint Properties Dialog Box" on page 522.

Tips for Using the Pointing Hand

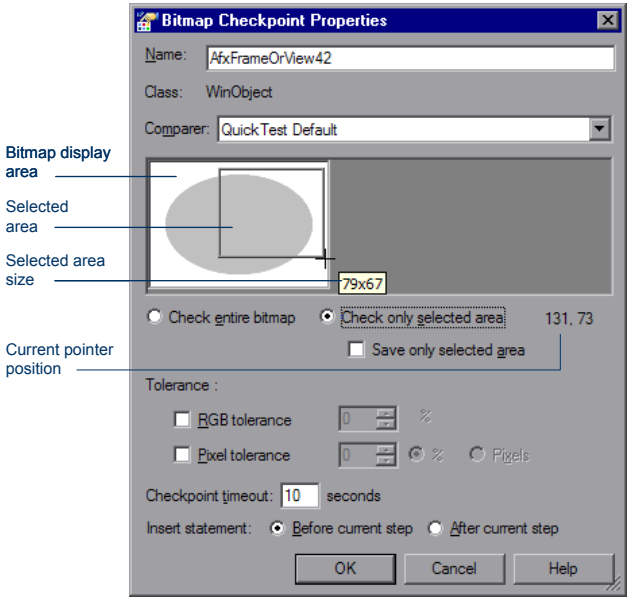
- You can hold the left CTRL key to change the pointing hand to a standard pointer. You can then change the window focus or perform operations in QuickTest or in your application, such as right-clicking, using the scroll bars, or moving the pointer over an object to display a context menu.
- If the window 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.

The Bitmap Checkpoint Properties Dialog Box

Description	Enables you to create or modify a bitmap checkpoint.
How to Access	See: "Creating and Modifying Bitmap Checkpoints" on page 518
Learn More	**Conceptual overview**: "About Checking Bitmaps" on page 515

The following image is an example. It shows the options that are available when selecting an area to check in the bitmap, in a new checkpoint being created while editing a test.



Bitmap Checkpoint Properties Dialog Box Details

This dialog box includes several groups of options, described in the following sections:

- "Descriptive Information" on page 524
- "Options for Selecting the Area to Check" on page 525
- "Tolerance Options" on page 526
- "Checkpoint Timeout and Statement Location Options" on page 527

Descriptive Information

The descriptive information is displayed in the top part of the Checkpoint Properties dialog box.

- **Name**. By default, the checkpoint name is the same as the name of the test object on which the checkpoint was created. Accept the name that QuickTest assigns to the checkpoint or specify another name for it.

  If you rename the checkpoint, make sure that the name:

  - is unique
  - does not begin or end with a space
  - does not contain " (double quotation mark)
  - does not contain the following character combinations:
    :=
    @@

- **Class.** The type of test object on which the checkpoint was created. (Read-only)

- **Comparer.** Enables you to select the comparer for QuickTest to use to run the checkpoint. You can select the QuickTest default comparer or a custom comparer. If you select a custom comparer, some of the options in this dialog box are different. For more information, see "Custom Comparer Options in the Bitmap Checkpoint Properties Dialog Box" on page 527.

  This option is available only if any custom comparers are installed and registered on the QuickTest computer. Otherwise, the QuickTest default comparer is used. For more information, see "Fine-Tuning the Bitmap Comparison" on page 516.

- **Bitmap display area.** Displays a bitmap of the object you selected.

- **Find in Repository.** To view the checkpoint in its repository, click the **Find in Repository** button located to the right of the **Name** box.

  This option is not available when creating a new checkpoint. It is available only when editing an existing checkpoint.

Options for Selecting the Area to Check

The options for selecting the area to check are displayed beneath the bitmap display area.

- **Check entire bitmap / Check only selected area.** Enables you to specify whether the checkpoint compares the entire bitmap or only a specific area of the bitmap. If you select **Check only selected area**, the cursor turns into a crosshairs pointer when you hover over the bitmap display area. Use the crosshairs pointer to draw a rectangle specifying the area that you want to select. To remove the rectangle, click again.

  While the crosshairs pointer is visible, QuickTest displays the coordinates of the pointer's current position beneath the bottom-right corner of the bitmap display area. As you draw the rectangle using the crosshairs, QuickTest displays a tooltip with the current selected area size near the crosshairs pointer.

  If you define the checkpoint to compare only a specific area of the bitmap, the selected area is highlighted also in the actual and expected bitmaps displayed in the Test Results window.

- **Save only selected area.** Enables you to save only the selected area of the object with your test (to save disk space). The bitmap stored in the checkpoint is cropped when you click **OK**. The Test Results window displays only the selected area of the bitmap.

  This option is available only after you select **Check only selected area** and draw the rectangle that specifies the area.

Note: If you select the **Save only selected area** check box, you can later modify the checkpoint by selecting a smaller area within the selected area, but you cannot return the bitmap to its former size. The **Update Run Mode** option (**Automation > Update Run Mode**) only updates 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.

Tolerance Options

The tolerance options are displayed beneath the options for selecting the area to check.

- **RGB tolerance.** Enables you to define the percent by which the RGB values of the pixels in the actual bitmap can differ from those of the expected bitmap and allow the checkpoint to pass.

  Select the check box and modify the percentage manually or by using the up and down arrows. For more information, see "Fine-Tuning the Bitmap Comparison" on page 516.

  This option is limited to bitmaps with a color depth of 24 bits.

- **Pixel tolerance.** Enables you to define the number or percentage of pixels in the actual bitmap that can differ from those in the expected bitmap and allow the checkpoint to pass.

  Select the check box, select either the **Percent** or **Pixels** radio button, and modify the value manually or by using the up and down arrows. If you switch between the **Percent** and **Pixels** radio buttons after entering the tolerance value, the value is recalculated based on your selection. (100% is the total number of pixels in the expected bitmap or selected area.)

  For more information, see "Fine-Tuning the Bitmap Comparison" on page 516.

Checkpoint Timeout and Statement Location Options

The checkpoint timeout and statement location options are displayed in the bottom part of the Checkpoint Properties dialog box.

➤ **Checkpoint timeout.** Enables you to define the time interval (in seconds) during which QuickTest attempts to perform the checkpoint successfully. QuickTest continues to perform the checkpoint until it passes or until the timeout occurs. If the checkpoint does not pass before the timeout occurs, the checkpoint fails.

For example, suppose it takes some time for an object to achieve an expected state. Increasing the checkpoint timeout value in this case can help ensure that the object has sufficient time to achieve that state, enabling the checkpoint to pass (if the data matches) before the maximum timeout is reached.

If you specify a checkpoint timeout other than **0**, and the checkpoint fails, the Test Results window displays information on the checkpoint timeout.

➤ **Before current step / After current step.** Enables you to insert the bitmap checkpoint before or after the highlighted step.

This option is available only when creating a checkpoint while editing a test.

Custom Comparer Options in the Bitmap Checkpoint Properties Dialog Box

In the Bitmap Checkpoint Properties dialog box, if you select a custom comparer to run the bitmap comparison, the options for selecting an area of the bitmap and for setting tolerance levels are not available.

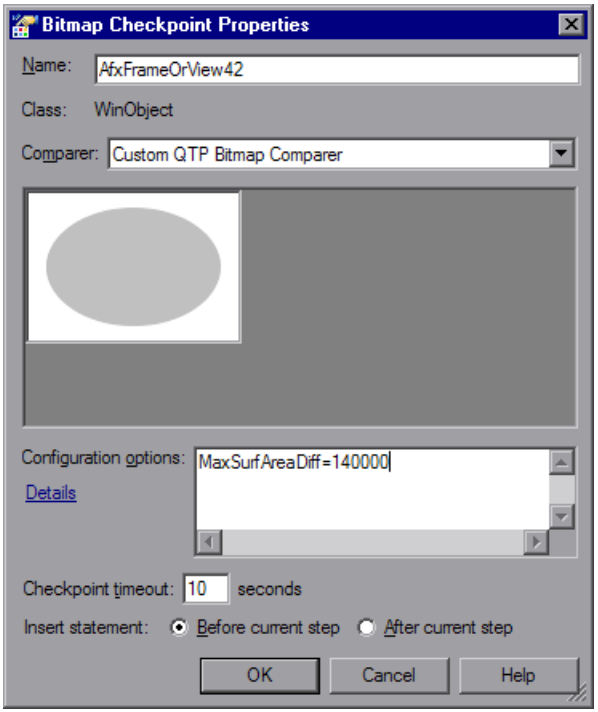
Instead, the following options are available (as supported by the custom comparer):

➤ **Configuration options.** Enables you to provide input (in string format) to the custom comparer, for any configuration options it supports. For example, you might be able to specify tolerance levels, an acceptable deviation in size or location of the bitmap, and so on.

By default, this box displays a configuration string provided by the custom comparer (if available).

➤ **Details.** Opens help information provided by the custom comparer (if available). This help can include instructions for providing configuration input to the comparer, information about the algorithm that the custom comparer uses to compare the bitmaps, an explanation about when to use this custom comparer, and so on.

Below is an image of the Bitmap Checkpoint Properties dialog box with a custom comparer selected:



20

Checking Tables

You can add table checkpoints to check the content of tables displayed in your application.

This chapter includes:

About Checking Tables

By adding table checkpoints to your test, you can check the content of tables displayed in your application. For example, you can check that a specified value is displayed in a certain cell. For some environments, you can also check the properties of the table object. For example, you can check that a table has the expected number of rows and columns.

When you run the test, the table checkpoint compares the actual data to the expected data, as defined in the checkpoint. If the results match, the checkpoint passes. You can view the results of the checkpoint in the Test Results window. For more information, see Chapter 33, "Viewing Run Session Results."

Table checkpoints are supported for table objects in a variety of add-in environments, such as Web, ActiveX, and Java. Table checkpoints are also supported for some list view objects, such as WinListView and VbListView.

Creating a Table Checkpoint

You can add a table checkpoint while recording or editing your test. To add a table checkpoint, you use the Table Checkpoint Properties dialog box.

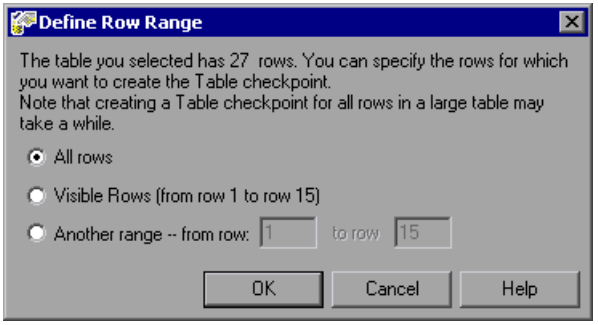
To add a table checkpoint while recording:

1 Select **Insert** > **Checkpoint** > **Standard Checkpoint** or click the **Insert Checkpoint or Output Value** button. The QuickTest window is hidden, and the pointer changes to a pointing hand. For more information on using the pointing hand feature, see "Tips for Using the Pointing Hand" on page 532.

2 Click the table you want to check. The Object Selection - Checkpoint Properties dialog box opens.

3 Select a table item from the displayed object tree and click **OK**. If the Table Checkpoint Properties dialog box opens, skip to step 4.

Otherwise, for certain objects in certain environments, the Define Row Range dialog box opens.



Select the range of rows you want to include in your checkpoint. You can include:

- **All rows.** Includes all of the rows in the table. Note that capturing all of the data for large table or list view objects may take some time.
- **Visible Rows (from row X to row Y).** Includes only the rows visible on the screen. Note that this option may not be available for some environments or object types.
- **Another range -- from row _ to row _.** You can specify any row range in the table.

Click **OK**. The Define Row Range dialog box closes, and the Table Checkpoint Properties dialog box displays the rows you specified (above the grid area).

4 In the Table Checkpoint Properties dialog box, specify the settings for the checkpoint. For more information, see "Understanding the Table Checkpoint Properties Dialog Box" on page 535.

Note: For some environments, the Table Checkpoint Properties dialog box contains two tabs: Table Content and Properties. For other environments, the Table Checkpoint Properties dialog box displays only the options available in the Table Content tab, but does not contain any tabs.

---

5 Click **OK** to close the dialog box. A checkpoint statement is added for the selected object in the Keyword View and Expert View.

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.

To add a table checkpoint while editing:

1 Depending on whether the object on which you want to perform a check is already in a step, do one of the following:

- If you already recorded a step on the object you want to check, right-click the step and select **Insert Standard Checkpoint**. Alternatively, select the step and select **Insert** > **Checkpoint** > **Standard Checkpoint**.

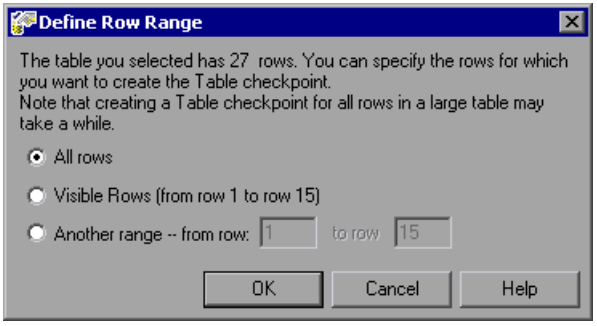
- If you have not recorded a step on the object you want to check, make sure the **Active Screen** button is selected and the Active Screen is visible. Click a step in your test where you want to add a checkpoint. The Active Screen displays the Web page or application screen corresponding to the highlighted step. Right-click the table in the Active Screen and select **Insert Standard Checkpoint**. The Object Selection - Checkpoint Properties dialog box opens. Select a table item from the displayed object tree and click **OK**.

Note: In some environments, you must have the table open in your application to insert a checkpoint on it.

---

2 If the Table Checkpoint Properties dialog box opens, skip to step 3.

Otherwise, for certain objects in certain environments, the Define Row Range dialog box opens.



Select the range of rows you want to include in your checkpoint. You can include:

- **All rows.** Includes all of the rows in the table. Note that capturing all of the data for large table or list view objects may take some time.
- **Visible Rows (from row X to row Y).** Includes only the rows visible on the screen. Note that this option may not be available for some environments or object types.
- **Another range -- from row X to row Y.** You can specify any row range between 1 and the number of rows listed in the table.

Click **OK**. The Define Row Range dialog box closes, and the Table Checkpoint Properties dialog box displays the rows you specified (above the grid area).

3 In the Table Checkpoint Properties dialog box, specify the settings for the checkpoint. For more information, see "Understanding the Table Checkpoint Properties Dialog Box" on page 535.

Note: For some environments, the Table Checkpoint Properties dialog box contains two tabs: Table Content and Properties. For other environments, the Table Checkpoint Properties dialog box displays only the options available in the Table Content tab, but does not contain any tabs.

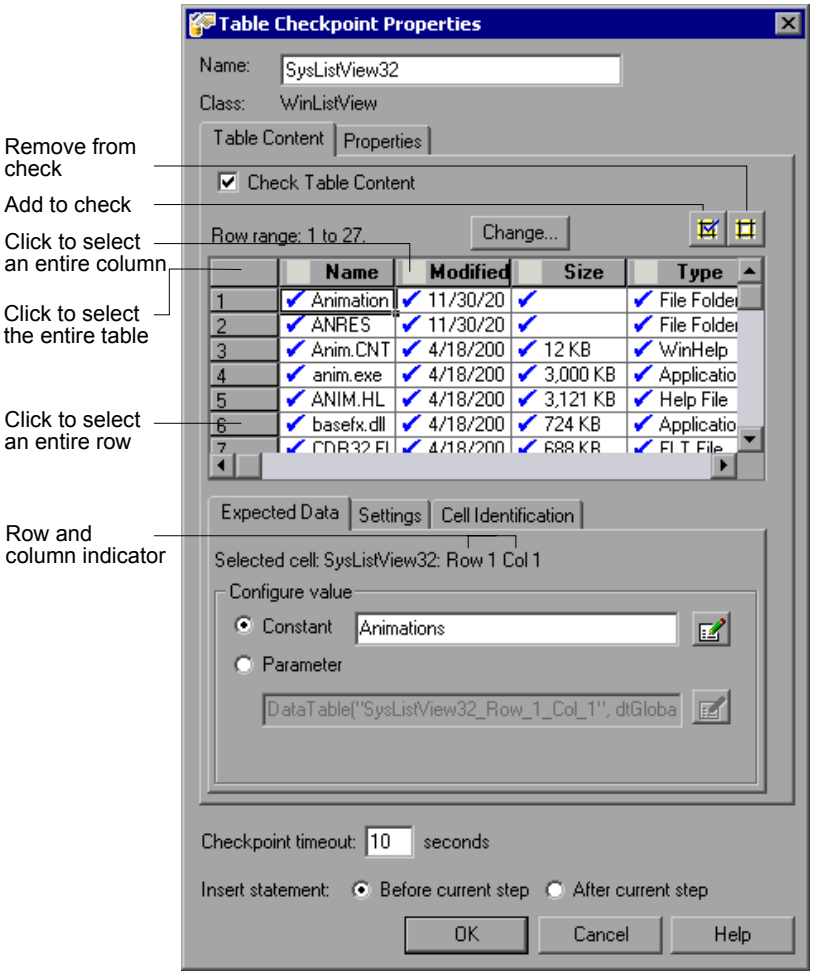
---

4 Click **OK** to close the dialog box. A checkpoint statement is added for the selected object.

Understanding the Table Checkpoint Properties Dialog Box

The Table Checkpoint Properties dialog box enables you to specify which cell contents of your table to check and which verification method and type to use. You can also edit or parameterize the expected data for the cells included in the check.

For some environments, the Table Checkpoint Properties Dialog Box also enables you to check the properties of the object (using the Properties tab), in addition to checking the content (using the Table Content tab).



Note: Some of the options shown in this example are available only in certain environments and only for certain objects.

For information on the options in the Table Content tab (or the entire dialog box if no tab is displayed), see the sections below. For information on the options in the Properties tab, see "Checking Table Properties" on page 546.

Checking Table Content

The Table Checkpoint Properties dialog box enables you to check table content.

Note: If the Table Checkpoint Properties dialog box contains tabs, you use the Table Content tab to check table content.

You can:

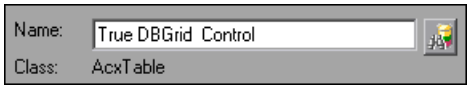
- understand and set general table checkpoint options
- specify which cells to check
- specify the expected data (Expected Data tab)
- specify the value type criteria (Settings tab)
- specify how QuickTest should locate the cells to be checked (Cell Identification tab)

Understanding and Setting General Table Checkpoint Options

This section describes the general settings and options displayed in the Table Checkpoint Properties dialog box. Most of the options described in this section are available regardless of whether the Table Checkpoint Properties dialog box contains tabs.

Descriptive Information

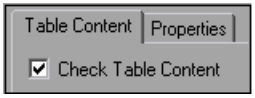
The top part of the Table Checkpoint Properties dialog box contains the following options:

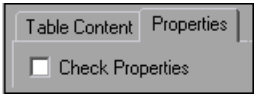


Name	The name that QuickTest assigns to the checkpoint. By default, the checkpoint name is the name of the test object on which the checkpoint is being performed. You can specify a different name for the checkpoint or accept the default name. If you rename the checkpoint, make sure that the name: ➤ is unique ➤ does not begin or end with a space ➤ does not contain " (double quotation mark) ➤ does not contain the following character combinations: := @@
Class	Specifies the type of object (read-only). This may be a table-type object or a list view-type object.
Find in Repository button	Displays the checkpoint in its object repository. **Note:** This option is available only when editing an existing checkpoint. It is not available when creating a new checkpoint.

Tabs (If Available)

If the Table Checkpoint Properties dialog box contains tabs, each tab displays a check box. You can select one or both of these check boxes to specify the type of data to check.



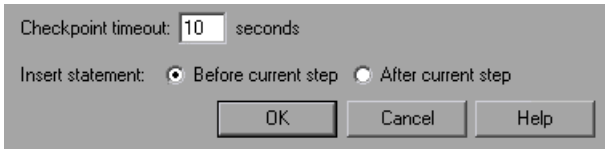


Check Table Content check box	(Table Content tab) Selecting the **Check Table Content** check box instructs QuickTest to check the content of the table object. (Selected by default.)
Check Properties check box	(Properties tab) Selecting the **Check Properties** check box instructs QuickTest to check the properties of the table object. (Cleared by default.)

Note: These check boxes are displayed only if the Table Checkpoint Properties dialog box contains tabs. If the Table Checkpoint Properties dialog box does not contain tabs, QuickTest automatically checks table content as defined in the dialog box.

Timeout and Statement Location

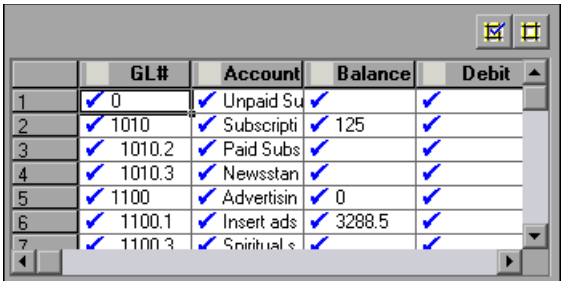
The bottom part of the Table Checkpoint Properties dialog box contains the following options:



Checkpoint timeout	Specifies the time interval (in seconds) during which QuickTest attempts to perform the checkpoint successfully. QuickTest continues to perform the checkpoint until the checkpoint passes or until the timeout occurs. If the checkpoint does not pass before the timeout occurs, the checkpoint fails. For example, if it takes a long time for data to load in a table, increasing the checkpoint timeout value can help ensure that the data has sufficient time to load. This enables the checkpoint to pass (if the data matches) before the end of the timeout period is reached. If you specify a checkpoint timeout other than **0**, and the checkpoint fails, the Test Results window displays information on the checkpoint timeout.
Insert statement	Specifies when to perform the checkpoint in the test. Select **Before current step** if you want to check the table content before the highlighted step is performed. Select **After current step** if you want to check the table content after the highlighted step is performed. **Note:** The **Insert statement** option is available only when adding a new checkpoint while editing an existing test. (This option is not available during recording.)

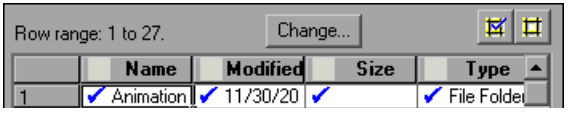
Specifying Which Cells to Check

The grid area of the Table Checkpoint Properties dialog box represents the cells in the table. The column header names are captured from the table you selected for your checkpoint.



Tip: You can change the column widths and row heights of the grid by dragging the column and row header dividers.

Note: Some environments and objects support selecting a row range. This enables you to specify which rows are displayed in the grid area. If row range selection is supported, the row range you specify when creating the checkpoint is displayed above the grid:



Clicking the **Change** button enables you to modify the row range. For more information, see “Modifying a Table Checkpoint” on page 548.

When you create a new table checkpoint, all cells contain a blue check mark, indicating they are all selected for verification. You can instruct QuickTest to check the entire table, specific rows, specific columns, or specific cells. QuickTest checks only cells containing a check mark.

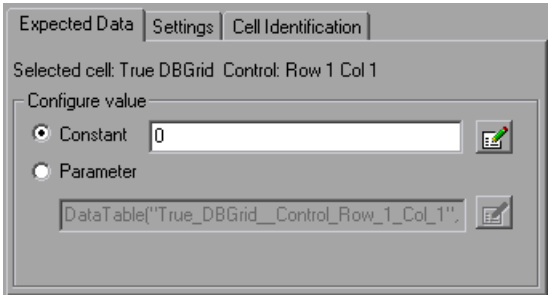
To:	Do this:
Add **a single cell** to or remove it from the check	Double-click the cell
Add **an entire row** to or remove it from the check	Double-click the row header
Add **an entire column** to or remove it from the check	Double-click the column header
Add **all cells** to or remove all cells from the check	Double-click the top-left corner of the grid
Add **a range of cells** to the check	Select the cells to add to the check and click the **Add to Check** button 
Remove **a range of cells** from the check	Select the cells to remove from the check and click the **Remove from Check** button 

Notes:

- Double-clicking on the grid toggles the settings for all selected cells. Therefore, if you double-click a row header, a column header, or the top left corner of the grid, any cells that were previously included in the check are removed from it, and any cells that were not previously included in the check are added to it.
- When more than one cell is selected, the options in the Expected Data tab are disabled.

Specifying the Expected Data

The Expected Data tab displays options for setting the expected value of the selected cell in the table.



You can modify the value of a cell or you can parameterize it to use a value from an external source, such as the Data Table or an environment variable. During the run session, QuickTest compares the value specified in this tab with the actual value that it finds during the run session. If the expected value and the actual value do not match, the checkpoint fails.

To modify or parameterize several cells in the table, select a cell and then set your preferences for that cell in the Expected Data tab. Repeat the process for each cell you want to modify.

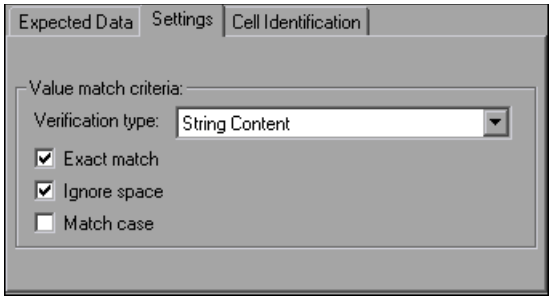
The Expected Data tab includes the following:

Selected cell	Indicates the table name and the row and column numbers of the selected cell.
Configure value	Enables you to set the expected value of the cell as a constant or parameter. For more information on modifying values, see “Setting Values in the Configure Value Area” on page 757.

Note: When more than one cell is selected, the options in the Expected Data tab are disabled.

Specifying the Value Type Criteria

The Settings tab includes options that determine how the actual cell values are compared with the expected cell values. The settings in this tab apply to all selected cells.



The default setting is to treat cell values as strings and to check for the exact text, while ignoring spaces.

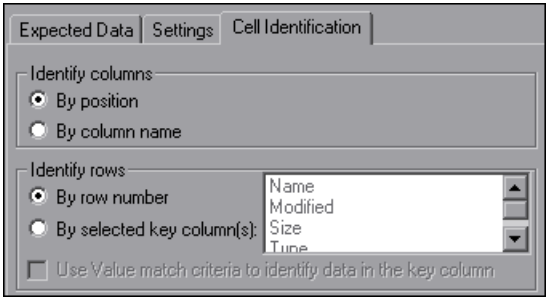
The Settings tab includes the following options:

Option	Description
Verification type	Specifies how cell contents are compared: ➤ **String Content.** (Default) Evaluates the content of the cell as a string. For example, 2 and 2.00 are not recognized as the same string. ➤ **Numeric Content.** Evaluates the content of the cell according to numeric values. For example, 2 and 2.00 are recognized as the same number. ➤ **Numeric Range.** Compares the content of the cell against a numeric range, where the minimum and maximum values are any real number that you specify. This comparison differs from string and numeric content verification in that the table data is compared against the range that you defined and not against a specific expected value.

Option	Description
Exact match	(Default) Checks that the exact text, and no other text, is displayed in the cell. Clear this check box if you want to check that a value is displayed in a cell as part of the contents of the cell. **Note:** QuickTest displays this option only when **String Content** is selected as the **Verification type**.
Ignore space	(Default) Ignores spaces in the captured content when performing the check. The presence or absence of spaces does not affect the outcome of the check. **Note:** QuickTest displays this option only when **String Content** is selected as the **Verification type**.
Match case	Conducts a case sensitive search. **Note:** QuickTest displays this option only when **String Content** is selected as the **Verification type**.
Min / Max	Specifies the numeric range against which the content of the cell is compared. The range values can be any real number. **Note:** QuickTest displays this option only when **Numeric Range** is selected as the **Verification type**.

Specifying the Cell Identification Settings

The settings in the Cell Identification tab determine how QuickTest locates the cells to be checked. The settings in this tab apply to all selected cells.

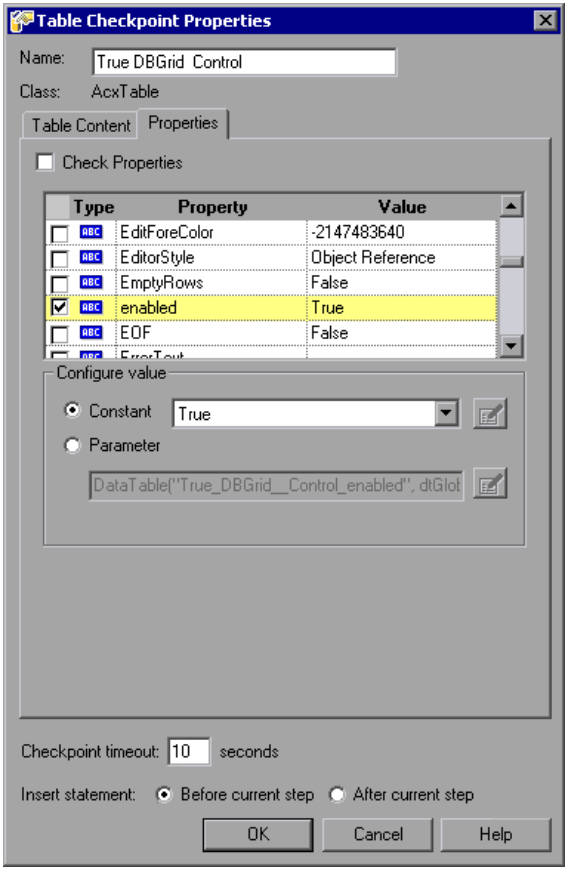


The Cell Identification tab includes the following options:

Identify columns	Specifies the location of the column (in your actual table) containing the cell(s) to which you want to compare the expected data. ➤ **By position.** (Default) Locates cells according to the column position. A shift in the position of the columns within the table results in a mismatch. ➤ **By column name.** Locates cells according to the column name. A shift in the position of the columns within the table does not result in a mismatch. (Enabled only when the table contains more than one column.)
Identify rows	Specifies the location of the row (in your actual table) containing the cell(s) to which you want to compare the expected data. ➤ **By row number.** (Default) Locates cells according to the row position. A shift in the position of any of the rows within the table results in a mismatch. ➤ **By selected key column(s).** Locates the row(s) containing the cells to be checked by matching the value of the cell whose column was previously selected as a **key column**. A shift in the position of the row(s) does not result in a mismatch. If more than one row is identified, QuickTest checks the first matching row. You can use more than one key column to uniquely identify any row. **Note:** A key symbol  is displayed in the header of selected key columns.
Use value match criteria to identify data in the key column	Instructs QuickTest to use the verification type settings from the Settings tab as the criteria for identifying data in the key column. Enabled only when you select to identify rows **By selected key column(s)**.

Checking Table Properties

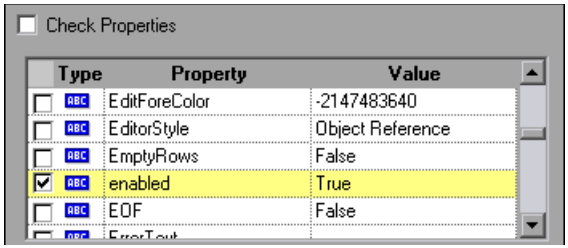
For certain environments, you can specify which table (or grid) properties you want to check. By default, when you create a table checkpoint on an object, QuickTest captures all the object's properties, but does not select any properties to check.



Note: For information on general table checkpoint options, such as **Name** and **Checkpoint timeout**, see "Understanding and Setting General Table Checkpoint Options" on page 537.

Selecting Properties to Check

When you create a table checkpoint, the Properties pane displays the table object's default properties, including the properties, their values, and their types.



You instruct QuickTest to perform a properties check by selecting the **Check Properties** check box. (This check box is cleared by default.)

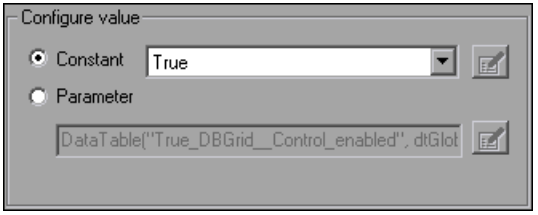
The Properties pane for the object contains the following:

Check box	For each object class, QuickTest recommends default property checks. You can accept the default checks or modify them accordingly. ➤ To check a property, select the corresponding check box. ➤ To remove a property from the check, clear the corresponding check box.
Type	The ABC icon indicates that the value of the property is currently a constant. The icon indicates that the value of the property is currently a test or action parameter. The icon indicates that the value of the property is currently a Data Table parameter. The icon indicates that the value of the property is currently an environment variable parameter. The icon indicates that the value of the property is currently a random number parameter.

Property	The name of the property.
Value	The expected value of the property. For more information on modifying the value of a property, see "Setting Values in the Configure Value Area" on page 757.

Editing the Expected Value of a Table Property

The **Configure value** area enables you to define the expected value of the property as a **Constant** or a **Parameter**.



For information on modifying property values, see "Setting Values in the Configure Value Area" on page 757.

Modifying a Table Checkpoint

You can change the expected data, settings and cell identification options for an existing table checkpoint.

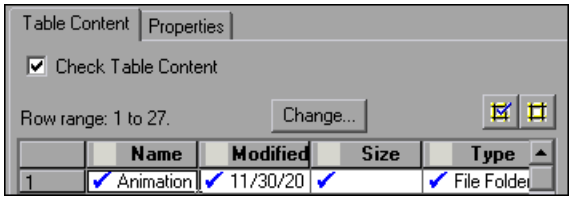
To modify the settings of the table checkpoint:

1. In the Keyword View or Expert View, right-click the table checkpoint that you want to modify and select **Checkpoint Properties**. Alternatively, select the step containing the checkpoint and select **Edit > Step Properties > Checkpoint Properties**. The Table Checkpoint Properties dialog box opens.
2. Modify the settings as described in "Understanding the Table Checkpoint Properties Dialog Box" on page 535.

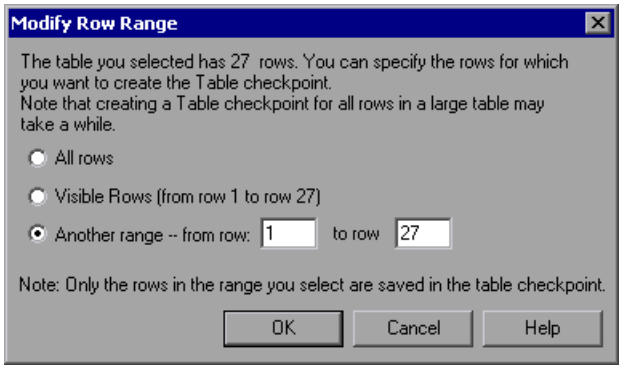
To modify the number of rows in an existing table checkpoint:

1 Open the application containing the table or list view object you want to check and display the object in the application.

2 In the Keyword View or Expert View, right-click the table checkpoint that you want to modify and select **Checkpoint Properties**. Alternatively, select the step containing the checkpoint and select **Edit** > **Step Properties** > **Checkpoint Properties**. The Table Checkpoint Properties dialog box opens, displaying the currently selected row range.



3 In the Table Content tab, click the **Change** button at the top of the dialog box (above the grid area). The Modify Row Range dialog box opens.



4 Select the range of rows you want to include in your checkpoint. You can include all the rows, only the visible rows, or another range that you specify.

Note: The **Visible Rows** option may not be available for some environments or object types.

---

5 Click **OK**. The Modify Row Range dialog box closes, and the Table Checkpoint Properties dialog box displays the rows you specified in the Modify Row Range dialog box.

- If your modified row range includes new rows, QuickTest captures the current values of the new rows from the open application.
- If your modified row range includes some or all of the rows that were already included in the checkpoint, the expected values of those cells are not changed. This enables you to modify the row range without losing parameterization, regular expressions, or other changes you may have made to the expected cell values in your checkpoint.

  Therefore, you cannot use the Modify Row Range dialog box to update the expected values of an existing table checkpoint. To update the expected values of your checkpoint, use the **Update Run Mode** option. For more information, see "Updating a Test Using the Update Run Mode Option" on page 1125.
- If your modified row range excludes some or all of the rows that were previously included in your checkpoint, those rows (and any modifications you made to the expected values) are deleted from the checkpoint.

21

Checking Text

QuickTest can check that a text string is displayed in the appropriate place in an application.

This chapter includes:

About Checking Text

You can check that a specified text string is displayed by adding one of the following checkpoints to your test.

- **Standard Checkpoint.** Enables you to check the **text** property of an object. You can use standard checkpoints to check text in Windows-based and other types of applications (including Web-based applications). For more information on standard checkpoints, see "Creating Standard Checkpoints" on page 506.
- **Text Area Checkpoint.** Enables you to check that a text string appears within a defined area in a Windows application, according to specified criteria. It is supported for a variety of QuickTest add-in environments, such as Standard Windows, Java, Visual Basic, and ActiveX. For more information, see the *HP QuickTest Professional Add-ins Guide*.

➤ **Text Checkpoint.** Enables you to check that the text is displayed in a screen, window, or Web page, according to specified criteria. Text checkpoints are supported for many QuickTest add-in environments (as listed in "Supported Checkpoints" on page 504). For more information, see the *HP QuickTest Professional Add-ins Guide*.

When checking text, QuickTest tries to retrieve the text directly from the object. If QuickTest cannot retrieve the text in this manner (for example, because the text is part of a picture), it tries to retrieve the text using an OCR (optical character recognition) mechanism. The OCR mechanism translates images of handwritten or typewritten text into machine-editable text.

Creating a Text Checkpoint

You can add a text checkpoint while recording or editing steps in a Windows- or Web-based application.

Note: Before you create a text checkpoint, make sure you configure the required capture settings in the General > Text Recognition pane (**Tools** > **Options** > **Text Recognition** node). For more information, see "The Options Dialog Box: General > Text Recognition Pane" on page 742 and "About Working with Text Recognition for Windows-Based Objects" on page 742.

To add a text checkpoint while recording:

1 Display the page, window, or screen containing the text you want to check.

2 Select **Insert** > **Checkpoint** > **Text Checkpoint**, or click the **Insert Checkpoint or Output Value** toolbar button and select **Text Checkpoint**.

The QuickTest window is hidden, and the pointer changes into a pointing hand. For more information about using the pointing hand feature, see "Tips for Using the Pointing Hand" on page 553.

3 Click the text string for which you want to create the checkpoint. The Text Checkpoint Properties dialog box opens.

4 Specify the checkpoint settings. For more information, see "The Text / Text Area Checkpoint Properties Dialog Box" on page 557.

5 Click **OK** to close the dialog box. A checkpoint statement is added for the selected object.

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.

To add a text checkpoint while editing a test:

1 Make sure the **Active Screen** toolbar button is selected.

2 Click the step where you want to add a checkpoint. The Active Screen displays the page or screen corresponding to the highlighted step.

3 Highlight a text string on the Active Screen.

4 Right-click the text string and select **Insert Text Checkpoint**. The Text Checkpoint Properties dialog box opens.

5 Specify the settings for the checkpoint. For more information, see "The Text / Text Area Checkpoint Properties Dialog Box" on page 557.

6 Click **OK** to close the dialog box. A checkpoint statement is added for the selected object.

Creating a Text Area Checkpoint

You can add a text area checkpoint only while recording a test on Windows-based applications, such as Standard Windows, Java, Visual Basic, and ActiveX. To determine whether text area checkpoints are supported for a specific QuickTest add-in environment, see the *HP QuickTest Professional Add-ins Guide*.

Note: Before you create a text area checkpoint, make sure you configure the required capture settings in the General > Text Recognition pane (**Tools** > **Options** > **Text Recognition** node). For more information, see "The Options Dialog Box: General > Text Recognition Pane" on page 742 and "About Working with Text Recognition for Windows-Based Objects" on page 742.

To add a text area checkpoint:

1 Select **Insert** > **Checkpoint** > **Text Area Checkpoint**, or click the arrow next to the **Insert Checkpoint** toolbar button and select **Text Area Checkpoint**.

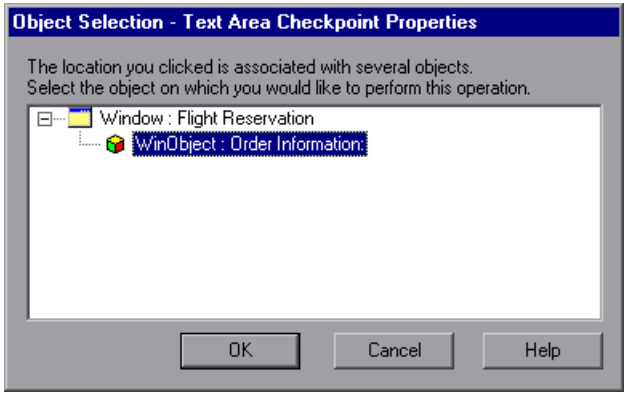
The QuickTest window is hidden, and the mouse pointer turns into a crosshairs pointer.

2 Define the area containing the text you want QuickTest to check by clicking and dragging the crosshairs pointer. (See "Considerations for Defining the Text Area" on page 556.)

Tip: Hold down the left mouse button and use the arrow keys to make precise adjustments to the defined area.

Release the mouse button after outlining the area required.

If the area you defined is associated with more than one object, the Object Selection–Text Area Checkpoint Properties dialog box opens.



3 Select the object for which you are creating the checkpoint. The Text Area Checkpoint Properties dialog box opens.

4 Specify the checkpoint settings. For more information, see "The Text / Text Area Checkpoint Properties Dialog Box" on page 557.

5 Click **OK** to close the dialog box. A checkpoint statement is added for the selected object in the Keyword View and Expert View.

Considerations for Defining the Text Area

When checking text displayed in a Windows-based application, it is often advisable to define a text area larger than the actual text you want QuickTest to check. You then use the Text Area Checkpoint Properties dialog box to configure the relative position of the Checked Text within the captured string. When QuickTest runs the test, it checks for the selected text within the defined area, according to the settings you configured.

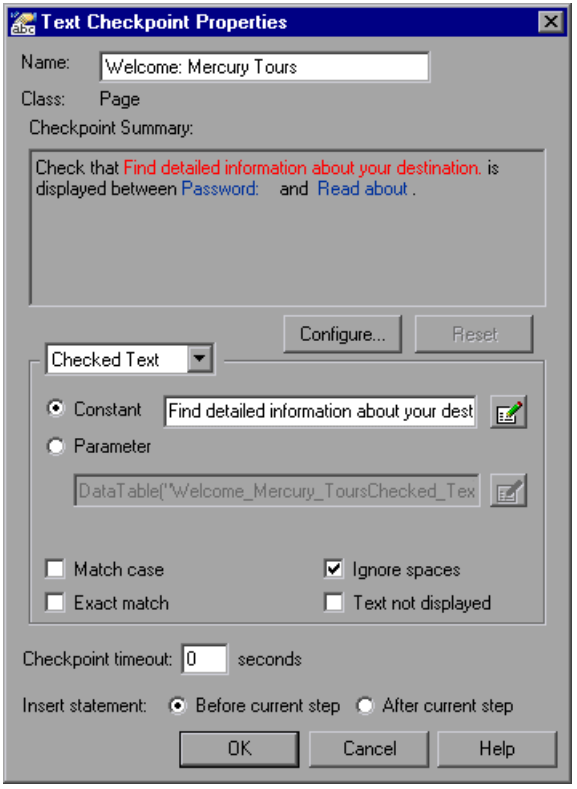
Consider the following when defining the area for a text area checkpoint:

- If you parameterize a text string, the captured area must be large enough to accommodate any string that might replace the one selected during a run session.
- The captured area must be large enough to include all parts of the required text (**Checked Text / Text Before / Text After**).
- Text may change its position during run sessions; therefore, make sure that the area you capture is large enough to allow for acceptable position shifts. If the defined area is too small, even a slight shift in the text's position will cause the run to fail, although the changed position may be acceptable to you. If, on the other hand, the position of the text on the screen is critical, or if you do not want it to exceed certain boundaries, set the defined area accordingly.

The Text / Text Area Checkpoint Properties Dialog Box

Description	Enables you to specify the text to be checked, as well as specify which text is displayed before and after the checked text. These configuration options are particularly helpful when the text string you want to check appears several times or when it could change in a predictable way during run sessions. For example, suppose you want to check the third occurrence of a particular text string in a page. To check for this string, you can specify which text precedes and/or follows it and to which occurrence of the specified text string you are referring.
How to Access	➤ "Creating a Text Checkpoint" on page 552 ➤ "Creating a Text Area Checkpoint" on page 554 ➤ "Modifying a Text or Text Area Checkpoint" on page 570
Important Information	Before you create a text or text area checkpoint, make sure you set the required options, as described in "The Options Dialog Box: General > Text Recognition Pane" on page 742.
Learn More	**Conceptual overview**: "About Checking Text" on page 551 **Primary tasks**: ➤ "Creating a Text Checkpoint" on page 552 ➤ "Creating a Text Area Checkpoint" on page 554 ➤ "Modifying a Text or Text Area Checkpoint" on page 570 **Additional related topics**: "Use-Case Scenario: Checking Text in an Image" on page 750

Below is an image of the Text/Text Area Checkpoint Properties dialog box:



This image is an example of the Text Checkpoint Properties dialog box that opens when adding a text checkpoint to an existing test during an editing session. The Text Checkpoint Properties dialog box options differ slightly during a recording session or when editing an existing checkpoint. The Text Area Checkpoint Properties dialog box is similar to the Text Checkpoint Properties dialog box.

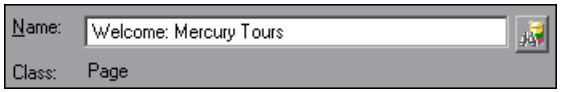
Text/Text Area Checkpoint Properties Dialog Box Options

Information	Description
General checkpoint information area	Enables you to view and specify the name of the checkpoint and to view the type of object. See: "Understanding and Setting General Text Checkpoint Information" on page 561.
Checkpoint Summary area	Summarizes the selected text for the checkpoint. It displays the text you selected when creating the checkpoint, plus the text before and after it. QuickTest automatically displays the checked text in red, and the text before and after the checked text in blue. For text checkpoints in Web-based environments, it displays the text you selected when creating the checkpoint, plus some text before and after it. For text and text area checkpoints in Windows-based environments, it displays the text you selected when creating the checkpoint. **Note:** In Windows-based environments, if there is more than one line of text selected, the Checkpoint Summary area displays **[complex value]** instead of the selected text string. You can then click **Configure** to view and manipulate the actual selected text for the checkpoint. You can designate parts of the captured string as **Checked Text** and other parts as **Text Before** and **Text After** by clicking the **Configure** button. For more information, see "Configuring the Text Selection" on page 561.

Information	Description
Options for checked text area	Set parameterization and other preferences for each of the string elements in your checkpoint by selecting the string element type (**Checked Text** / **Text Before** / **Text After**) from the list box and selecting your preferences. See: ➤ "Setting Options for Checked Text" on page 564 ➤ "Setting Options for Text Displayed Before the Checked Text" on page 566 ➤ "Setting Options for Text Displayed After the Checked Text" on page 567
Checkpoint timeout and Insert Statement area	Specify when QuickTest should perform the checkpoint by specifying the timeout and location of the checkpoint. These options are available only when inserting a new checkpoint See "Setting Checkpoint Timeout and Statement Location Options" on page 569.

Understanding and Setting General Text Checkpoint Information

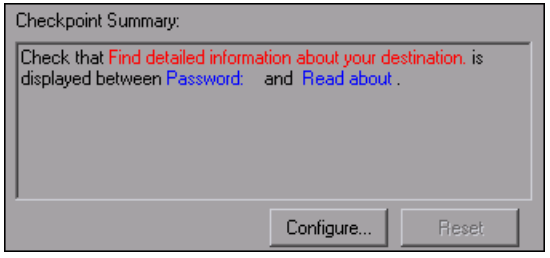
The top part of the Text/Text Area Checkpoint Properties dialog box contains the following options:



Name	The name that QuickTest assigns to the checkpoint. By default, the checkpoint name is the name of the test object on which the checkpoint is being performed. You can specify a different name for the checkpoint or accept the default name. If you rename the checkpoint, make sure that the name: ➤ is unique ➤ does not begin or end with a space ➤ does not contain " (double quotation mark) ➤ does not contain the following character combinations: := @@
Class	Specifies the type of object (read-only).
Find in Repository button	Displays the checkpoint in its object repository. **Note:** This option is available only when editing an existing checkpoint. It is not available when creating a new checkpoint.

Configuring the Text Selection

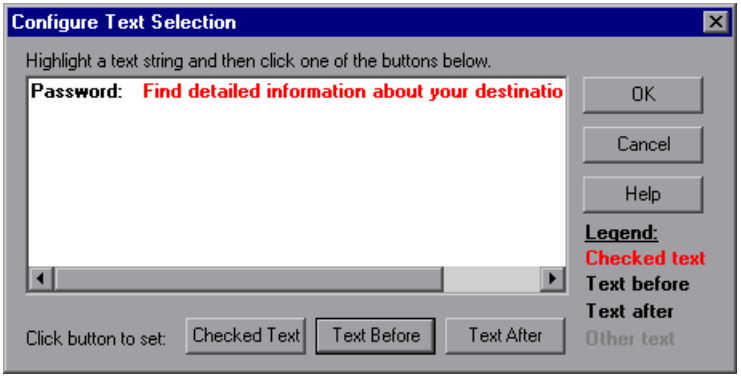
You can view and modify the text selection displayed in the Checkpoint Summary area.



The Checkpoint Summary area contains the following options:

Option	Description
Configure	Opens the Configure Text Selection dialog box, where you can specify the checked text, the text before (if any), and the text after (if any).
Reset	Resets the text selection to the previous configuration.

To designate **Checked Text**, **Text Before**, and **Text After**, you use the Configure Text Selection dialog box (opened by clicking the **Configure** button). The Configure Text Selection dialog box displays the text you captured when creating the text checkpoint, as well as text before and after the selected text. QuickTest displays the checked text in red and the text before and after it in black (as indicated in the **Legend** displayed in the dialog box).



To modify which text is checked and how that text is found, based on the text before and after it, highlight the text you want to set for one of these items and then click the appropriate button.

Option	Description
Checked Text	Sets the highlighted text as the checked text. QuickTest displays this text in red and the remainder in black.
Text Before	Sets the highlighted text as the text before the checked text.
Text After	Sets the highlighted text as the text after the checked text.

Note: If no text is selected in the Configure Text Selection dialog box, then nothing happens when you click these buttons.

To remove text from the current text selection configuration, highlight only the text you want included as the before or after text and click the appropriate button. Any text that is not selected as **Checked Text**, **Text Before**, or **Text After** is displayed in gray. The gray text is not displayed the next time the Configure Text Selection dialog box is opened.

For example, in the sample image above, if you wanted to check only the word information, and you wanted QuickTest to look for this text between Find detailed and about your destination, then:

- Highlight the word information, and click **Checked Text**. The word information remains red, and the other text turns black.
- Highlight the words Find detailed and click **Text Before**. The words Find detailed remain black, and all text preceding it turns gray. This gray text will be removed from the text configuration when you click **OK**.
- The words about your destination are already marked in black as the text after, so there is no need to modify this configuration.

Note: If you want to configure more text than is displayed, you must cancel the text checkpoint and select a larger text area or selection in your application.

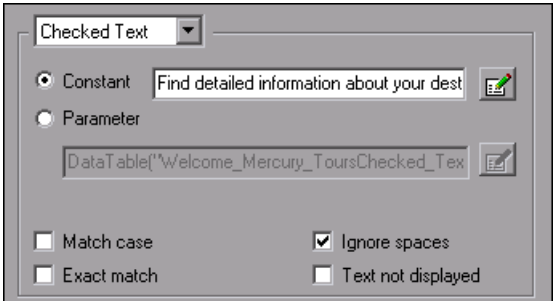
When you close the Configure Text Selection dialog box, the Checkpoint Summary area displays the new text selection configuration.

Setting Options for the Text to be Checked

The middle area of the Text/Text Area Checkpoint Properties dialog box enables you to set options for the checked text, text before, and text after, as described in the following sections.

Setting Options for Checked Text

You set options for the checked text by choosing **Checked Text** from the list box. In the Checked Text area, you can indicate whether you want the checked text to be a constant or a parameter, and you can set the criteria for a successful match.



You can choose from the following options for the checked text:

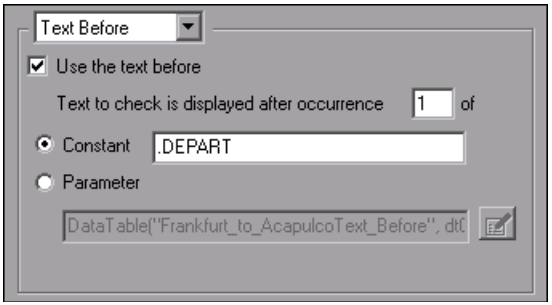
- **Constant.** (Default) Sets the expected value of the checked text as a constant. For information on modifying values, see "Setting Values in the Configure Value Area" on page 757.

Tip: The **Constant** box displays the checked text. You can change the checked text by typing in the **Constant** box or by using the Configure Text Selection dialog box.

- **Parameter.** Sets the expected value of the checked text as a parameter. For information on modifying values, see "Setting Values in the Configure Value Area" on page 757.
- **Match case.** Conducts a case-sensitive check.
- **Exact match.** Checks for the exact expected text. For example, if you create a checkpoint with the following description, Check that New York is displayed between Flight departing from and to San Francisco, and select **Exact match**, if the actual text is New York City, the checkpoint fails. If you do not select **Exact match**, the checkpoint passes because the expected text is contained within the actual text.
- **Ignore spaces.** Ignores spaces in the captured text when performing the check. The presence or absence of spaces does not affect the outcome of the check.
- **Text not displayed.** Checks that the text string is not displayed. For example, if you create a checkpoint with the following description, Check that New York is displayed between Flight departing from and to San Francisco, and select **Text not displayed**, QuickTest checks that the text New York is not displayed.

Setting Options for Text Displayed Before the Checked Text

You set options for the text displayed before the checked text by choosing **Text Before** from the list box. In the Text Before area, you can set the text before the checked text as a constant or a parameter.



You can choose from the following options when setting the text displayed before the checked text:

- **Use the text before.** Checks the text before the checked text. To ignore this text, clear this check box.
- **Text to check is displayed after occurrence.** Checks that the checked text is displayed after the specified text.

  If the identical text string you specify is displayed more than once on the page, you can specify the occurrence of the string to which you are referring.

  If you accept the default text that QuickTest recommends, the number in the dialog box will be correct. If you modify the text, confirm that the occurrence number is accurate.

  If you choose a non-unique text string, change the occurrence number appropriately. For example, if you want to check that the words Mercury Tours are displayed after the fourth occurrence of the word the, enter 4 in the **Text to check is displayed after occurrence** box.
- **Constant.** (Default) Sets the expected value of the text before the checked text as a constant. For information on modifying values, see "Setting Values in the Configure Value Area" on page 757.

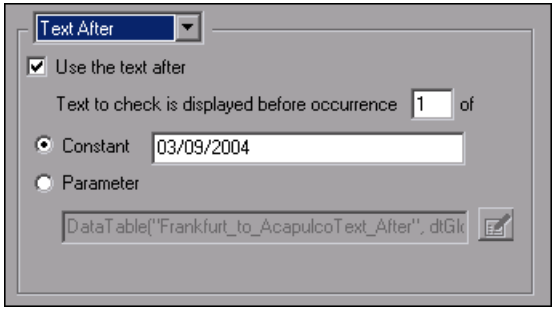
  If you modify the text, whenever possible, use a string that is unique within the object so that the occurrence number is 1.

Tip: The **Constant** box displays the text before the checked text. You can change the text by typing in the **Constant** box or by using the Configure Text Selection dialog box.

➤ **Parameter.** Sets the expected value of the text before the checked text as a parameter. For information on modifying values, see "Setting Values in the Configure Value Area" on page 757.

Setting Options for Text Displayed After the Checked Text

You set options for or the text displayed after the checked text by choosing **Text After** from the list box. In the Text After area, you can set the text after the checked text as a constant or a parameter.



You can choose from the following options when setting the text displayed after the checked text:

➤ **Use the text after.** Checks the text after the checked text. To ignore this text, clear this check box.

➤ **Text to check is displayed before occurrence.** Checks that the checked text is displayed before the specified text. If the identical text string you specify is displayed more than once on the page, you can specify to which occurrence of the string you are referring.

If you accept the default text that QuickTest recommends, the number in the dialog box will be correct. If you modify the text, confirm that the occurrence number is also accurate.

If you choose a non-unique text string, change the occurrence number appropriately. For example, if you want to check that the words Mercury Tours are displayed before the fourth occurrence of the word the, enter 4 in the **Text to check is displayed before occurrence** box.

➤ **Constant.** (Default) Sets the expected value of the text after the checked text as a constant. For information on modifying values, see "Setting Values in the Configure Value Area" on page 757.

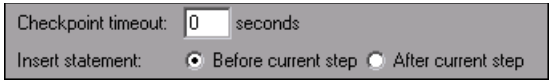
If you modify the text, whenever possible, use a string that is unique within the object so that the occurrence number is 1.

Tip: The **Constant** box displays the text after the checked text. You can change the text by typing in the **Constant** box or by using the Configure Text Selection dialog box.

➤ **Parameter.** Sets the expected value of the text after the checked text as a parameter. For information on modifying values, see "Setting Values in the Configure Value Area" on page 757.

Setting Checkpoint Timeout and Statement Location Options

You can specify the time interval during which QuickTest attempts to perform the checkpoint successfully by modifying the selections in the bottom part of the Text/Text Area Checkpoint Properties dialog box. You can also specify when to perform the checkpoint.



- **Checkpoint timeout.** Specifies the time interval (in seconds) during which QuickTest attempts to perform the checkpoint successfully. QuickTest continues to perform the checkpoint until it passes or until the timeout occurs. If the checkpoint does not pass before the timeout occurs, the checkpoint fails.

  For example, suppose it takes some time for an object to achieve an expected state. Increasing the checkpoint timeout value in this case can help ensure that the object has sufficient time to achieve that state, enabling the checkpoint to pass (if the data matches) before the maximum timeout is reached.

  If you specify a checkpoint timeout other than **0**, and the checkpoint fails, the Test Results window displays information on the checkpoint timeout.

- **Insert statement.** Specifies when to perform the checkpoint. Select **Before current step** if you want to check the value of the text before the highlighted step is performed. Select **After current step** if you want to check the value of the text after the highlighted step is performed.

Note: The **Insert statement** option is not available when adding a new text checkpoint or a text area checkpoint during recording, or when modifying an existing checkpoint. It is available only when adding a new text checkpoint to an existing test while editing.

Modifying a Text or Text Area Checkpoint

You can modify an existing text or text area checkpoint.

To modify a text or text area checkpoint:

1. In the Keyword View or Expert View, right-click the checkpoint that you want to modify and select **Checkpoint Properties**. Alternatively, select the step containing the checkpoint and select **Edit** > **Step Properties** > **Checkpoint Properties**. The Text/Text Area Checkpoint Properties dialog box opens.
2. Modify the settings. For more information, see "The Text / Text Area Checkpoint Properties Dialog Box" on page 557.

Creating a Standard Checkpoint for Checking Text

You can check the text property of an object in Windows-based and other types of applications (including Web-based applications) by using a standard checkpoint.

To add a standard checkpoint for checking text while recording:

1. Select **Insert** > **Checkpoint** > **Standard Checkpoint** or click the **Insert Checkpoint or Output Value** toolbar button and select **Standard Checkpoint**. The QuickTest window is hidden, and the pointer changes into a pointing hand. For more information about using the pointing hand feature, see "Tips for Using the Pointing Hand" on page 573.
2. Click the object whose text you want to check. The Object Selection - Checkpoint Properties dialog box opens.
3. Select the item you want to check from the displayed object tree.
4. Click **OK**. The Checkpoint Properties dialog box opens.
5. In the **Name** box, either accept the name that QuickTest assigns to the checkpoint or specify another name for it. By default, the checkpoint name is the name of the test object on which the checkpoint is being performed.

If you rename the checkpoint, make sure that the name:

- is unique
- does not begin or end with a space
- does not contain " (double quotation mark)
- does not contain the following character combinations:
  :=
  @@

Note: The **Class** area displays the type of test object on which the checkpoint is being performed.

---

6. Select the **text** property.
7. If necessary, edit the **text** value you want QuickTest to check. Note that you can parameterize this value.
8. If you want to check only text, clear the other check boxes in the dialog box.
9. Click **OK** to close the dialog box. A checkpoint statement is added for the selected object.

To add a standard checkpoint for checking text while editing:

1 Right-click the step for the object whose text you want to check, and select **Insert Standard Checkpoint**. The Checkpoint Properties dialog box opens.

2 In the **Name** box, either accept the name that QuickTest assigns to the checkpoint or specify another name for it. By default, the checkpoint name is the name of the test object on which the checkpoint is being performed.

If you rename the checkpoint, make sure that the name:

- is unique
- does not begin or end with a space
- does not contain " (double quotation mark)
- does not contain the following character combinations:
  :=
  @@

Note: The **Class** area displays the type of test object on which the checkpoint is being performed.

---

3 Select the **text** property.

4 If necessary, edit the text value you want QuickTest to check. Note that you can parameterize this value.

5 If you want to check only text, clear the other check boxes in the dialog box.

6 Click **OK** to close the dialog box. A checkpoint statement is added for the selected object.

For more information on creating standard checkpoints, see Chapter 18, "Checking Object Property Values Using Standard Checkpoints."

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.

22

Checking Databases

By adding database checkpoints, you can check the contents of databases accessed by your application. Database checkpoints are supported by all environments.

This chapter includes:

About Checking Databases

You can use database checkpoints in your test to check databases accessed by your application, and to detect defects. To do this, you define a query on your database. Then you create a database checkpoint that checks the results of the query.

You can define a database query in the following ways:

- Using Microsoft Query. You can install Microsoft Query from the custom installation of Microsoft Office.
- By manually defining an SQL statement.

Creating a Check on a Database

You create a database checkpoint based on the results of the query (**result set**) you defined on a database. You can create a check on a database to check the contents of the entire result set, or a part of it. QuickTest captures the current data from the database, saves this information as **expected data**, and inserts a database checkpoint into the test. This checkpoint is displayed in the Expert View as a DbTable.Check CheckPoint statement and as a step in the Keyword View, as follows:

When you run the test, the database checkpoint compares the current data in the database to the expected data defined in the Database Checkpoint Properties dialog box. If the expected data and the current results do not match, the database checkpoint fails. You can view the results of the checkpoint in the Test Results window. For more information, see Chapter 33, "Viewing Run Session Results."

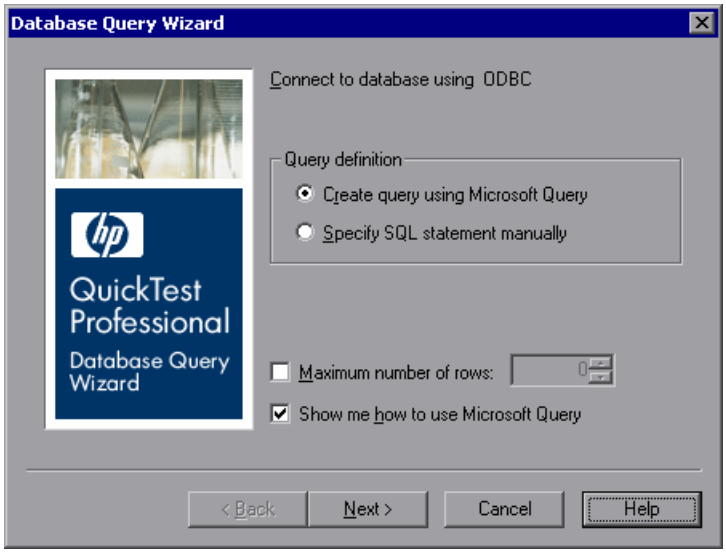
You can modify the expected data of a database checkpoint before you run your test. You can also make changes to the query in an existing database checkpoint. This can be useful if you move the database to a new location on the network.

Creating a Database Checkpoint

You can define the query for your checkpoint using Microsoft Query or by manually entering a database connection and SQL statement.

To create a database checkpoint:

1 Select **Insert** > **Checkpoint** > **Database Checkpoint**. The Database Query Wizard opens.



2 Select your database selection preferences. 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 return to QuickTest. This option is available only 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.

- **Maximum number of rows.** Select this check box if you would like to limit the number of rows and enter the maximum number of database rows to check. You can specify a maximum of 32,000 rows.
- **Show me how to use Microsoft Query.** Displays an instruction screen when you click **Next** before opening Microsoft Query. (Enabled only when **Create query using Microsoft Query** is selected).

3 Click **Next**. The screen that opens depends on the option you selected in the previous step.

- 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 579.

Note: If you chose **Show me how to use Microsoft Query**, the Instruction for Microsoft Query screen opens. When you click **OK**, Microsoft Query opens.

- If you chose **Specify SQL statement manually** in the previous step, the Specify SQL statement screen opens. Specify the connection string and the SQL statement, and click **Finish**. For more information on specifying SQL statements, see "Specifying SQL Statements" on page 580.

The Database Checkpoint Properties dialog box opens.

4 Select the checks to perform on the result set as described in "Understanding the Database Checkpoint Properties Dialog Box" on page 581. You can also modify the expected data in the result set.

5 Click **OK** to close the dialog box. A checkpoint statement is added for the selected object in the Keyword View and Expert View.

Creating a Query in Microsoft Query

You can use Microsoft Query to choose a data source and define a query within the data source.

To choose a data source and define a query in Microsoft Query:

1. When Microsoft Query opens during the insert database checkpoint process, choose a new or an existing data source.
2. Define a query.
3. When you are done, in the Finish screen of the Query Wizard, select **Exit and return to QuickTest Professional** 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 Professional** to close Microsoft Query and return to QuickTest.
4. The Database Checkpoint Properties dialog box opens. Select the checks to perform on the result set as described in "Understanding the Database Checkpoint Properties Dialog Box" on page 581. You can also modify the expected data in the result set.
5. Click **OK** to close the dialog box. A checkpoint statement is added for the selected object in the Keyword View and Expert View.

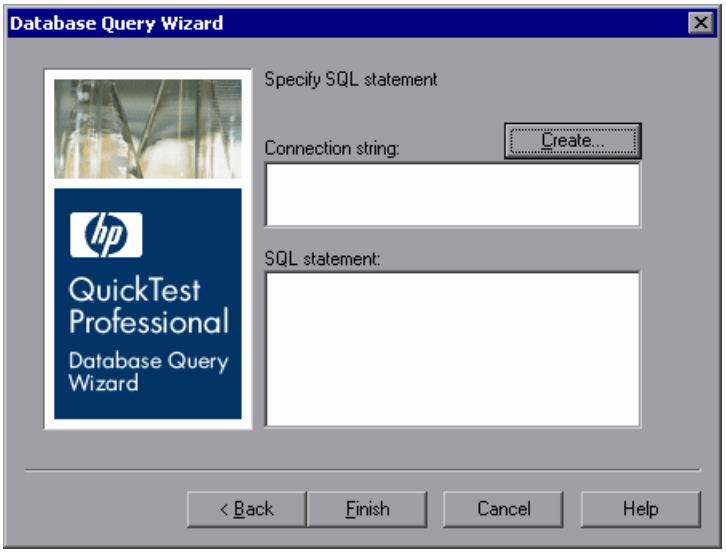
For more information on working with Microsoft Query, see your Microsoft Query documentation.

Specifying SQL Statements

You can manually specify the database connection string and the SQL statement.

To specify SQL statements:

1 Select **Specify SQL statement** in the Database Query Wizard. The following screen opens:



2 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 the Database Query Wizard insert the connection string in the box for you.
- **SQL statement.** Enter the SQL statement.

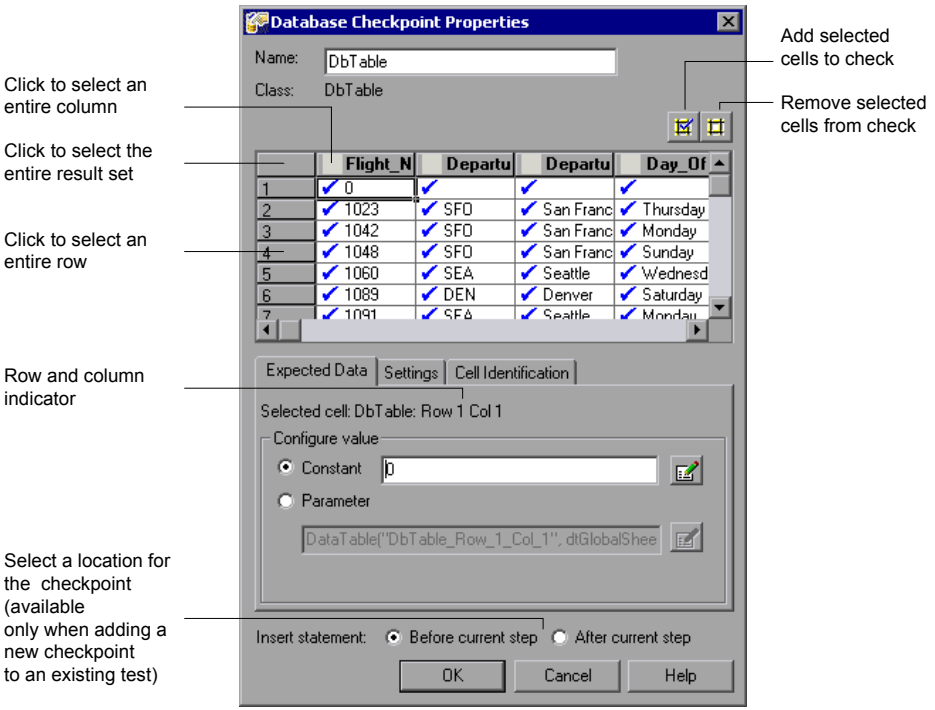
QuickTest takes several seconds to capture the database query and restore the QuickTest window.

3 Select the checks to perform on the result set as described in "Understanding the Database Checkpoint Properties Dialog Box" on page 581. You can also modify the expected data in the result set.

4 Click **OK** to close the dialog box. A checkpoint statement is added for the selected object in the Keyword View and Expert View.

Understanding the Database Checkpoint Properties Dialog Box

The Database Checkpoint Properties dialog box enables you to specify which cell contents of your database to check and which verification method and type to use. You can also edit or parameterize the expected data for the cells included in the check.



The Database Checkpoint Properties dialog box enables you to check database content.

- The information area at the top of the dialog box displays the name of the checkpoint and the class of the test object on which the checkpoint is being performed. You can rename the checkpoint, if required. When editing an existing checkpoint, you can also view it in the object repository. For more information, see "Identifying the Database Checkpoint" on page 583.
- The grid area displays the data that was captured for the checkpoint. This is the expected data. You use this area to specify which cells you want to check. For more information, see "Specifying Which Cells to Check" on page 583.
- **Expected Data tab.** Enables you to set each checked cell as a constant or parameterized value. For more information, see "Specifying the Expected Data" on page 585.
- **Settings tab.** Enables you to set the criteria for a successful match between the expected and actual values. For more information, see "Specifying the Value Type Criteria in the Settings Tab" on page 586.
- **Cell Identification tab.** Enables you to instruct QuickTest how to locate the cells to be checked. For more information, see "Specifying the Cell Identification Settings" on page 588.

Tip: The value matching settings and cell identification criteria apply to all selected cells in the checkpoint. If you want to use different value matching or cell identification criteria for different cells in the database, create separate checkpoints and specify the relevant cells for each.

- The **Insert statement** option enables you to specify the location of the checkpoint within the test. This option is available only when you add a new checkpoint while editing a test. For more information, see "Specifying The Statement Location" on page 589.

Identifying the Database Checkpoint

The top part of the Database Checkpoint Properties dialog box contains the following options:

Name	The name that QuickTest assigns to the checkpoint. By default, the checkpoint name is the name of the test object on which the checkpoint is being performed. You can specify a different name for the checkpoint or accept the default name. If you rename the checkpoint, make sure that the name: ➤ is unique ➤ does not begin or end with a space ➤ does not contain " (double quotation mark) ➤ does not contain the following character combinations: := @@
Class	Specifies the type of object (read-only).
Find in Repository button	Displays the checkpoint in its object repository. **Note:** This option is available only when editing an existing checkpoint. It is not available when creating a new checkpoint.

Specifying Which Cells to Check

The grid area of the Database Checkpoint Properties dialog box represents the cells in the captured result set.

Tip: You can change the column widths and row heights of the grid by dragging the boundaries of the column and row headers.

When you create a new database checkpoint, all cells contain a blue check mark, indicating they are selected for verification. You can select to check the entire results set, specific rows, specific columns, or specific cells. QuickTest checks only cells containing a check mark.

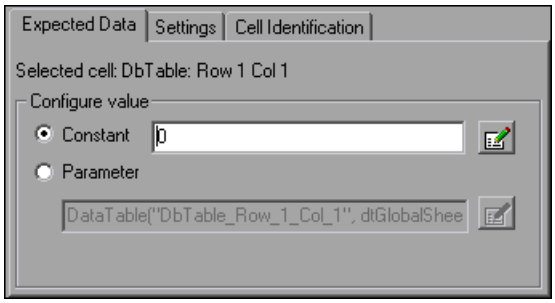
To:	Do this:
Add **a single cell** to or remove it from the check	Double-click the cell
Add **an entire row** to or remove it from the check	Double-click the row header
Add **an entire column** to or remove it from the check	Double-click the column header
Add **the entire result set** to or remove all cells from the check	Double-click the top-left corner of the grid
Add **a range of cells** to the check	Select the cells to add to the check and click the **Add to Check** button 
Remove **a range of cells** from the check	Select the cells to remove from the check and click the **Remove from Check** button 

Notes:

- Double-clicking on the grid toggles the settings for all selected cells. Therefore, if you double-click a row header, a column header, or the top left corner of the grid, any cells that were previously included in the check are removed from it, and any cells that were not previously included in the check are added to it.
- When more than one cell is selected, the options on the Expected Data tab are disabled.

Specifying the Expected Data

The Expected Data tab displays options for setting the expected value of the selected cell in the result set.



You can modify the value of a cell or you can parameterize it to use a value from an external source, such as the Data Table or an environment variable. During the run session, QuickTest compares the value specified in this tab with the actual value that it finds. If the expected value and the actual value do not match, the checkpoint fails. For example, you can instruct QuickTest to use a value from the Data Table as the expected value for a particular cell.

To modify or parameterize several cells in the result set, select a cell and then set your preferences for that cell in the Expected Data tab. Repeat the process for each cell you want to modify.

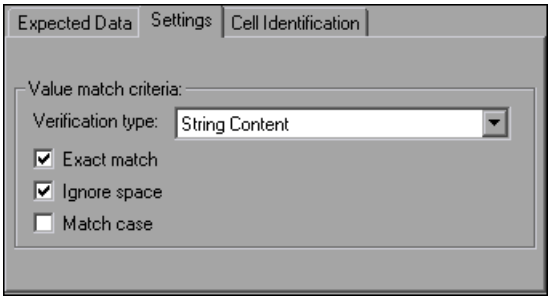
The Expected Data tab includes the following:

Selected cell	Indicates the table name and the row and column numbers of the selected cell.
Configure value area	Enables you to set the expected value of the cell as a constant or parameter. For more information on modifying values, see “Setting Values in the Configure Value Area” on page 757.

Note: When more than one cell is selected, the options on the Expected Data tab are disabled.

Specifying the Value Type Criteria in the Settings Tab

The Settings tab includes options that determine how the actual cell values are compared with the expected cell values. For example, you can instruct QuickTest to treat the value as a number so that 45 or 45.00 are treated as the same value, or you can instruct QuickTest to ignore spaces when comparing the values. The settings in this tab apply to all selected cells.



The default setting is to treat cell values as strings and to check for the exact text, while ignoring spaces.

The Settings tab includes the following options:

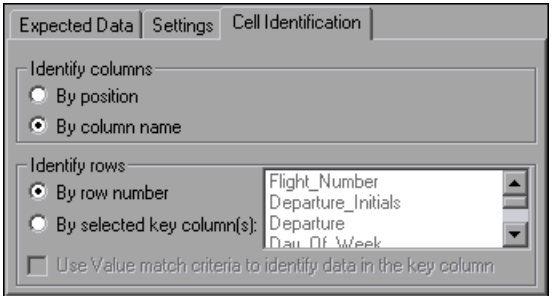
Option	Description
Verification type	Specifies how cell contents are compared: ➤ **String Content.** (Default) Evaluates the content of the cell as a string. For example, 2 and 2.00 are not recognized as the same string. ➤ **Numeric Content.** Evaluates the content of the cell according to numeric values. For example, 2 and 2.00 are recognized as the same number. ➤ **Numeric Range.** Compares the content of the cell against a numeric range, where the minimum and maximum values are any real number that you specify. This comparison differs from string and numeric content verification in that the actual result set data is compared against the range that you defined and not against a specific expected value.

Option	Description
Exact match	(Default) Checks that the exact text, and no other text, is displayed in the cell. Clear this box if you want to check that a value is displayed in a cell as part of the contents of the cell. **Note:** QuickTest displays this option only when **String Content** is selected as the **Verification type**.
Ignore space	(Default) Ignores spaces in the captured content when performing the check. The presence or absence of spaces does not affect the outcome of the check. **Note:** QuickTest displays this option only when **String Content** is selected as the **Verification type**.
Match case	Conducts a case sensitive search. **Note:** QuickTest displays this option only when **String Content** is selected as the **Verification type**.
Min / Max	Specifies the numeric range against which the content of the cell is compared. The range values can be any real number. **Note:** QuickTest displays this option only when **Numeric Range** is selected as the **Verification type**.

Specifying the Cell Identification Settings

The settings in the Cell Identification tab determine how QuickTest locates the cells to be checked. For example, suppose you want to check the data that is displayed in the first row and second column in the Database Checkpoint Properties dialog box. However, you know that each time you run your test, it is possible that the rows may be in a different order, depending on the sorting that was performed in a previous step. Therefore, rather than finding the data based on row and column numbers, you may want QuickTest to identify the cell based on the column name and the row containing a known value in a **key column**.

The settings in this tab apply to all selected cells.



The Cell Identification tab includes the following options:

Identify columns	Specifies the location of the column (in your actual database) containing the cell(s) to which you want to compare the expected data. ➤ **By position.** Locates cells according to the column position. A shift in the position of the columns within the database results in a mismatch. ➤ **By column name.** (Default) Locates cells according to the column name. A shift in the position of the columns within the database does not result in a mismatch.

Identify rows	Specifies the location of the row (in your actual database) containing the cell(s) to which you want to compare the expected data. ➤ **By row number.** (Default) Locates cells according to the row position. A shift in the position of any of the rows within the database results in a mismatch. ➤ **By selected key column(s).** Locates the row(s) containing the cells to be checked by matching the value of the cell whose column was previously selected as a **key column**. A shift in the position of the row(s) does not result in a mismatch. If more than one row is identified, QuickTest checks the first matching row. You can use more than one key column to uniquely identify any row. **Note:** A key symbol is displayed in the header of selected key columns.
Use value match criteria to identify data in the key column	Instructs QuickTest to use the verification type settings from the Settings tab as the criteria for identifying data in the key column. Enabled only when you select to identify rows **By selected key column(s)**.

Specifying The Statement Location

The **Insert statement** option specifies when to perform the checkpoint in the test.

➤ Select **Before current step** if you want to check the value of the object property before the highlighted step is performed.

➤ Select **After current step** if you want to check the value of the property after the highlighted step is performed.

Note: The **Insert statement** option is not available when adding a checkpoint during recording or when modifying an existing object checkpoint. It is available only when adding a new checkpoint to an existing test while editing it.

Modifying a Database Checkpoint

You can modify the SQL query definition and the expected data in an existing database checkpoint.

To modify the SQL query definition:

1. In the Keyword View, right-click the database object that you want to modify.
2. Select **Object Properties**.
3. Modify the SQL and connection string properties as necessary and click **OK**.

To modify the expected data in a database checkpoint:

1. In the Keyword View or Expert View, right-click the database checkpoint that you want to modify and select **Checkpoint Properties**. Alternatively, select the step containing the checkpoint and select **Edit** > **Step Properties** > **Checkpoint Properties**. The Database Checkpoint Properties dialog box opens.
2. Modify the settings as described "Understanding the Database Checkpoint Properties Dialog Box" on page 581.

23

Checking XML

By adding XML checkpoints to your test, you can check the contents of individual XML data files or documents that are part of your Web application.

This chapter includes:

About Checking XML

XML (Extensible Markup Language) is a meta-markup language for text documents that is endorsed as a standard by the World Wide Web Consortium (W3C). XML makes the complex data structures portable between different computer environments/operating systems and programming languages, facilitating the sharing of data.

XML files contain text with simple tags that describe the data within an XML document. These tags describe the data content, but not the presentation of the data. Applications that display an XML document or file use either Cascading Style Sheets (CSS) or XSL Formatting Objects (XSL-FO) to present the data.

You can verify the data content of XML files with XML checkpoints. A few common uses of XML checkpoints are described below:

- An XML file can be a static data file that is accessed to retrieve commonly used data for which a quick response time is needed—for example, country names, zip codes, or area codes. Although this data can change over time, it is normally quite static. You can use an XML file checkpoint to validate that the data has not changed from one application release to another.
- An XML file can consist of elements with attributes and values (character data). There is a parent and child relationship between the elements, and elements can have attributes associated with them. If any part of this hierarchy (including data) changes, your application's ability to process the XML file may be affected. Using an XML checkpoint, you can check the content of an element to make sure that its tags, attributes, and values have not changed.
- Web service operations often return XML values. If the Web Services Add-in is installed on your computer, you can send a Web service operation command to the service and use an XML checkpoint to verify that the service returns the XML in the expected structure and with the expected values.

- XML files are often an intermediary that retrieves dynamically changing data from one system. The data is then accessed by another system using Document Type Definitions (DTD), enabling the accessing system to read and display the information in the file. You can use an XML checkpoint and parameterize the captured data values if you want to check an XML document or file whose data changes in a predictable way.
- XML documents and files often need a well-defined structure to be portable across platforms and development systems. One way to accomplish this is by developing an XML schema, which describes the structure of the XML elements and data types. You can use schema validation to check that each item of content in an XML file adheres to the schema description of the element in which the content is to be placed.

Note: XML checkpoints are compatible with namespace standards. A change in namespace between expected and actual values results in a failed checkpoint.

For more information on XML standards, see http://www.w3.org/XML/

For more information on namespace standards, see http://www.w3.org/TR/1999/REC-xml-names-19990114/

Creating XML Checkpoints

You can perform checkpoints on XML documents contained in Web pages or frames, on XML files, and on test objects that support XML. An XML checkpoint is a verification point that compares a current value for a specified XML element, attribute and/or value with its expected value. When you insert a checkpoint, QuickTest adds a checkpoint step in the Keyword View and adds a Check CheckPoint statement in the Expert View. 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.

After you run your test, you can view summary results of the XML checkpoint in the Test Results window. You can also view detailed results by opening the XML Checkpoint Results window. For more information, see Chapter 33, "Viewing Run Session Results."

You can create three types of XML checkpoints:

- **XML Web Page/Frame Checkpoint.** Checks an XML document within a Web page or frame.
- **XML File Checkpoint.** Checks a specified XML file.
- **XML Test Object Checkpoint.** Checks the XML data for an object or operation.

Creating XML Checkpoints for Web Pages and Frames

You can create an XML checkpoint for any XML document contained in a Web page or frame using the **XML Checkpoint (From Application)** option. You can create an **XML Checkpoint (From Application)** only while recording.

To create an XML Checkpoint on XML contained in a Web page or frame:

1 Begin recording your test.

2 Select **Insert** > **Checkpoint** > **XML Checkpoint (From Application)**, or click the **Insert Checkpoint or Output Value** toolbar button and select **XML Checkpoint (From Application)**.

Note: The **XML Checkpoint (From Application)** option is available only when the Web Add-in is installed and loaded. For more information on loading add-ins, see the section on working with QuickTest add-ins in the *HP QuickTest Professional Add-ins Guide*.

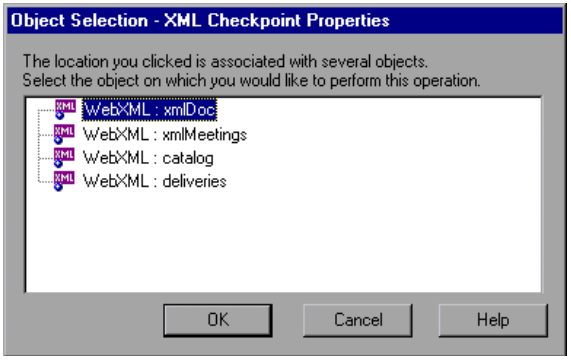
You can also insert a Web page or frame checkpoint using the **XML (From Resource)** option by selecting an existing WebXML test object as long as the actual XML object is currently available (open in the browser). For more information, see "Creating XML Test Object Checkpoints" on page 603.

The QuickTest window is hidden, and the pointer changes into a pointing hand. For more information about using the pointing hand feature, see "Tips for Using the Pointing Hand" on page 599.

3 Click a Web page or frame that contains XML documents.

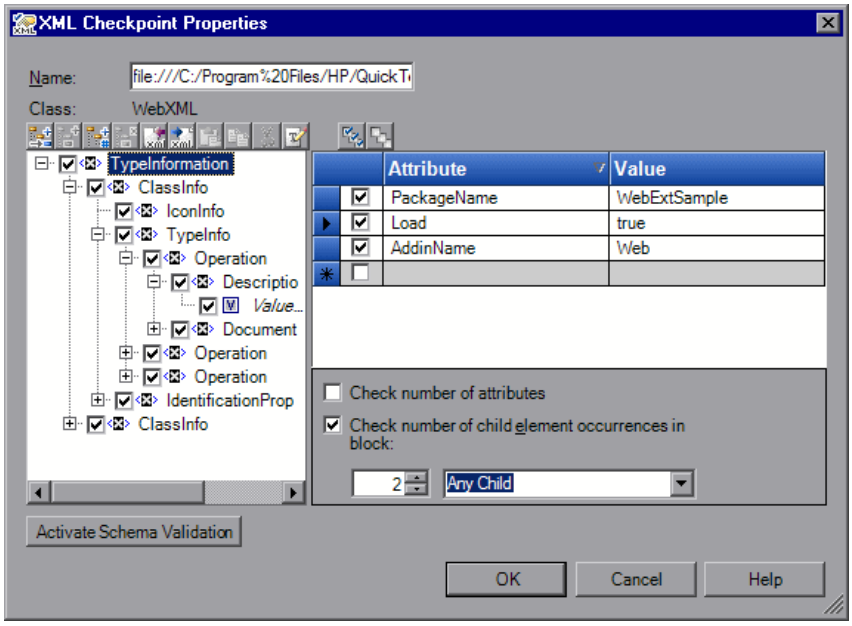
If only one XML file is associated with the Web page or frame, the XML Checkpoint Properties dialog box opens. In this case, proceed to step 5.

If more than one XML document is associated with the selected location, the Object Selection - XML Checkpoint Properties dialog box opens.



4 Select the XML document you want to check, and click **OK**.

The XML Checkpoint Properties dialog box opens and displays the root element of the selected XML document. You can expand the XML tree to view the element hierarchy and values (character data).



Note: If the XML source on which you base your checkpoint is in a valid XML format, but not to W3 standards, an error message informs you that the XML tree in the dialog box will be displayed in read-only format and that you must fix the XML source using the Edit XML as Text dialog box. For more information on this dialog box, see "Understanding the Edit XML as Text Dialog Box" on page 613.

---

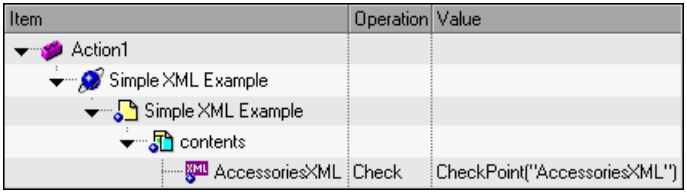
5 In the **Name** box, either accept the name that QuickTest assigns to the checkpoint or specify another name for it. By default, the checkpoint name is the name of the test object on which the checkpoint is being performed.

If you rename the checkpoint, make sure that the name:

- is unique
- does not begin or end with a space
- does not contain " (double quotation mark)
- does not contain the following character combinations:
  :=
  @@

6 Select the items to check for the checkpoint. For more information, see "Understanding the XML Checkpoint Properties Dialog Box" on page 607.

7 When you finish setting your checkpoint preferences, click **OK** to add the XML checkpoint. A checkpoint similar to the following is added to the Keyword View.



QuickTest records this step in the Expert View as:

```
Browser("Simple XML Example").Page("Simple XML Example").
    Frame("contents").WebXML("AccessoriesXML").
        Check CheckPoint("AccessoriesXML")
```

Tips for Using the Pointing Hand

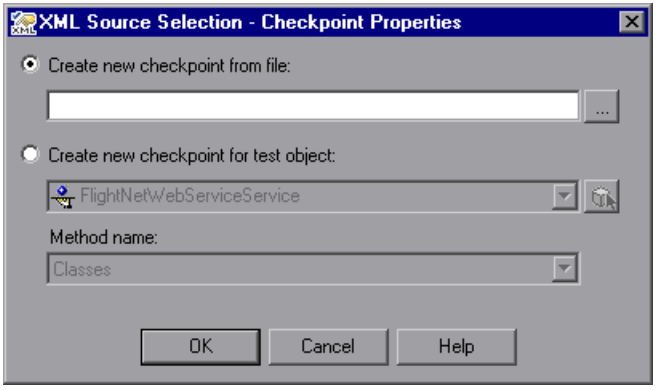
- You can hold the left CTRL key to change the pointing hand to a standard pointer. You can then change the window focus or perform operations in QuickTest or in your application, such as right-clicking, using the scroll bars, or moving the pointer over an object to display a context menu.
- If the window 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.

Creating XML File Checkpoints

You create XML file checkpoints to directly access and verify specific XML files in your system. You can create an XML file checkpoint while you are recording or editing your test.

To create an XML file checkpoint:

1 Select **Insert** > **Checkpoint** > **XML Checkpoint (From Resource)** or click the **Insert Checkpoint or Output Value** toolbar button, and select **XML Checkpoint (From Resource)**. The XML Source Selection - Checkpoint Properties dialog box opens.



Tip: You can also insert an XML file checkpoint by selecting an existing XMLFile test object as long as the source file for the test object exists in the location stored with the test object. For more information, see “Creating XML Test Object Checkpoints” on page 603.

---

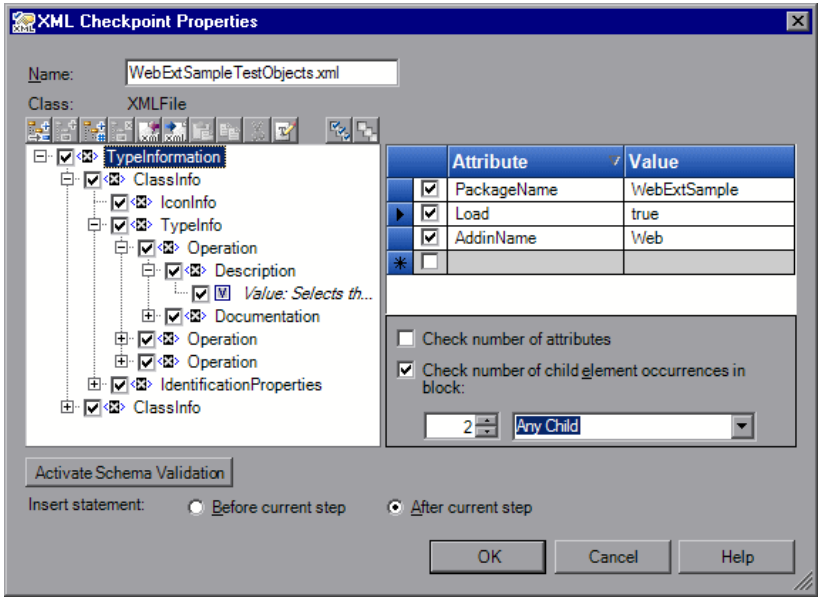
2 Select **Create new checkpoint from file**. Enter the file path or Internet address of the XML file.

Alternatively, click the browse button to open the Open XML File dialog box. In the sidebar, select the location of the XML file, and then navigate to the XML file for which you want to create a checkpoint. You can specify an XML file either from your file system or from Quality Center. Select the file and click **Open**. The file path and name are entered in the box.

Note: If you enter a relative path, QuickTest searches for the XML file in the folders listed in the Folders pane of the Options dialog box. After QuickTest locates the file, it saves it as an absolute path and uses the absolute path during the run session. For more information, see "Setting Folder Testing Options" on page 1237.

---

3 Click **OK** in the XML Source Selection - Checkpoint Properties dialog box. The XML Checkpoint Properties dialog box opens and displays the root element of the selected XML file. You can expand the XML tree to view the element hierarchy and values (character data).



Note: If the XML source on which you base your checkpoint is in a valid XML format, but not to W3 standards, an error message informs you that the XML tree in the dialog box will be displayed in read-only format and that you must fix the XML source manually using the Edit XML as Text dialog box. For more information on this dialog box, see "Understanding the Edit XML as Text Dialog Box" on page 613.

---

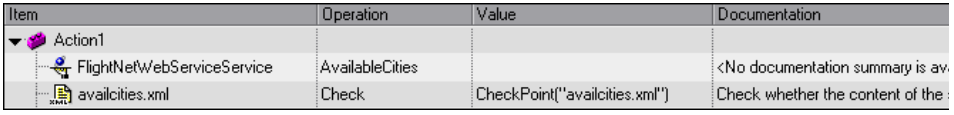
4 In the **Name** box, either accept the name that QuickTest assigns to the checkpoint or specify another name for it. By default, the checkpoint name is the name of the test object on which the checkpoint is being performed.

If you rename the checkpoint, make sure that the name:

- is unique
- does not begin or end with a space
- does not contain " (double quotation mark)
- does not contain the following character combinations:
  :=
  @@

5 Select the items to check for the checkpoint. For more information, see "Understanding the XML Checkpoint Properties Dialog Box" on page 607.

6 When you finish setting your checkpoint preferences, click **OK** to add the XML checkpoint. A checkpoint similar to the following is added to the Keyword View.



QuickTest inserts this step as follows in the Expert View:

```
XMLFile("availcities.xml").Check CheckPoint("availcities.xml")
```

Creating XML Test Object Checkpoints

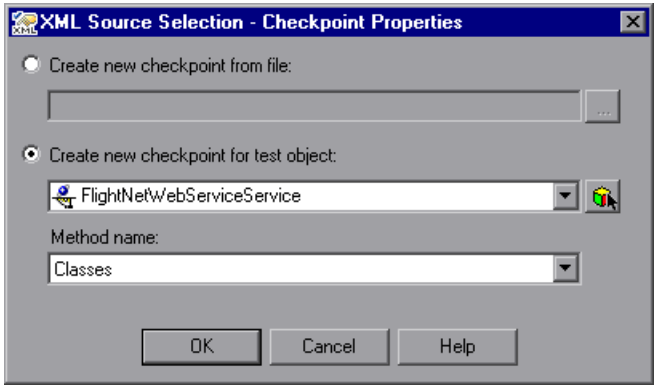
You can create an XML test object checkpoint to check the elements, attributes and/or values of XML associated with the selected test object. For example, you can check the XML that is returned from an operation performed on a Web service. You can create an XML test object checkpoint while you are recording or editing your test.

Note: You cannot insert an XML checkpoint from the Active Screen.

To create an XML test object checkpoint:

1 Select **Insert** > **Checkpoint** > **XML Checkpoint (From Resource)**, or click the **Insert Checkpoint or Output Value** toolbar button and select **XML Checkpoint (From Resource)**.

The XML Source Selection - Checkpoint Properties dialog box opens.



2 Select **Create new checkpoint for test object** and select the test object you want to check.

To select an object that is not displayed in the list, click **Object from Repository**. Then select an XML test object from the object repository on which to create a new checkpoint. The selected object must support XML. For information on selecting objects, see “Selecting an Object from the Repository or Application” on page 788.

You can select an existing WebXML or XMLFile test object type as long as the actual XML object is currently available (in an open browser or in the file system, as relevant) or, if you are working with the QuickTest Web Services Add-in you can select a WebService test object.

Note: Selecting a WebXML or XMLFile test object is identical to using the **XML Checkpoint (From Application)** or **Create new checkpoint from file** options, but may be faster than browsing to these objects and can be inserted while recording or editing. However, to use this option, the XML source must be available when you select the test object (the Web page must be open or the file must exist in the same location as when the test object was defined).

---

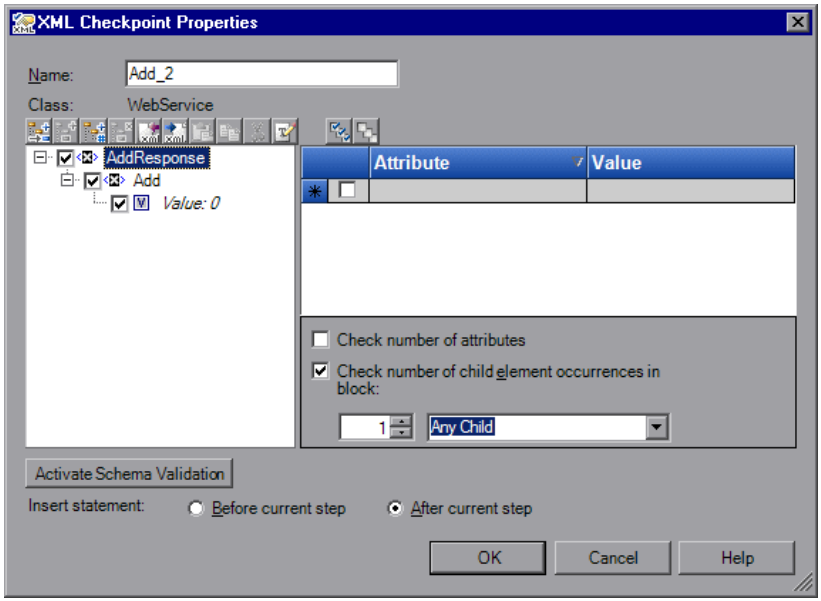
3 If you select a WebService test object, then the **Method name** box is enabled. Select the Web service operation whose return values you want to check.

Notes:

- The **Method name** box is available only if the Web Services Add-in is installed and loaded. The **Method name** box is enabled only if you select a WebService test object.
- XML Checkpoints on Web service operations compare the expected values of the checkpoint to the actual values returned from the last native Web service operation performed on the test object. If a different Web service operation step is performed prior to the checkpoint, then the checkpoint fails.

---

4 Click **OK**. The XML Checkpoint Properties dialog box opens and displays the root element of the selected XML test object. You can expand the XML tree to view the element hierarchy and values (character data).



Note: When you create an XML checkpoint for a test object operation return value (for a WebService test object), only a generic XML tree is created and shown in the XML Checkpoint Properties dialog box. The data that is expected when each operation is called during the test is not included. You must update the XML hierarchy by populating the XML tree with the actual elements, attributes, and values you want to check. For more information, see "Updating the XML Hierarchy for XML Test Object Operation Checkpoints (for WebService Test Objects Only)" on page 614.

---

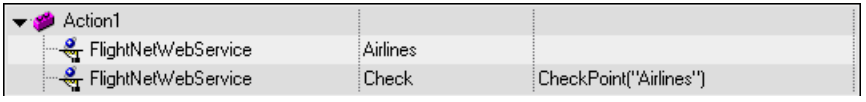
5 In the **Name** box, either accept the name that QuickTest assigns to the checkpoint or specify another name for it. By default, the checkpoint name is the name of the test object on which the checkpoint is being performed.

If you rename the checkpoint, make sure that the name:

- is unique
- does not begin or end with a space
- does not contain " (double quotation mark)
- does not contain the following character combinations:
  :=
  @@

6 Select the items to check for the checkpoint. For more information, see "Understanding the XML Checkpoint Properties Dialog Box" on page 607.

7 When you finish setting your checkpoint preferences, click **OK** to add the XML checkpoint. A checkpoint similar to the following is added to the Keyword View.

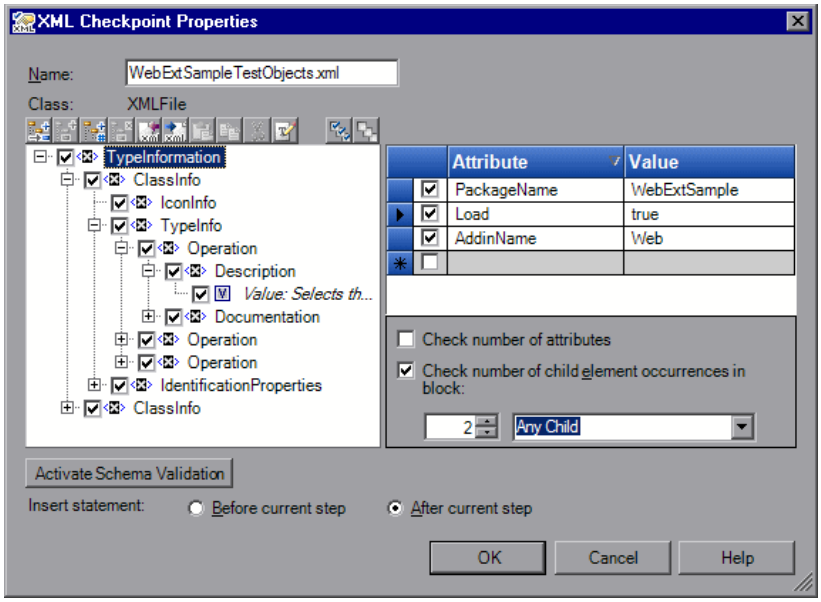


QuickTest records this step in the Expert View as:

```
WebService("FlightNetWebService").Check CheckPoint("Airlines")
```

Understanding the XML Checkpoint Properties Dialog Box

The XML Checkpoint Properties dialog box enables you to choose which elements, attributes, and/or values to check. You can also add, modify and delete elements, attributes, and values in the XML tree



In the XML tree, select the check boxes of the elements, attributes, and/or values that you want to check. For each element you want to check, select the checks you want to perform. For each attribute or value you want to check, select the checks you want to perform, or the parameterization options you want to set.

Identifying the Object

The top part of the dialog box displays test object information on the test object for which you are creating a checkpoint:

Option	Description
Name	The name that QuickTest assigns to the checkpoint. By default, the checkpoint name is the name of the test object on which the checkpoint is being performed. You can specify a different name for the checkpoint or accept the default name. If you rename the checkpoint, make sure that the name: ➤ is unique ➤ does not begin or end with a space ➤ does not contain " (double quotation mark) ➤ does not contain the following character combinations: := @@
Class	The test object class on which you are creating the checkpoint. This can be: **XMLFile** (for files), **WebXML** (for Web pages or frames), or **WebService** (for a Web service).
Find in Repository button 	Displays the checkpoint in its object repository. **Note:** This option is available only when editing an existing checkpoint. It is not available when creating a new checkpoint.

Modifying the XML Tree

The following commands are available according to the node you select in the tree:

Command	Icon	Description
Add Child		Adds a child node below the selected node in the tree.
Insert Sibling		Adds a sibling node at the same level as the selected node in the tree.
Add Value		Enables you to assign a constant or parameterized value to the selected element.
Delete		Deletes the selected node. Note that you cannot delete the root node of the checkpoint.
Import XML		Enables you to browse to an existing XML file and import it. The new file replaces the selected node and its current sub-tree.
Export XML		Enables you to save the content of the checkpoint tree to an XML file. Enabled only when the root node of the tree is selected.
Paste		Pastes a cut or copied node as a child node below the selected node in the XML tree. **Note:** You cannot paste an XML element node as its own descendant.
Copy		Makes a copy of the selected node, which you can then paste in another location in the XML tree.
Cut		Prepares the selected node to be cut and copies it to the clipboard. When you paste the node in the new location, it is removed from the original location in the XML tree.
Edit XML as Text		Opens the Edit XML as Text dialog box, enabling you to modify the XML text of the selected node and its subnodes in a test editor. For more information, see “Understanding the Edit XML as Text Dialog Box” on page 613.

Command	Icon	Description
Select All		Selects all element and value nodes in the XML tree as well as all element attributes.
Clear All		Clears all element and value nodes in the XML tree as well as all element attributes.
Duplicate		Adds a new node, identical to the selected one, as a sibling node at the same level as the selected node in the XML tree. **Note:** This command is available only from the context menu (right-click menu).

XML Tree

The XML tree displays the hierarchical relationship between each element and value in the XML tree, enabling you to select the specific elements, attributes and values you want to check. Each element is displayed with a  icon. Each value is displayed with a  icon.

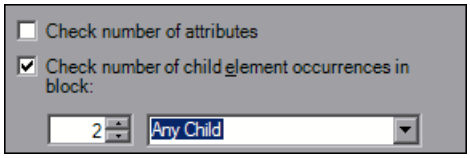
Select the check box next to an element or value node to include that item in the checkpoint. Select an element node in the XML tree to display, edit, or parameterize its expected attributes and values on the right of the XML Checkpoint Properties dialog box. Select a value node in the XML tree to display, edit, or parameterize its expected value on the right of the XML Checkpoint Properties dialog box.

Tip: The XML tree pane and the **Attribute** and **Value** columns in the right pane are resizable.

Checkpoint Options

The checkpoint options area on the bottom right of the XML Checkpoint Properties dialog box enables you to select the types of checks you want to perform on selected elements.

When you select an element in the XML Tree, the checkpoint options area includes the name of the selected element and the available element checks.



Element Checks

The following element checks are available:

Check	Description
Check number of attributes	Checks the number of attributes that are attached to the element.
Check number of child element occurrences in block	Displays the number of child elements associated with the selected parent element. If you select this option, QuickTest verifies that the number of child elements in your XML tree (with the specified name, if applicable) corresponds to the number that appears in the **Check number of child element occurrences in block** field. You can specify the child element name for the Number of child element occurrences check. If you select a child element name, QuickTest verifies that the number of child elements with that name corresponds to the number that you specify in the **Number of child element occurrences in block** field. Select **Any Child** (default) to check the total number of child elements associated with the selected parent element.

Schema Validation

You can use the **Activate Schema Validation** button to confirm that the XML in your application or file adheres to the structure defined in a specific XML schema or schemas. You can validate the structure of the XML you are checking using one or more external schema files or using schemas embedded within your XML document. For more information, see "Understanding the Schema Validation Dialog Box" on page 618.

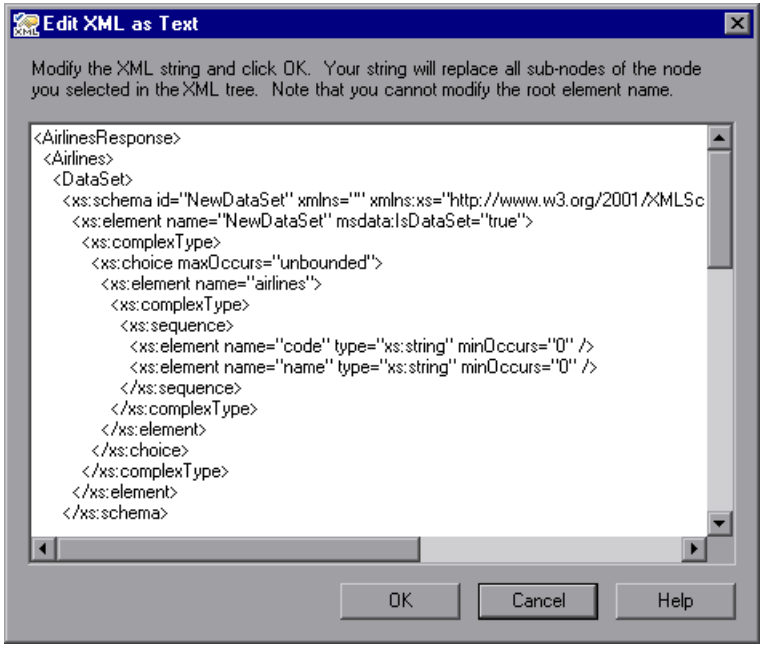
Insert Statement Options

If you are inserting a checkpoint while editing your test, the bottom part of the XML Checkpoint Properties dialog box displays **Insert statement** options, enabling you to choose whether you want to insert the XML checkpoint before or after the step that you selected. Select **Before current step** if you want to check the value of the text before the highlighted step is performed. Select **After current step** (default) if you want to check the value of the text after the highlighted step is performed.

Note: The **Insert statement** options are not available if you are adding an XML checkpoint while recording or if you are modifying an existing XML checkpoint. They are available only if you are adding a new XML checkpoint to an existing test.

Understanding the Edit XML as Text Dialog Box

The Edit XML as Text dialog box enables you to edit XML content from the XML tree in a text editor.



This dialog box is used mainly for constructing an entire XML segment from a string or for fixing syntax problems that prevent the dialog box from displaying the XML tree correctly. It is also useful when you want to use copy-paste functionality to edit the tree.

When you click **OK** in the Edit XML as Text dialog box, the sub-tree of the node you previously selected in the XML tree (or the entire tree if no node was selected or if the root node was selected) is completely replaced by the XML content from the Edit XML as Text dialog box.

Note: You cannot modify the name of the root element displayed in the Edit XML as Text dialog box.

Updating the XML Hierarchy for XML Test Object Operation Checkpoints (for WebService Test Objects Only)

This section is relevant only when working with XML Checkpoints on WebService test object operations (with the QuickTest Professional Web Services Add-in).

When you create an XML checkpoint for a test object operation (for a WebService test object), the expected operation return value data cannot be generated. Therefore, only a generic XML tree is created. To check the operation return values, you must first populate the XML tree with the actual elements, attributes, and values that the operation is expected to return.

You can use one of the three methods below to populate the XML tree:

- Updating the XML Tree Manually
- Importing an XML Tree from a File
- Updating the XML Tree Using Update Run Mode

Updating the XML Tree Manually

You can update the XML tree by adding elements, attributes, and values manually in the XML Checkpoint Properties dialog box.

To update the XML tree manually:

1 In the Keyword View, select the checkpoint whose XML tree you want to update. Click in the **Value** cell.

2 Click the **Checkpoint Properties** button or right-click and select **Checkpoint Properties**. The XML Checkpoint Properties dialog box opens.

3 Select a node in the XML tree and then click a toolbar button or select an option from the context (right-click) menu to:

- Add an element at the same level as the selected node.

- Add an element below the selected node.

- Add a value to the selected node.

- Copy the selected node.

➤ Cut the selected node (the selected node is removed only after you paste it in another location).

➤ Paste a cut or copied node as a child node below the selected node.

➤ Edit the XML text of the selected node.

➤ Delete the selected node.

➤ Duplicate the selected node, adding an identical node as a sibling node at the same level. (This command is available only from the right-click context menu.)

For more information on the available tools in the XML Checkpoint Properties dialog box, see "Understanding the XML Checkpoint Properties Dialog Box" on page 607.

Importing an XML Tree from a File

You can import an XML tree from an existing file for a specific element in the XML tree hierarchy or for the whole tree.

To import an existing XML tree from a file:

1 In the Keyword View, select the checkpoint whose XML tree you want to update.

2 Click in the **Value** cell and then click the **Checkpoint Properties** button. The XML Checkpoint Properties dialog box opens.

3 If you want to import an XML hierarchy for the whole XML tree, select the root node. If you want to import an XML hierarchy for a specific element, select the element in the XML tree hierarchy.

4 Click the **Import XML** button. A message warns you that the imported hierarchy overwrites the selected node and its sub-tree. Click **Yes** to close the message.

5 In the Import XML from File dialog box, browse to the required XML file and click **Open**. The XML hierarchy is imported from the file.

6 If required, configure a constant or parameterized value for each of the element and value nodes in the XML tree. For more information on parameterizing values, see Chapter 24, "Parameterizing Values."

Updating the XML Tree Using Update Run Mode

QuickTest cannot generate the expected return values of an operation when you insert an XML checkpoint on a Web service operation, but it can generate this information after it runs the operation. Therefore, you can run your Web service test in Update Run mode to automatically populate or update the elements, attributes and values in your XML tree.

To generate a new XML tree based on the current return values of the Web service operation, ensure that none of the node, attribute, or value check boxes are selected in the XML checkpoint.

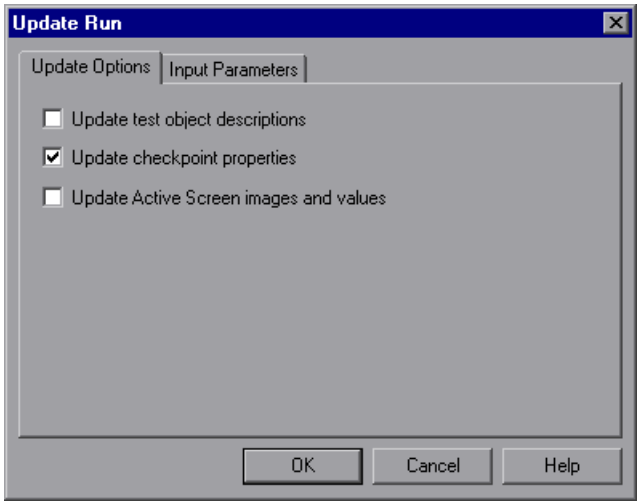
To maintain the current hierarchy in the XML tree and update only the expected values, select one or more node, attribute, or value check boxes in the dialog box.

Note: XML Checkpoints on Web service operations check the actual values returned from the last native Web service operation performed on the test object. If a different Web service operation step is performed prior to the checkpoint, then the update run cannot update the XML tree for that operation.

To update an XML tree using Update Run mode:

1 Open a test containing XML Test Object checkpoints for Web service operations.

2 Click the down arrow next to the **Run** button in the toolbar and select **Update Run Mode** or select **Automation** > **Update Run Mode**. The Update Run dialog box opens.

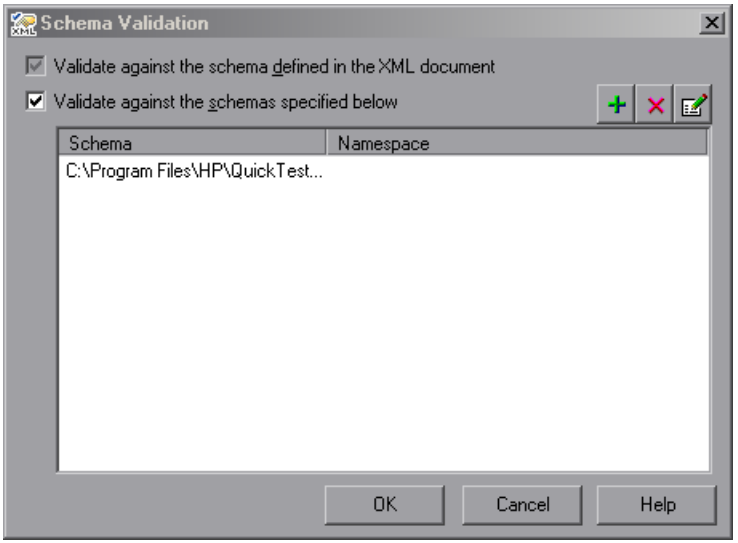


3 Select **Update checkpoint properties** and click **OK**. QuickTest runs the test and updates the XML hierarchy for each XML checkpoint.

4 If you want to confirm that QuickTest successfully updated your checkpoint, expand the tree in the Test Results window and select the XML checkpoint. Then check that Update done is displayed in the right-hand pane. (If the Test Results window did not open automatically at the end of the run, click the **Results** button or select **Automation** > **Results**.)

Understanding the Schema Validation Dialog Box

The Schema Validation dialog box enables you to specify an XML schema against which you want to validate the hierarchy of the XML in your application or file.



The Schema Validation dialog box contains the following options:

- **Validate against the schema defined in the XML document.** Instructs QuickTest to use the schema or schemas defined within your XML document to validate the hierarchy of the XML in your Web page/frame, XML file, or XML test object.
- **Validate against the schemas specified below.** Instructs QuickTest to use one or more external XML schema files to validate the hierarchy of your XML. If you select this option, any schemas defined within your XML document are also checked. (The **Validate against the schema defined in the XML document** is automatically selected and is disabled.)

When you select the **Validate against the schemas specified below** option, the **Add Schema** button is enabled. Clicking this button opens the Add Schema dialog box, in which you can specify the following:

- **Schema path or URL.** Enter the path or URL of your XML schema file. Alternatively, click the browse button to navigate to the XML schema you want to use to validate the XML in your Web page/frame, XML file, or XML test object. You can specify schema files either from your file system or from Quality Center. For each external file you add, you must specify its path or URL and namespace.
- **Schema namespace.** (If applicable.) If your schema file has a namespace, specify it. QuickTest checks that the namespace matches the schema file as part of the validation process. If the schema file has a namespace and you do not specify it, or if the namespace you specify is different to the one specified in the schema file, the validation fails.

  Click **OK** in the Add Schema dialog box to add the selected schema to the list in the Schema Validation dialog box. Click the **Add Schema** button again if you want to add another schema.

If you select **Use external schemas**, the following toolbar buttons are enabled when appropriate:

Button	Description
	Enables you to add an external schema file to the list. For more information, see “Understanding the Add Schema Dialog Box” on page 621.
	Enables you to modify the details of the selected external schema file in the list. For more information, see “Understanding the Edit Schema Dialog Box” on page 621.
	Enables you to remove the selected external schema file from the list.

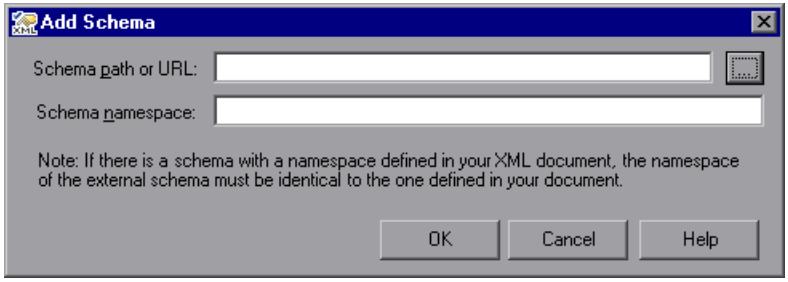
Guidelines for Schema Validation

Following are specific guidelines to consider when specifying a schema file to validate your XML.

- If you are validating an XML file using a schema defined in the XML file, the schema can be defined with an absolute or relative path. When you specify a relative path, QuickTest searches for the schema in the folders listed in the Folders pane of the Options dialog box. For more information, see "Setting Folder Testing Options" on page 1237.
- If you are validating an XML document located on the Web with a schema file located on your file system, you cannot use UNC format (for example, \\ComputerName\Path\To\Schema) to specify the schema file location. Instead, map the schema file location to a network drive.
- If there is a schema with a namespace defined in your XML document, the namespace of the external schema must be identical to the one defined in your document. Using an external XML schema file to validate an XML document may cause an unexpected result if the XML document has an XML schema declaration, and the namespace in the external schema file and the schema defined in the document are not identical.
- When you perform a schema validation, QuickTest validates all of the elements in the XML document, even if certain XML elements are not associated with a schema file. Any XML elements that are not associated with a schema file cause the schema validation to fail.

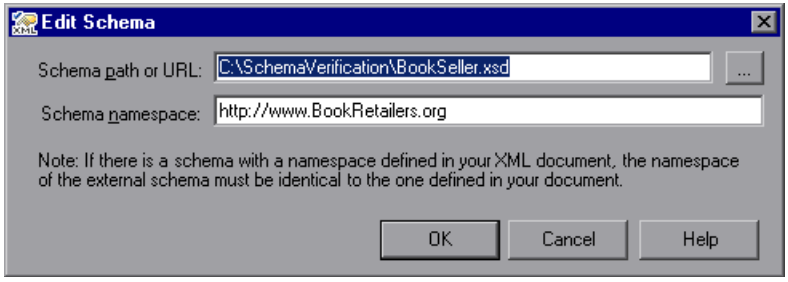
Understanding the Add Schema Dialog Box

The Add Schema dialog box enables you to specify the path or URL of an external schema file and its namespace. If there is a schema with a namespace defined in your XML document, the namespace of the external schema must be identical to the one defined in your document.



Understanding the Edit Schema Dialog Box

The Edit Schema dialog box displays the path and namespace of the schema file you selected in the list. You can modify the path or URL of the selected schema file, and its namespace.



Modifying XML Checkpoints

You can change the expected data and settings of an existing XML checkpoint.

To modify an XML checkpoint:

1. In the Keyword View or the Expert View, right-click the XML checkpoint that you want to modify and select **Checkpoint Properties**. Alternatively, select the step containing the XML checkpoint and select **Edit** > **Step Properties** > **Checkpoint Properties**. The XML Checkpoint Properties dialog box opens.
2. Modify the settings as described in the previous sections.

Reviewing XML Checkpoint Results

By adding XML checkpoints to your tests, you can verify that the data and structure in your XML documents, XML files, or XML test objects have 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.

You can view summary results of the XML checkpoint in the Test Results window. You can view detailed results by opening the XML Checkpoint Results window. For more information on XML checkpoint results, see "Analyzing XML Checkpoint Results" on page 1037.

Note: XML Checkpoints on Web service operations compare the expected values of the checkpoint to the actual values returned from the last native Web service operation performed on the test object. If a different Web service operation step is performed prior to the checkpoint, then the checkpoint fails.

Using XML Objects and Methods to Enhance Your Test

QuickTest provides several scripting methods that you can use with XML data. You can use these scripting methods to retrieve data and return new XML objects from existing XML data. You do this by using the XMLUtil, or WebXML objects to return XML data and then using the supported XMLData objects and methods to manipulate the returned data.

Tip: All XMLData objects and methods are compatible with namespace and XPath standards.

For more information on XML standards, see http://www.w3.org/XML/

For more information on namespace standards, see http://www.w3.org/TR/1999/REC-xml-names-19990114/

For more information on XPath standards, see http://www.w3.org/TR/1999/REC-xpath-19991116

For more information on programming in the Expert View, see Chapter 29, "Working in the Expert View and Function Library Windows." For more information on XML objects and methods, see the Supplemental section of the *HP QuickTest Professional Object Model Reference*.

24

Parameterizing Values

QuickTest enables you to expand the scope of a basic test by replacing fixed values with parameters. This process, known as **parameterization**, greatly increases the power and flexibility of your test.

This chapter includes:

About Parameterizing Values

You can use the parameter feature in QuickTest to enhance your test by parameterizing the values that it uses. A **parameter** is a variable that is assigned a value from an external data source or generator.

You can parameterize the values in steps or the values of action parameters using one of the following parameter types:

➤ **Test/action parameters.** Test parameters enable you to use values passed from your test. Action parameters enable you to pass values from other actions in your test.

To use a value within a specific action, you must pass the value down through the action hierarchy of your test to the required action. You can then use that parameter value to parameterize a step in your test. For example, suppose that Action3 is a nested action of Action1 (a top-level action), and you want to parameterize a step in Action3 using a value that is passed into your test from the external application that runs (calls) the test. You can pass the value from the test level to Action1, then to Action3, and then parameterize the required step using this action input parameter value (that was passed through from the external application).

Alternatively, you can pass an output action parameter value from an action step to a later sibling action at the same hierarchical level. For example, suppose that Action2, Action3, and Action4 are sibling actions at the same hierarchical level, and that these are all nested actions of Action1. You can parameterize a call to Action4 based on an output value retrieved from Action2 or Action3. You can then use these parameters in your action step.

For more information, see “Guidelines for Working with Action Parameters” on page 479.

➤ **Data Table parameters.** Enable 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.

For example, suppose your application includes a feature that enables users to search for contact information from a membership database. When the user enters a member's name, the member's contact information is displayed, together with a button labelled **View <MemName>'s Picture**, where **<MemName>** is the name of the member. You can parameterize the name property of the button using a list of values so that during each iteration of the run session, QuickTest can identify the different picture buttons.

➤ **Environment variable parameters.** Enable you to use variable values from other sources during the run session. These may be values you supply, or values that QuickTest generates for you based on conditions and options you choose.

For example, you can have QuickTest read all the values for filling in a Web form from an external file, or you can use one of QuickTest's built-in environment variables to insert current information about the computer running the test.

➤ **Random number parameters.** Enable you to insert random numbers as values in your test. For example, to check how your application handles small and large ticket orders, you can have QuickTest generate a random number and insert it in a **number of tickets** edit box.

Tips:

➤ If you want to parameterize the same value in several steps in your test, you may want to consider using the Data Driver rather than adding parameters manually. For more information see, "Using the Data Driver to Parameterize Your Test" on page 662.

➤ You can also parameterize identification property values of test objects in the object repository using repository parameters. For more information, see "Working with Repository Parameters" on page 228.

Parameterizing Values in Steps and Checkpoints

You can parameterize values in steps and checkpoints while working with your test.

You can parameterize the values of object properties for a selected step. You can also parameterize the values of the operation arguments defined for the step.

For example, your application may include a form with an edit box into which the user types the user name. You may want to test whether your application reads this information and displays it correctly in a dialog box. You can insert a text checkpoint that uses the built-in environment variable for the logged-in user name, to check whether the displayed information is correct.

Note: When you parameterize the value of an object property for a local object, you are modifying the test object description in the local object repository. Therefore, all occurrences of the specified object within the action are parameterized. For more information on the local object repository, see Chapter 5, "Managing Test Objects in Object Repositories."

Parameterizing the value of a checkpoint property enables you to check how an application performs the same operation based on different data.

For example, if you are testing the Mercury Tours sample Web site, you may create a checkpoint to check that once you book a ticket, it is booked correctly. Suppose that you want to check that flights are booked correctly for a variety of different destinations. Rather than create a separate test with a separate checkpoint for each destination, you can add a Data Table parameter for the destination information. This enables you to create a list of different destinations. QuickTest will then check the flight information for a different destination for each iteration of the test.

For more information on using checkpoints, see Chapter 17, "Understanding Checkpoints."

When you define a value as a parameter, you specify the parameter type and its settings.

For more information on using specific parameter types, see:

- "Using Test and Action Input Parameters" on page 635
- "Using Data Table Parameters" on page 639
- "Using Environment Variable Parameters" on page 645
- "Using Random Number Parameters" on page 655

For more information on parameterizing values:

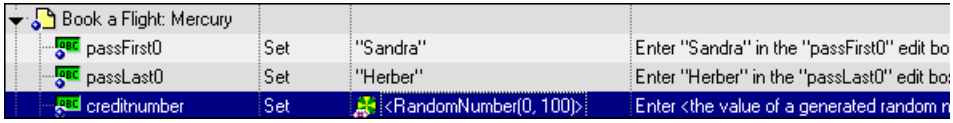
- "Parameterizing Values for Operations" on page 629
- "Parameterizing Property Values for Objects and Checkpoints" on page 631

Tip: When you use the Step Generator to add new steps, you can parameterize the values for the operation you select. For more information, see "Inserting Steps Using the Step Generator" on page 777.

Parameterizing Values for Operations

If the method, property, or function used in the step has arguments, you can parameterize the argument values as required. For example, if the operation uses the **Click** method, you can parameterize the values for the **x** argument, the **y** argument, or both.

When you select a parameterized value in the Keyword View, the icon for the parameter type is displayed. For example, in the following segment, the value of the **Set** method has been defined as a random number parameter. QuickTest enters a random number value into the **creditnumber** edit box each time the test runs.

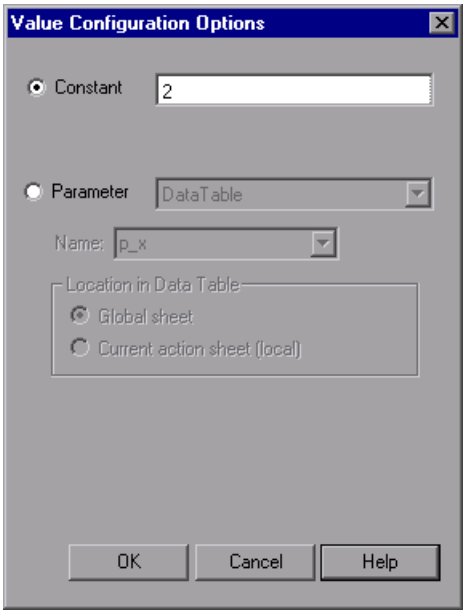


You can parameterize operation values using the parameterization icon in the **Value** column of the Keyword View.

To parameterize a value for an operation using the parameterization icon:

1 In the Keyword View, click in the **Value** column of the required step.

2 Click the parameterization icon for the value that you want to parameterize. The Value Configuration Options dialog box opens, showing the currently defined value.



Note: The parameter options shown in this dialog box change according to the parameter type selected in the **Parameter** box.

---

3 Select **Parameter**. If the value is already parameterized, the **Parameter** section displays the current parameter definition for the value. If the value is not yet parameterized, the **Parameter** section displays the default parameter definition for the value. For more information, see "Understanding Default Parameter Values" on page 634.

4 Accept or change the parameter definition:

- Click **OK** to accept the displayed parameter statement and close the dialog box.
- Modify the value settings for the selected parameter type and click **OK**.
- Change the parameter type. The options in the **Parameter** section change according to the parameter type you select.

For more information on configuring values for specific parameter types, see:

- "Defining the Settings for a Test or Action Parameter" on page 637
- "Defining the Settings for a Data Table Parameter" on page 642
- "Defining the Settings for an Environment Variable Parameter" on page 652
- "Defining Settings for a Random Number Parameter" on page 656

Parameterizing Property Values for Objects and Checkpoints

You can parameterize the values for one or more properties of an object stored in the local object repository in the Object Properties dialog box or Object Repository window. You can parameterize the values for one or more properties of a checkpoint in the Checkpoint Properties dialog box.

Note: For information on parameterizing a property value for an object in a shared object repository, see Chapter 7, "Managing Object Repositories."

To parameterize local object values:

1 Open the dialog box for the object properties in one of the following ways:

➤ Select a step and select **Edit** > **Step Properties** > **Object Properties**, or right-click a step and select **Object Properties**. The Object Properties dialog box opens.

➤ Open the **Object Repository** dialog box and select the object,

2 Click in the **Value** cell for the property that you want to parameterize, and click the parameterization icon  . The Value Configuration Options dialog box opens.

3 Select **Parameter**. If the value is already parameterized, the **Parameter** box displays the current parameter definition for the value. If the value is not yet parameterized, the **Parameter** box displays the default parameter definition for the value. For more information, see "Configuring a Selected Value" on page 760.

4 Click **OK** to accept the displayed parameter statement or change the displayed parameter definition, and then click **OK**.

5 To accept the displayed parameter statement and parameterize another of the displayed values, select another property and follow the previous steps.

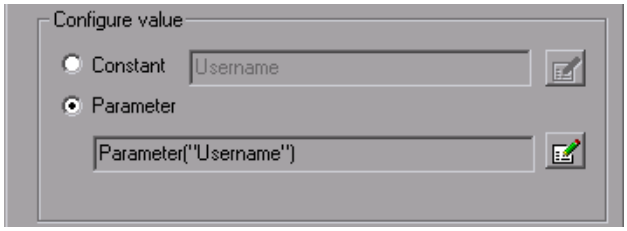
To parameterize checkpoint property values:

1 Open the dialog box for the checkpoint properties in one of the following ways:

➤ Select **Edit** > **Step Properties** > **Checkpoint Properties**, or right-click the checkpoint and select **Checkpoint Properties**.

➤ Open the **Object Repository** dialog box and select the checkpoint.

2 In the **Configure value** area of the dialog box, select **Parameter**.



If the value is already parameterized, the **Parameter** box displays the current parameter definition for the value. If the value is not yet parameterized, the **Parameter** box displays the default parameter definition for the value. For more information, see "Understanding Default Parameter Values" on page 634.

3 Accept or change the displayed parameter definition:

- To accept the displayed parameter statement and close the dialog box, click **OK**.

- To change the parameter type or modify the value settings for the selected property, click the **Parameter Options** button. The Parameter Options dialog box opens for the displayed parameter type.

4 To accept the displayed parameter statement and parameterize another of the displayed values, select another property and follow the previous steps.

For more information on defining value settings for specific parameter types, see:

- "Setting Test and Action Parameter Options" on page 636
- "Setting Data Table Parameter Options" on page 641
- "Choosing Global or Action Data Table Parameters" on page 643
- "Using Random Number Parameters" on page 655

Understanding Default Parameter Values

When you select a value that has not yet been parameterized, QuickTest generates a default parameter definition for the value. The following table describes how the default parameter settings are determined:

When parameterizing	Condition	Default parameter type	Default parameter name
A value for a step or a checkpoint in an action	At least one input action parameter is defined in the current action	Action parameter	The first input parameter displayed in the Parameters tab of the Action Properties dialog box
An input action parameter value for a nested action	At least one input action parameter is defined for the action calling the nested action	Action parameter	The first input parameter displayed in the Parameters tab of the Action Properties dialog box of the calling action
An input action parameter value for a top-level action call	At least one input parameter is defined for the test	Test parameter	The first input parameter displayed in the Parameters pane of the Test Settings dialog box

If the relevant condition described above is not true, the default parameter type is Data Table. If you accept the default parameter details, QuickTest creates a new Data Table parameter with a name based on the selected value. Data Table parameters are created in the Global sheet.

For more information on Data Table sheets, see Chapter 42, "Working with Data Tables."

Using Test and Action Input Parameters

You can parameterize a step using a test or action input parameter. This enables the step to use values that have been passed from the application that ran (called) your test. For example, you can use an input test parameter as the value for a method argument.

You can parameterize a value using a test or action parameter only if the parameter has been defined for the test or action. For more information on defining parameters, see "Defining Parameters for Your Test" on page 1280, "Setting Action Parameters" on page 472, and "Setting Action Call Parameter Values" on page 483.

You can parameterize steps by selecting input parameters in the Parameter Options or Value Configuration Options dialog box. The parameter options that are available in these dialog boxes depend on where you are currently located in your test, and whether test or action parameters are defined. For more information, see "Using Action Parameters" on page 476 and "Defining Parameters for Your Test" on page 1280.

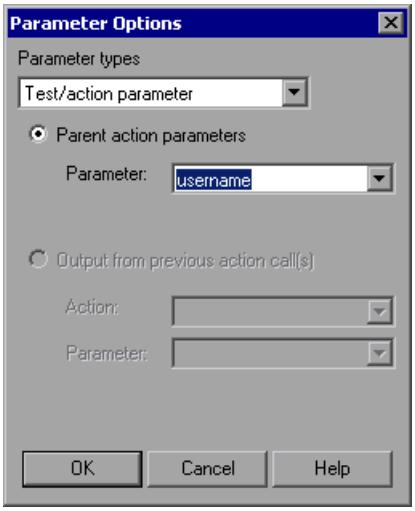
Alternatively, you can enter the parameter name in the Expert View using the Parameter utility object, in the format: Parameter("ParameterName") for the current action, or Parameter("ActionName", "ParameterName") to use the output parameter from a previous action as an input parameter in the current action. For more information, see "Using Action Parameters in Steps in the Expert View" on page 638.

Tip: You can also create test or action parameter output values that retrieve values during the run session and store them for use at another point in the run session. You can then use these output values to parameterize a step in your test. For more information, see "Outputting a Value to an Action Parameter" on page 684.

Setting Test and Action Parameter Options

When you choose to parameterize a value, the dialog box that opens enables you to select a parameter type and the parameter options to use. The image below shows the dialog box that opens when you select to parameterize a checkpoint expected value. The dialog boxes for parameterizing other value types such as argument values, object property values, and output storage locations provide similar options.

When **Test/action parameter** is selected as the parameter type, you can select the required parameter from a list of existing parameters.



Tip: When you open the dialog box to parameterize a value, the default parameter type may be set to **Test/action parameter**. For more information on default parameter type settings, see "Understanding Default Parameter Values" on page 634.

Defining the Settings for a Test or Action Parameter

The following options are available for configuring test or action parameters:

- **Test parameters** or **Parent action parameters.** Parameter defined in the test or parent action. (If no output parameters are defined in the test or parent action, this area is disabled.) **Test parameters** are available only for top-level actions. They are defined in the Parameters pane of the Test Settings dialog box. **Parent action parameters** are available for subsequent steps and for nested actions. They are defined in the action containing the steps or in the action that calls the nested action.
  - **Parameter.** Specifies the name of the input parameter. The read-only list of available parameters contains the names and full descriptions of the currently defined input parameters for the action. You can resize the display, as needed, and, if the list of parameters is long, you can scroll through the list.
- **Output from previous action call(s).** Any previous action in the same hierarchical level for which output parameters are defined. (If no output parameters are defined in previous actions, this area is disabled.)
  - **Action.** Specifies the previous action from which you can choose an output parameter. You can choose any action in the list.
  - **Parameter.** Specifies the name of the output parameter. The read-only list of available parameters contains the names and full descriptions of the currently defined output parameters from the previous action(s). You can resize the display, as needed, and, if the list of parameters is long, you can scroll through the list.

You can also use test or action parameter variables using parameterization objects and methods in the Expert View. For more information, see the *HP QuickTest Professional Object Model Reference*.

Using Action Parameters in Steps in the Expert View

Instead of selecting input (or output) parameters from the appropriate dialog boxes while parameterizing steps or inserting output value steps, you can enter input and output parameters as values in the Expert View using the Parameter utility object in the format: Parameter("ParameterName").

Suppose you have test steps that enter information in a form to display a list of purchase orders in a table, and then return the total value of the orders displayed in the table.

You can define input parameters, called **SoldToCode** and **MaterialCode**, for the codes entered in the **Sold to** and **Materials** edit boxes of the form so that the Orders table that is opened is controlled by the input parameter values passed when the test is called.

You can define an output parameter, called **TotalValue**, to store the returned value. The output value (**TotalValue**) could then be returned to the application that called the test.

The example described above might look something like this (parameters are in bold font):

```
Browser("Mercury").Page("List Of Sales").WebEdit("Sold to").
    Set Parameter("SoldToCode")
Browser("Mercury").Page("List Of Sales").WebEdit("Materials").
    Set Parameter("MaterialCode")
Browser("Mercury").Page("List Of Sales").WebButton("Enter").Click
NumTableRows = Browser("Mercury").Page("List Of Sales").
    WebTable("Orders").RowCount
Parameter("TotalValue") = Browser("Mercury").Page("List Of Sales").
    WebTable("Orders").GetCellData(NumTableRows,"Total")
```

Using Data Table Parameters

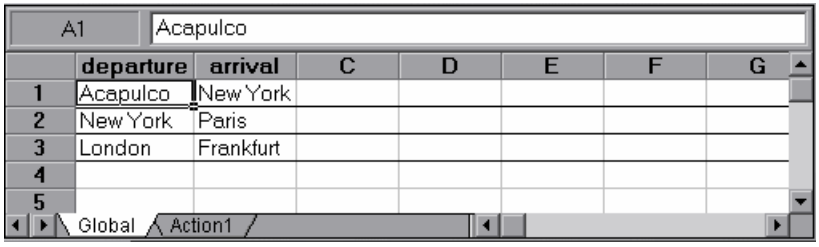
You can supply the list of possible values for a parameter by creating a Data Table parameter. Data Table parameters enable you to create a data-driven test, or action that runs several times using the data you supply. In each repetition, or **iteration**, QuickTest uses a different value from the Data Table (taken from the subsequent row in the Data Table).

For example, consider the Mercury Tours sample Web site, which enables you to book flight requests. To book a flight, you supply the flight itinerary and click the **Continue** button. The site returns the available flights for the requested itinerary.

You could conduct the test by accessing the Web site and submitting numerous queries. This is a slow, laborious, and inefficient solution. By using Data Table parameters, you can run the test for multiple queries in succession.

When you parameterize your test, you first create steps that access the Web site and check for the available flights for one requested itinerary.

You then substitute the existing itinerary with a Data Table parameter and add your own sets of data to the relevant sheet of the Data Table, one for each itinerary.



When you create a new Data Table parameter, a new column is added in the Data Table and the current value you parameterized is placed in the first row. If you parameterize a value and select an existing Data Table parameter, then the values in the column for the selected parameter are retained, and are not overwritten by the current value of the parameter.

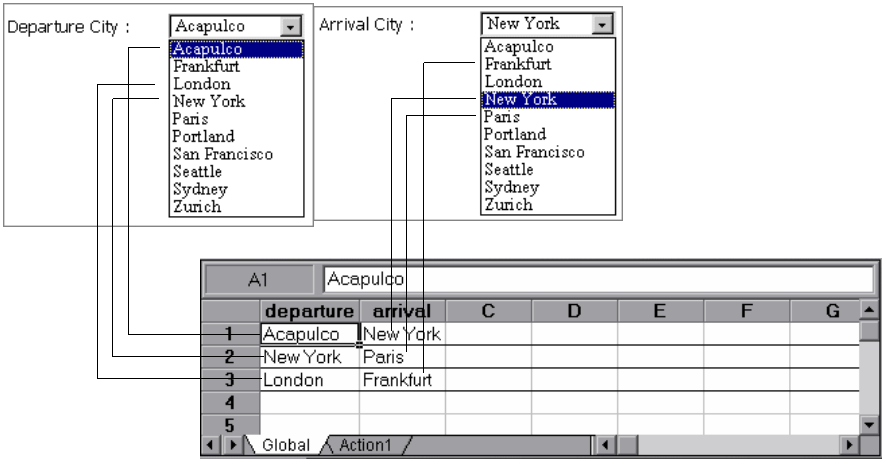
Each column in the table represents the list of values for a single Data Table parameter. The column header is the parameter name.

Each row in the table represents a set of values that QuickTest submits for all the parameters during a single iteration of the test. When you run your test, QuickTest runs one iteration of the test for each row of data in the table. For example, a test with ten rows in the Global sheet of the Data Table will run ten times.

For more information on entering values in the Data Table, see Chapter 42, "Working with Data Tables."

Tip: You can also create Data Table output values, which retrieve values during the run session and insert them into a column in the Data Table. You can then use these columns as Data Table parameters in your test. For more information, see Chapter 25, "Outputting Values."

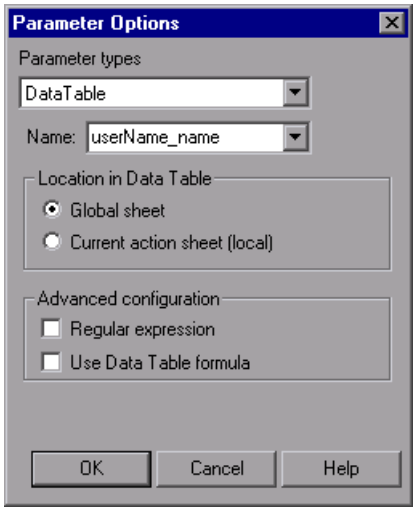
In the previous example, QuickTest submits a separate query for each itinerary when you run the test.



Setting Data Table Parameter Options

When you choose to parameterize a value, the dialog box that opens enables you to select a parameter type and the parameter options to use. The image below shows the dialog box that opens when you select to parameterize a checkpoint expected value. The dialog boxes for parameterizing other value types such as argument values, object property values, and output storage locations provide similar options.

When **Data Table** is selected as the parameter type, you can configure your parameter to use values from the Data Table.



Tip: When you open the dialog box to parameterize a value, **Data Table** may be set as the default parameter type. For more information on default parameter type settings, see “Understanding Default Parameter Values” on page 634.

Defining the Settings for a Data Table Parameter

The following options are available for configuring Data Table parameters:

Name. Specifies the name of the parameter in the Data Table. You can create a new parameter by using the default parameter name or entering a new, descriptive name. Alternatively, you can select an existing Data Table parameter from the list.

Note: The parameter name must be unique in the sheet. It can contain letters, numbers, periods, and underscores. The first character of the parameter name must be a letter or an underscore. If you specify an invalid name, QuickTest displays a warning message when you click **OK**. You can choose to edit the name manually or to instruct QuickTest to fix the name automatically (by adding an underscore at the beginning of the name).

Location in Data Table. Specifies whether to store the parameter in the global or current action sheet in the Data Table.

For more information on global and action Data Table parameters, see "Choosing Global or Action Data Table Parameters" on page 643. For more information on actions, see Chapter 15, "Working with Actions" and Chapter 16, "Working with Advanced Action Features."

Advanced configuration (if applicable)**:**

- **Regular expression.** Sets the value of the parameter as a regular expression. For more information, see "Understanding and Using Regular Expressions" on page 762. Note that this option is available only when parameterizing checkpoint and object property values.
- **Use Data Table formula.** (If applicable.) Inserts two columns in the Data Table. The first column contains a formula that checks the validity of output in the second column. QuickTest uses the data in the output column to compute the formula, and inserts a value of TRUE or FALSE in the table cell of the formula column. Note that this option is available only for checkpoints. For more information on using Data Table formulas, see "Using Formulas in the Data Table" on page 1216.

Note: You can also define Data Table variables using parameterization objects and methods in the Expert View. For more information, see the *HP QuickTest Professional Object Model Reference*.

Choosing Global or Action Data Table Parameters

When you parameterize a step in a test using the Data Table, you must decide whether you want to make it a **global Data Table parameter** (per test) or a **local Data Table parameter** (per action).

This decision should be based on whether you want the data to be used only for a single action (use local Data Table parameters), or available to other actions (use global Data Table parameters) and when you want subsequent iterations (different data) to be used for a particular parameter (each time the test repeats or each time the action repeats within the test).

➤ Global Data Table parameters take data from the Global sheet in the Data Table. The Global sheet contains the data that replaces global parameters in each iteration of the test. By default, the test runs one iteration for each row in the Global sheet of the Data Table. Using the Run pane of the Test Settings dialog box, you can also set the test to run only one iteration, or to run iterations on specified rows within the Global sheet of the Data Table. You can use the parameters defined in the Global data sheet in any action.

Tip: By outputting values to the global Data Table sheet from one action and using them as input parameters in another action, you can pass values from one action to another. For more information, see Chapter 25, "Outputting Values."

For more information on setting global iteration preferences, see "Defining Run Settings for Your Test" on page 1270.

➤ Local Data Table parameters take data from the action's sheet in the Data Table. The data in the action's sheet replaces the action's Data Table parameters in each iteration of the action. By default, actions run only one iteration.

Using the Run tab of the Action Call Properties dialog box, you can also set a particular call of the action to run iterations for all rows in the action's sheet or to run iterations on specified rows within the action's sheet. When you set your action properties to run iterations on all rows, QuickTest inserts the next value from the action's data sheet into the corresponding action parameter during each **action iteration**, while the values of the global parameters stay constant.

For more information on setting action iteration preferences, see "Inserting a Call to an Existing Action" on page 468.

Note: After running a parameterized test, you can view the actual values taken from the Data Table in the Test Results Run-Time Data Table. For more information, see "Viewing the Run-Time Data Table" on page 1056.

If you have multiple rows in the Global data sheet, the entire test runs multiple times. If you have multiple rows in a local data sheet, the corresponding action runs multiple times before running the next action in the test. If you have multiple rows in both Global and local data sheets, each single test iteration runs all iterations of each action before running the next iteration of the test.

Using Environment Variable Parameters

QuickTest can insert a value from the Environment variable list, which is a list of variables and corresponding values that can be accessed from your test. Throughout the test run, the value of an environment variable remains the same, regardless of the number of iterations, unless you change the value of the variable programmatically in your script.

Tip: Environment parameters are especially useful for localization testing, when you want to test an application where the user interface strings change, depending on the selected language. Environment parameters can be used for testing the same application on different browsers. You can also vary the input values for each language by selecting a different Data Table file each time you run the test. For more information, see Chapter 42, "Working with Data Tables."

There are several types of environment variables:

➤ **User-Defined Internal.** Variables that you define within the test. These variables are saved with the test and are accessible only within the test in which they were defined.

You can create or modify internal, user-defined environment variables for your test in the Environment pane of the Test Settings dialog box or in the Parameter Options dialog box.

For more information on creating or modifying environment variables in the Test Settings dialog box, see "Defining Environment Settings for Your Test" on page 1283.

For information on creating or modifying environment variables in the Parameter Options dialog box, see "Setting Environment Variable Parameter Options" on page 652.

Tip: You can also create environment output values, which retrieve values during the test run and output them to internal environment variable parameters for use in your test. For more information, see Chapter 25, "Outputting Values."

- **User-Defined External.** Variables that you predefine in the active external environment variables file. You can create as many files as you want and select an appropriate file for each test, or change files for each test run. Note that external environment variable values are designated as read-only within the test. For more information, see "Using User-Defined External Environment Variables" on page 647.
- **Built-in.** Variables that represent information about the test and the computer on which the test is run, such as Test path and Operating system. These variables are accessible from all tests, and are designated as read-only. For more information, see "Using Built-in Environment Variables" on page 650.

Note: QuickTest also has a set of predefined environment variables that you can use to set the values of the Record and Run Settings dialog options. You should not use the names of these variables for any other purpose. For more information, see the section on using environment variables to specify the Record and Run details for your test in the *HP QuickTest Professional User Guide*.

Using User-Defined External Environment Variables

You can create a list of variable-value pairs in an external file in **.xml** format. You can then select the file as the active external environment variable file for a test and use the variables from the file as parameters.

You can set up your environment variable files manually, or you can define the variables in the Environment pane of the Test Settings dialog box and use the **Export** button to create the file with the correct structure. For more information on exporting environment variables, see Chapter 45, "Setting Options for Individual Tests."

Notes:

- You can also store environment variable files in Quality Center. For more information, see "Using Environment Variable Files with Quality Center" on page 649.
- You can create several external variable files with the same variable names and different values and then run the test several times, using a different file each time. This is especially useful for localization testing.

If you create your files manually, you must use the correct format, as defined below. You can use the QuickTest environment variable file schema in: **<QuickTest Professional installation folder>\help\QTEnvironment.xsd**

To create an external environment variables file:

1. Create an xml file using the editor of your choice.
2. Type <Environment> on the first line.
3. Type each variable name-value pair within <Variable> elements in the following format:

```
<Variable>
   <Name>This is the first variable's name</Name>
   <Value>This is the first variable's value</Value>
   <Description> This text is optional and can be used to add comments. It is
      shown only in the XML not in QuickTest</Description>
</Variable>
```

4 Type </Environment> on the last line.

For example, your environment variables file may look like this:

```
<Environment>
   <Variable>
      <Name>Address1</Name>
      <Value>25 Yellow Road</Value>
   </Variable>
   <Variable>
      <Name>Address2</Name>
      <Value>Greenville</Value>
   </Variable>
   <Variable>
      <Name>Name</Name>
      <Value>John Brown</Value>
   </Variable>
   <Variable>
      <Name>Telephone</Name>
      <Value>1-123-12345678</Value>
   </Variable>
</Environment>
```

5 Save the file in a location that is accessible from the QuickTest computer. The file must be in .xml format with an **.xml** file extension.

To select the active external environment variables file:

1 Select **File** > **Settings** to open the Test Settings dialog box. For more information on the Test Settings dialog box, see Chapter 45, "Setting Options for Individual Tests."

2 Click the **Environment** node.

3 Select **User-defined** from the **Variables type** list.

4 Select the **Load variables and values from external file (reloaded each run session)** check box.

5 Use the browse button or enter the full path of the external environment variables file you want to use with your test. The variables defined in the selected file are displayed in blue in the list of user-defined environment variables.

You can now select the variables in the active file as external user-defined environment parameters in your test. For more information, see "Setting Environment Variable Parameter Options" on page 652.

Using Environment Variable Files with Quality Center

When working with Quality Center and environment variable files, you must save the environment variable file in the Test Resources module in your Quality Center project before you specify the file in the Environment pane of the Test Settings dialog box.

You can add a new or an existing environment variable file to your Quality Center project. Note that adding an existing file from the file system to a Quality Center project creates a copy of the file in Quality Center. Thus, once you save the file to the project, changes made to the Quality Center environment variable file will not affect the file system file and vice versa.

To use an environment variable file with Quality Center:

1. To add a new environment variable file, create a new **.xml** file in your file system, as described in "Using User-Defined External Environment Variables" on page 647.
2. In Quality Center, create a new environment variable resource and then upload the **.xml** 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 QuickTest, connect to the Quality Center project. For more information, see "Connecting to and Disconnecting from Quality Center" on page 1418.
4. In the Test Settings dialog box, click the **Environment** node.
5. Select **User-defined** from the **Variables type** list.
6. Select **Load variables and values from external file (reload each run session)**.
7. In the **File** box, click the browse button to find the user-defined variable file in the Quality Center project.
8. Save your test. QuickTest saves the file to the Quality Center project.

For more information on working with Quality Center, see Chapter 51, "Integrating with Quality Center" and the *HP Quality Center User Guide*.

Using Built-in Environment Variables

QuickTest provides a set of built-in variables that enable you to use current information about the test and the QuickTest computer running your test. These can include the test name, the test path, the operating system type and version, and the local host name.

For example, you may want to perform different checks in your test based on the operating system being used by the computer that is running the test. To do this, you could include the OSVersion built-in environment variable in an If statement.

You can also select built-in environment variables when parameterizing values. For more information, see "Setting Environment Variable Parameter Options" on page 652.

The following built-in environment variables are available:

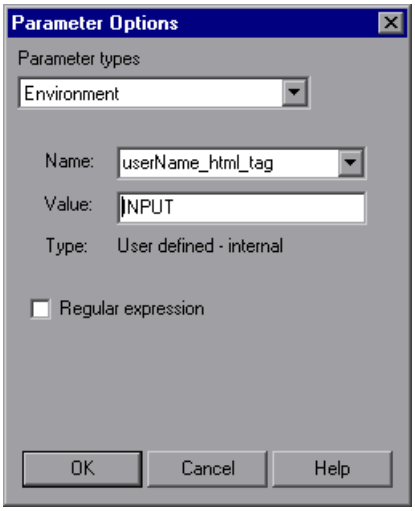
Name	Description
ActionIteration	The action iteration currently running.
ControllerHostName	The name of the controller's computer. This variable is relevant only when running as a GUI Vuser from the LoadRunner controller.
GroupName	The name of the group in the running scenario. This variable is relevant only when running as a GUI Vuser from the LoadRunner controller.
LocalHostName	The local host name.
OS	The operating system.
OSVersion	The operating system version.
ProductDir	The folder path where the product is installed.
ProductName	The product name.
ProductVer	The product version.

Name	Description
ResultDir	The path of the folder in which the current test results are located. **Note:** You cannot use the **ResultDir** environment variable when running a test from Business Availability Center, LoadRunner, or the Silent Test Runner in QuickTest.
ScenarioId	The identification number of the scenario. This variable is relevant only when running as a GUI Vuser from the LoadRunner controller.
SystemTempDir	The system temporary directory.
TestDir	The path of the folder in which the test is located.
TestIteration	The test iteration currently running.
TestName	The name of the test.
UpdatingActiveScreen	Indicates whether the Active Screen images and values are being updated during the update run process. For more information, see "Updating a Test Using the Update Run Mode Option" on page 1125.
UpdatingCheckpoints	Indicates whether checkpoints are being updated during the update run process. For more information, see "Updating a Test Using the Update Run Mode Option" on page 1125.
UpdatingTODescriptions	Indicates whether the set of properties used to identify test objects are being updated during the update run process. For more information, see "Updating a Test Using the Update Run Mode Option" on page 1125.
UserName	The Windows login user name.
VuserId	The Vuser identification under load. This variable is relevant only when running as a GUI VUser from the LoadRunner controller.

Setting Environment Variable Parameter Options

When you choose to parameterize a value, the dialog box that opens enables you to select a parameter type and the parameter options to use. The image below shows the dialog box that opens when you select to parameterize a checkpoint expected value. The dialog boxes for parameterizing other value types such as argument values, object property values, and output storage locations provide similar options.

When you select **Environment** as the parameter type, you can configure your parameter to use values from the Environment variable list.



Defining the Settings for an Environment Variable Parameter

The following options are available for configuring environment variable parameters:

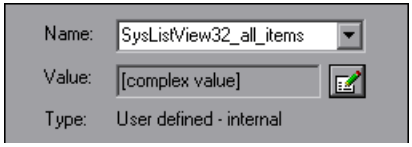
- **Name.** Specifies the name of the parameter. For an internal user-defined environment variable parameter, you can create a new parameter by using the default parameter name or entering a new, descriptive name. Alternatively, you can select an existing internal user-defined environment variable parameter from the list.

Notes:

- If you edit the name displayed in the **Name** box for an existing parameter, you create a new internal user-defined environment variable parameter. The original environment variable parameter is not modified.
- If you are parameterizing an argument that receives a predefined constant or number, only the environment variable parameters whose value is of type **integer** are shown in the **Name** list.

- **Value.** Specifies the value of the parameter. You can enter the value for a new user-defined internal parameter, or modify the value for an existing user-defined internal parameter. External and built-in environment variable parameter values cannot be modified in this dialog box.

  If the entire value of a selected environment variable parameter cannot be displayed in the **Value** box, it is shown as **[complex value]**. For example, the value of a list's **all items** property is a multi-line value, where each line contains the value of an item in the list.



You can view or edit a complex value by clicking the **View/Edit Complex Value** button. For more information, see "Viewing and Editing Complex Parameter Values" on page 654.

- **Type.** Specifies the type of environment variable parameter (read-only):
  - **internal user-defined**
  - **external user-defined**
  - **built-in**

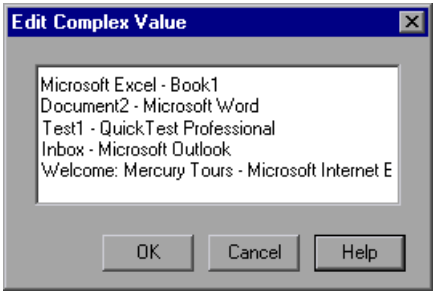
Tip: The value of an environment variable remains the same throughout the test run, regardless of the number of iterations, unless you change the value of the variable programmatically in your script.

➤ **Regular expression.** Sets the value of the parameter as a regular expression. This option is available only when parameterizing a checkpoint or object property text string value, and the selected environment variable parameter type is **internal user-defined**. For more information on regular expressions, see "Understanding and Using Regular Expressions" on page 762.

Note: You can also define environment variables using parameterization objects and methods in the Expert View. For more information, see the *HP QuickTest Professional Object Model Reference*.

Viewing and Editing Complex Parameter Values

When you click the **View/Edit Complex Value** button for a parameter with a value that cannot be displayed entirely in the **Value** box, the Edit Complex Value dialog box displays the full contents of the value.



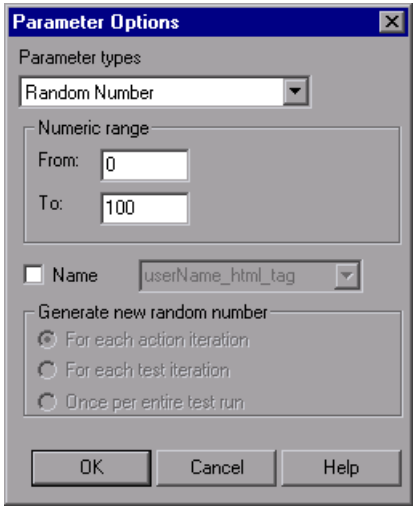
You can edit the value for an internal user-defined environment variable parameter.

For an external or built-in environment variable parameter, you can view the value but you cannot modify it in this dialog box.

Using Random Number Parameters

When you choose to parameterize a value, the dialog box that opens enables you to select a parameter type and the parameter options to use. The image below shows the dialog box that opens when you select to parameterize a checkpoint expected value. The dialog boxes for parameterizing other value types such as argument values, object property values, and output storage locations provide similar options.

When you select **Random Number** as the parameter type, the Parameter Options dialog box enables you to configure your parameter to use random numbers.



Defining Settings for a Random Number Parameter

The following options are available for configuring random number parameters:

- **Numeric range.** Specifies the range from which the random number is generated. By default, the random number range is between 0 and 100. You can modify the range by entering different values in the **From** and **To** boxes. The range must be between 0 and 2147483647 (inclusive).
- **Name.** Assigns a name to your parameter. Assigning a name to a random parameter enables you to use the same parameter several times in your test. You can select an existing named parameter or create a new named parameter by entering a new, descriptive name.
- **Generate new random number.** Defines the generation timing for a named random parameter. This box is enabled when you select the **Name** check box. You can select one of the following options:
  - **For each action iteration.** Generates a new number at the end of each action iteration.
  - **For each test iteration.** Generates a new number at the end of each global iteration.
  - **Once per entire test run.** Generates a new number the first time the parameter is used. The same number is used for the parameter throughout the test run.

Notes:

- Random number parameters are not appropriate for non-numeric values, such as text or hypertext links.
- If you select an existing parameter, then changing the settings in the dialog box affects all instances of that parameter in the test.
- You can also define random number variables using parameterization objects and methods in the Expert View. For more information, see the *HP QuickTest Professional Object Model Reference*.

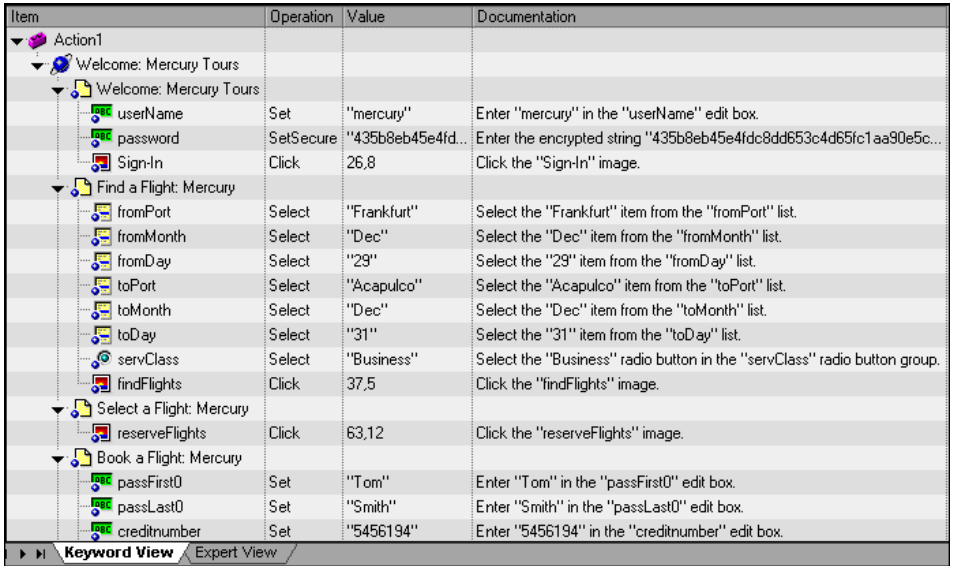
Example of a Parameterized Test

The following example shows how to parameterize a step method and a checkpoint using Data Table parameters.

When you test your application, you may want to check how it performs the same operations with multiple sets of data. For example, if you are testing the Mercury Tours sample Web site, you may want to check that the correct departure and the arrival cities are selected before you book a particular flight.

Suppose that you want to check that the flights are booked correctly for a variety of different locations. Rather than create a separate test with a separate checkpoint for each location, you can parameterize the location information. For each iteration of the test, QuickTest then checks the flight information for the different locations.

The following is a sample test of a flight booking procedure. The departure city is Frankfurt and the arrival city is Acapulco.

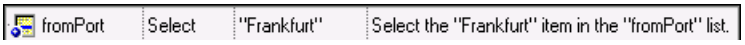


Item	Operation	Value	Documentation
Action1			
Welcome: Mercury Tours			
Welcome: Mercury Tours			
userName	Set	"mercury"	Enter "mercury" in the "userName" edit box.
password	SetSecure	"435b8eb45e4fd...	Enter the encrypted string "435b8eb45e4fdc8dd653c4d65fc1aa90e5c...
Sign-In	Click	26,8	Click the "Sign-In" image.
Find a Flight: Mercury			
fromPort	Select	"Frankfurt"	Select the "Frankfurt" item from the "fromPort" list.
fromMonth	Select	"Dec"	Select the "Dec" item from the "fromMonth" list.
fromDay	Select	"29"	Select the "29" item from the "fromDay" list.
toPort	Select	"Acapulco"	Select the "Acapulco" item from the "toPort" list.
toMonth	Select	"Dec"	Select the "Dec" item from the "toMonth" list.
toDay	Select	"31"	Select the "31" item from the "toDay" list.
servClass	Select	"Business"	Select the "Business" radio button in the "servClass" radio button group.
findFlights	Click	37,5	Click the "findFlights" image.
Select a Flight: Mercury			
reserveFlights	Click	63,12	Click the "reserveFlights" image.
Book a Flight: Mercury			
passFirst0	Set	"Tom"	Enter "Tom" in the "passFirst0" edit box.
passLast0	Set	"Smith"	Enter "Smith" in the "passLast0" edit box.
creditnumber	Set	"5456194"	Enter "5456194" in the "creditnumber" edit box.

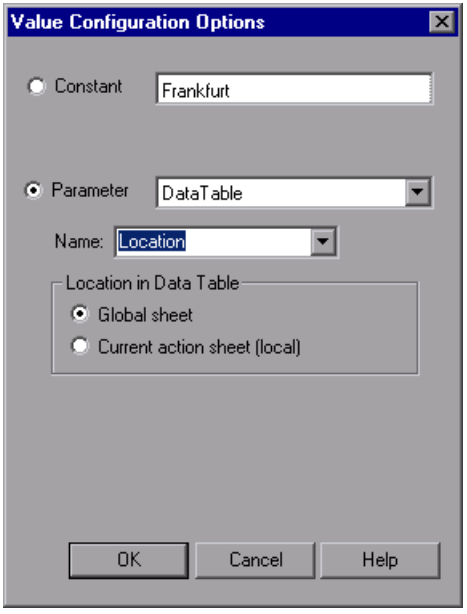
Keyword View | Expert View

Step 1: Parameterize a Step

Parameterize the method argument of the **fromPort** step:



In the Keyword View, click in the **Value** cell of the step and then click the parameterization icon . In the Value Configuration Options dialog box, select the **Parameter** radio button. In the **Name** box, rename p_item to **Location**.



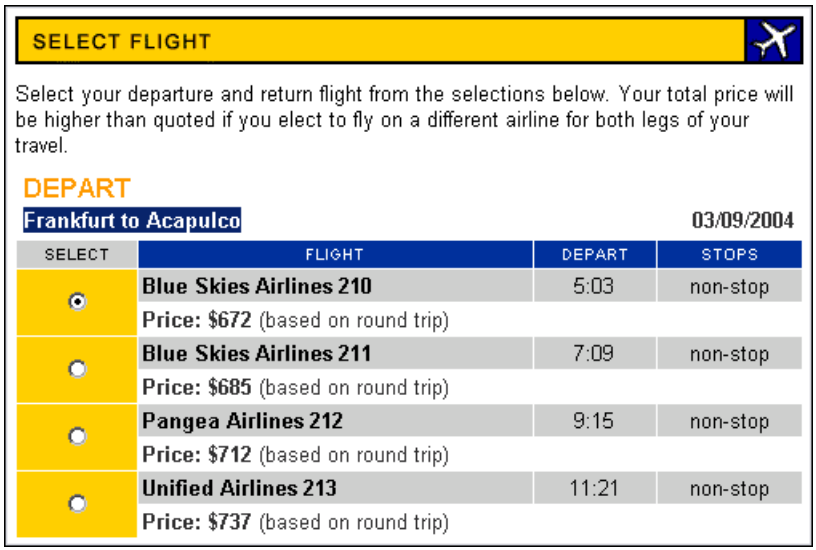
Click **OK**. The **Location** column is added to the Data Table.

For more information on parameterizing a step, see “Parameterizing Values in Steps and Checkpoints” on page 628.

Step 2: Parameterize a Checkpoint

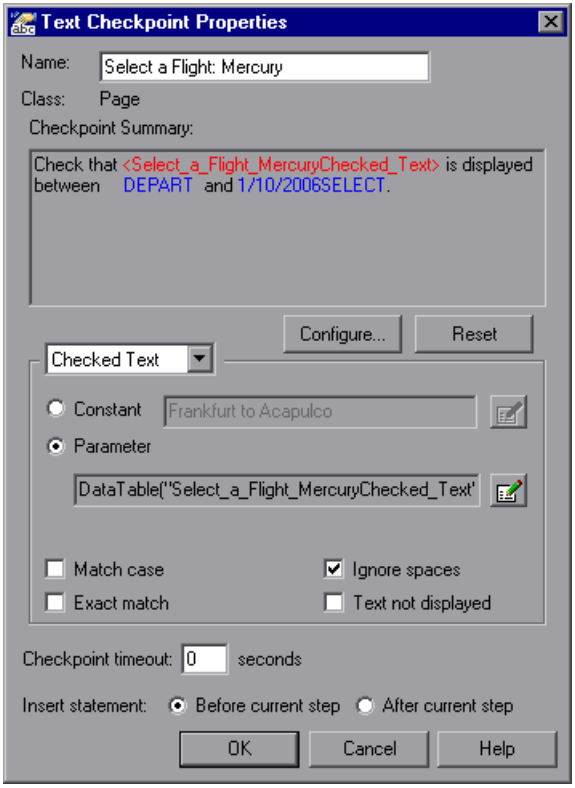
In the following example, you add a parameterized text checkpoint to check that the correct locations were selected before you book a flight.

Select the Select a Flight step. In the Active Screen, highlight the text Frankfurt to Acapulco, right-click and insert a text checkpoint:



In the Text Checkpoint Properties dialog box, select **Parameter** to parameterize the selected text. Select the **Parameter** radio button and click the **Parameter Options** button.

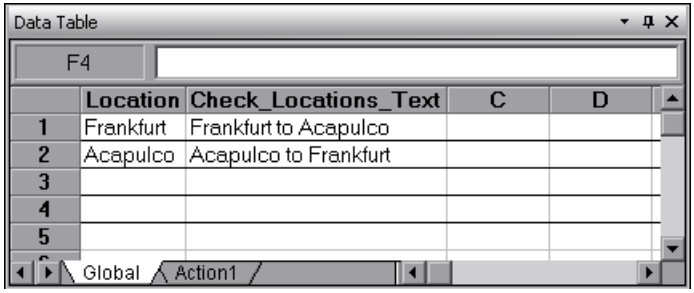
In the Parameter Options dialog box, rename the Data Table parameter to Check_Locations_Text. Click **OK** in the Parameter Options dialog box and in the Text Checkpoint Properties dialog box. A **Check_Locations_Text** column is added to the Data Table.



For more information on parameterizing a checkpoint, see “Parameterizing Values in Steps and Checkpoints” on page 628.

Step 3: Enter Data in the Data Table

Complete the Data Table. The Data Table may be displayed as follows:



For more information on Data Tables, see Chapter 42, “Working with Data Tables.”

Modified Test

The following example shows the test after parameterizing the step and creating a parameterized text checkpoint.

Welcome: Mercury T...			
userName	Set	"mercury"	Enter "mercury" in the "userName" edit box.
password	SetS...	"404ee5e998b25bdc6f116b031452...	Enter the encrypted string "404ee5e998b25bdc6f116b
Sign-In	Click	2,2	Click the "Sign-In" image.
Find a Flight: Mercury			
fromPort	Select	DataTable("Location", dtGlobalSheet)	Select the <the value of the specified Data Table colu
toPort	Select	"Acapulco"	Select the "Acapulco" item in the "toPort" list.
toMonth	Select	"Jun"	Select the "Jun" item in the "toMonth" list.
findFlights	Click	2,2	Click the "findFlights" image.
Select a Flight: Mercury	Check	CheckPoint("Frankfurt to Acapulco")	Check whether text in the "Select a Flight: Mercury" W
reserveFlights	Click	2,2	Click the "reserveFlights" image.
Book a Flight: Mercury			
passFirst0	Set	"John"	Enter "John" in the "passFirst0" edit box.
passLast0	Set	"Brown"	Enter "Brown" in the "passLast0" edit box.
creditnumber	Set	"333666777"	Enter "333666777" in the "creditnumber" edit box.

The parameterized value for the fromPort step is clearly shown as a Data Table parameter. To see the parameterization setting for the checkpoint, click in the **Value** column for the Select a Flight step.

Using the Data Driver to Parameterize Your Test

The Data Driver enables you to quickly parameterize several (or all) property values for test objects, checkpoints, and/or method arguments containing the same constant value within a given action.

You can choose to replace all occurrences of a selected constant value with a parameter, in the same way that you can use a **Find and Replace All** operation instead of a step-by-step **Find and Replace** process. QuickTest can also show you each occurrence of the constant so that you can decide whether or not to parameterize the value.

Notes:

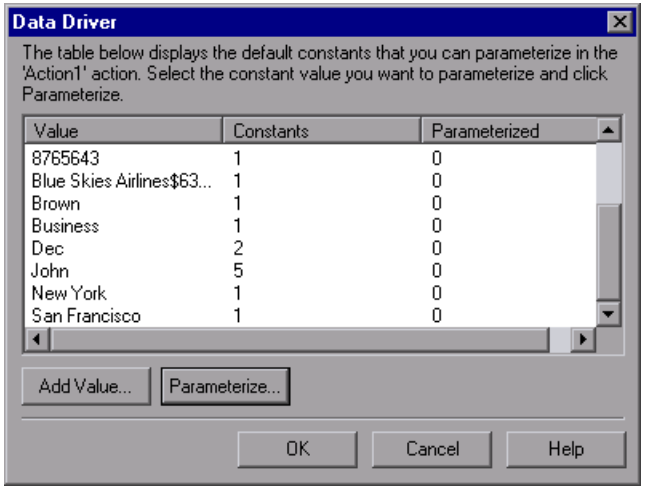
- When finding multiple occurrences of a selected value, QuickTest conducts a search that is case sensitive and searches only for exact matches. (It does not find values that include the selected value as part of a longer string.)
- You cannot use the Data Driver to parameterize the values of arguments for user-defined methods or VBScript functions.

To parameterize a value using the Data Driver:

1 Display the action you want to parameterize.

2 Select **Tools** > **Data Driver**.

QuickTest scans the test for constants before the Data Driver opens (this may take a few moments).



Note: If the action being scanned contains a large number of lines and constant values, QuickTest warns you that loading the constants may take some time. You can choose whether to wait for the constants to load, or to open the Data Driver wizard quickly without constants.

The Data Driver displays the Constants list for the action. For each constant value, it displays the number of times the constant value appears in the action.

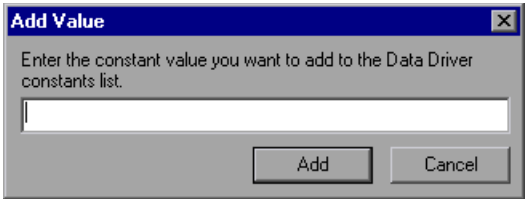
By default, the list displays the constants for one or more of the arguments of the following methods: **Activate**, **Collapse**, **Deselect**, **Expand**, **ExtendSelect**, **Press**, **Select**, **SelectColumn**, **SelectRange**, **SelectRow**, **Set**, **SetCellData**, **SetSecure**, **SetText**, **Type**, and **WaitProperty**.

For more information on how to work with testing methods, see Chapter 29, "Working in the Expert View and Function Library Windows." For syntax and method information, see the *HP QuickTest Professional Object Model Reference*.

Note: If you chose not to wait for the constants to load, the Data Driver opens with an empty Constants table. You can add the constant values that you want to parameterize to the Data Driver, as described below.

---

3 If you want to parameterize a value that is not currently displayed in the list (such as an object property value), click **Add Value**. The Add Value dialog box opens.

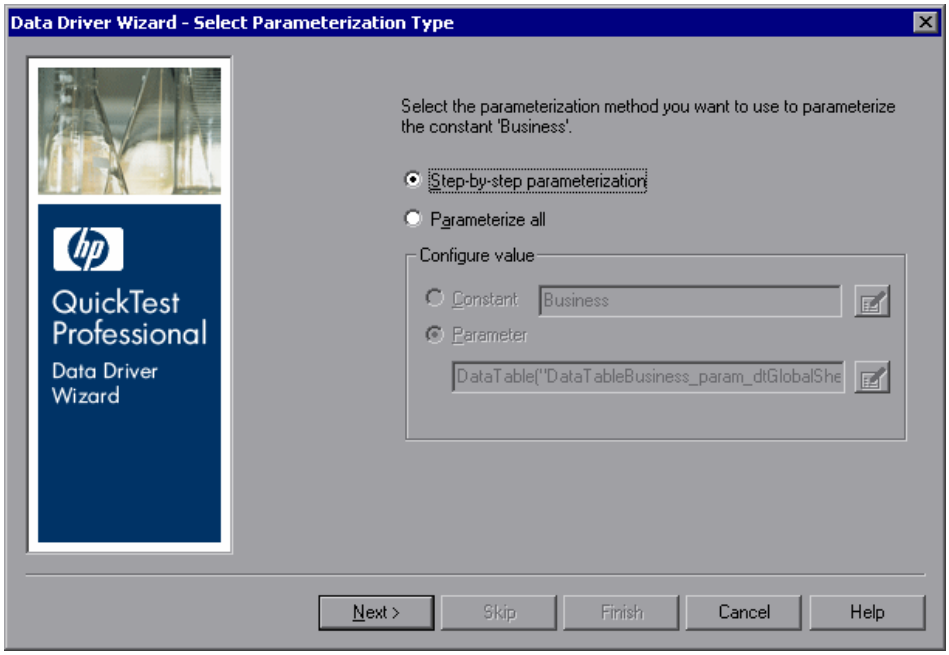


Enter a constant value in the dialog box and click **Add**. The constant is added to the list.

Note: You can add only constant values that currently exist in the test action.

---

4 Select the value you want to parameterize from the Constants list and click **Parameterize**. The Data Driver Wizard opens.



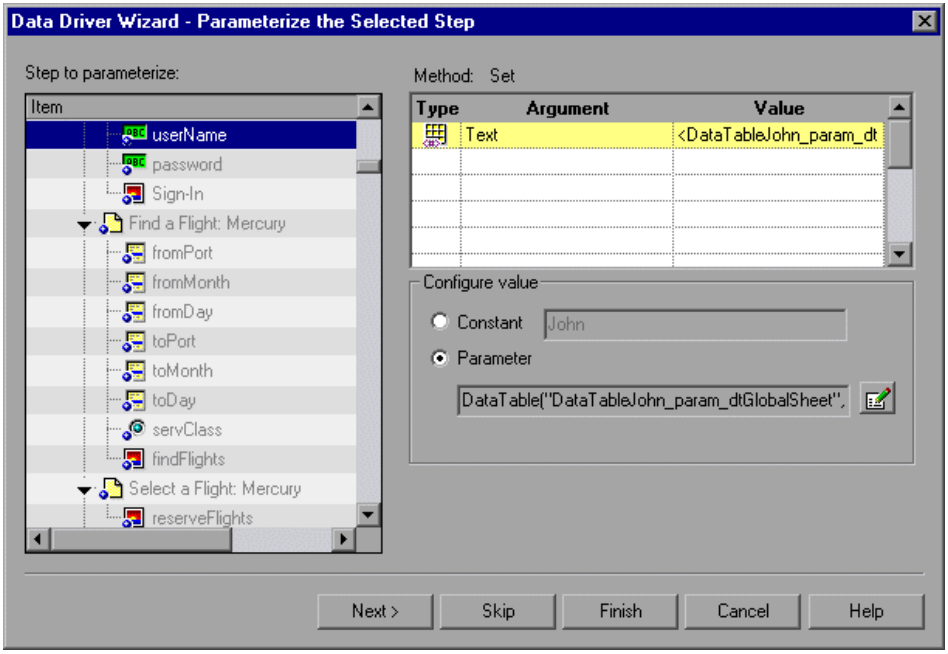
5 Select the type of parameterization you want to perform:

- **Step-by-step parameterization.** Enables you to view the current values of each step containing the selected value. For each step, you can choose whether or not to parameterize the value and if so, which parameterization options you want to use.
- **Parameterize all.** Enables you to parameterize all occurrences of the selected value throughout the action. You set your parameterization preferences one time and the same options are applied to all occurrences of the value.

6 If you selected **Step-by-step parameterization**, click **Next**. The Parameterize the Selected Step screen opens.

If you selected **Parameterize all**, the **Parameter** option is enabled in the **Configure value** area. Select your parameterization preferences the same way that you would for an individual step. For more information, see "Parameterizing Values in Steps and Checkpoints" on page 628. Proceed to step 9.

7 In the **Step to parameterize** area, the first step with an object property or checkpoint value containing the selected value is displayed in the test tree on the left. The parameterization options for the step are displayed on the right.



The default parameterization settings are displayed for the value. For more information on default parameterization settings, see "Understanding Default Parameter Values" on page 634.

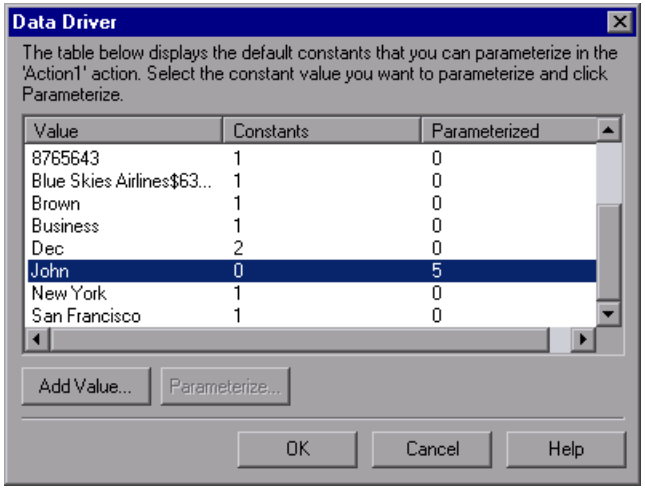
Accept the default parameterization settings or click the **Parameter Options** button to set the parameterization options you want to apply to this step. For more information, see "Parameterizing Values in Steps and Checkpoints" on page 628.

- Click **Next** to parameterize the selected step and view the next step containing the selected value.
- Click **Skip** if you do not want to parameterize the selected step.
- Click **Finish** to apply the parameterization settings of the current step to all remaining steps containing the selected value.

8 If you clicked **Next** in the previous step, and steps remain that contain the selected value, the Parameterize the Selected Step screen opens displaying the next relevant step. Repeat step 7 for each relevant step.

If there are no remaining steps containing the selected value, the Finished screen opens.

9 Click **Finish**. The Data Driver Wizard closes and the Data Driver main screen shows how many occurrences you selected to parameterize and how many remain as constants.



10 If you want to parameterize another constant value, select the value and repeat steps 4 to 9.

11 When you are finished parameterizing constants, click **OK**. The parameterization options you selected are applied to your action.

25

Outputting Values

QuickTest enables you to retrieve values in your test and store them in output value objects. You can subsequently retrieve these values and use them as input at a different stage in the run session.

This chapter includes:

About Outputting Values

An **output value** step is a step in which one or more values are captured at a specific point in your test 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.

You can output the property values of any object. You can also output values from text strings, table cells, databases, and XML documents.

When you create output value steps, you can determine where the values are stored during the run session and how they can be used. During the run session, QuickTest retrieves each value at the specified point and stores it in the specified location. When the value is needed later in the run session, QuickTest retrieves it from this location and uses it as required.

Output values are stored only for the duration of the run session. When the run session is repeated, the output values are reset.

Note: After the run session, you can view the output values retrieved during the session as part of the session results. For more information, see "Viewing Parameterized Values and Output Value Results" on page 1053.

Creating Output Values

When you add an output value step to your test, you first select the category of values to output, for example, property values, text values, or XML element values. For more information, see Output Value Categories.

You can then determine which values to output. For more information, see "Viewing and Editing Output Values" on page 675.

You also determine the storage location for each value. For more information, see "Storing Output Values" on page 673.

Output Value Categories

You can create the following categories of output values:

- Standard output values
- Text and text area output values
- Table output values
- Database output values
- XML output values

Standard Output Values

You can use standard output values to output the property values of most objects. For example, in a Web-based application, the number of links on a Web page may vary based on the selections a user makes on a form on the previous page. You could create an output value in your test to store the number of links on the page.

Note: You can also use standard output values to output the contents of table cells. For more information, see "Table Output Values" on page 672.

Tip: You can use standard output values to output text strings by specifying the **text** property of the object as an output value.

For more information on standard output values, see "Outputting Property Values" on page 676.

Text and Text Area Output Values

You can use text output values to output text strings displayed in an application. When creating a text output value, you can output a part of the object's text. You can also specify the text before and after the output text.

You can use text area output values to output text strings displayed within a defined area of a screen in a Windows-based application.

For example, suppose that you want to store the text of any error message that appears after a specific step in the Web application you are testing. Inside the If statement, you check whether a window exists with a known title bar value, for example Error. If it exists, you output the text in this window (assuming that the window size is the same for all possible error messages).

For more information on text output values, see "Outputting Text Values" on page 688. For more information on text area output values, see "Creating Text Area Output Values" on page 690.

Table Output Values

Table output values are a subset of standard output values, described above. You can use table output values to output the contents of table cells. For some types of tables, you can specify a row range from which to choose the table cells. During the run session, QuickTest retrieves the current data from the specified table cells according to the settings that you specified and outputs the values to the Data Table.

For more information, see "Outputting Table Values" on page 698.

Database Output Values

You can use database output values to output the value of the contents of database cells, based on the results of a query (result set) that you define on a database. You can create output values from the entire contents of the result set, or from a part of it. During the run session, QuickTest retrieves the current data from the database and outputs the values according to the settings that you specified.

For more information, see "Outputting Database Values" on page 713.

XML Output Values

You can use XML output values to output the values of XML elements and attributes in XML documents.

After the run session has finished, you can view summary results of the XML output values in the Test Results window. You can also view detailed results by opening the XML Output Value Results window. For more information, see Chapter 33, "Viewing Run Session Results."

For example, suppose that an XML document in a Web page contains a price list for new cars. You can output the price of a particular car by selecting the appropriate XML element value to output.

For more information on XML output values, see "Outputting XML Values" on page 718.

Output Value Categories and Environments

QuickTest add-ins help you to create and run tests and components on applications in a variety of development environments. For information about using output values for each add-in environment installed with QuickTest Professional, see "Supported Output Values" on page 1548.

Storing Output Values

When you define an output value, you can specify where and how each value is stored during the run session.

You can output a value to:

- a test or action parameter
- the run-time Data Table
- an environment variable

Note: Output values are stored only for the duration of the test, and are not saved with the test. If you select to output a value to an existing parameter, Data Table column, or environment variable, the existing value is overwritten when the output value step runs. When the run session ends, the original value is restored.

Storing Values in Test and Action Parameters

You can output a value to an action parameter, so that values from one part of a run session can be used later in the run session, or be passed back to the application that ran (called) the test.

For example, suppose you are testing a shopping application that calculates your purchases and automatically debits your account with the amount that you purchased. You want to test that the application correctly debits the purchase amount from the account each time that the action is run with a different list of items to purchase. You could output the total amount spent to an action parameter value, and then use that value later in your run session in the action that debits the account.

For more information on action parameters in general, see "Using Action Parameters" on page 476.

Storing Values in the Run-time Data Table

The option to output a value to the run-time Data Table is especially useful with a **data-driven** test (or action) that runs several times. In each repetition, or **iteration**, QuickTest retrieves the current value and stores it in the appropriate row in the run-time Data Table.

For example, suppose you are testing a flight reservation application and you design a test to create a new reservation and then view the reservation details. Every time you run the test, the application generates a unique order number for the new reservation. To view the reservation, the application requires the user to input the same order number. You do not know the order number before you run the test.

To solve this problem, you output a value to the Data Table for the unique order number generated when creating a new reservation. Then, in the View Reservation screen, you use the column containing the stored value to insert the output value into the order number input field.

When you run the test, QuickTest retrieves the unique order number generated by the site for the new reservation and enters this output value in the run-time Data Table. When the test reaches the order number input field required to view the reservation, QuickTest inserts the unique order number stored in the run-time Data Table into the order number field.

Storing Values in Environment Variables

When you output a value to an internal user-defined environment variable, you can use the environment variable input parameter at a later stage in the run session.

Note: You can output values only to internal user-defined environment variables and not to external or built-in environment variables, which are read-only.

For example, suppose you are testing an application that prompts the user to input an account number on a Welcome page and then displays the user's name. You can use a text output value to capture the value of the displayed name and store it in an environment variable.

You can then retrieve the value in the environment variable to enter the user's name in other places in the application. For example, in an Order Checkbook Web page, which for security reasons requires users to enter the name to appear on the checks, you could use the value to insert the user's name into the **Name** edit box.

Viewing and Editing Output Values

When you insert an output value step in your test, the Keyword View shows the step with Output displayed in the **Operation** column and CheckPoint displayed in the **Value** column, followed by the name assigned to the output value.

The output value statement is displayed in the Expert View with the following syntax:

Object.Output CheckPoint(*Name*)

You can view or edit the output value or its details in the relevant Output Value Properties dialog box, by right-clicking the step and choosing **Output Value Properties**. Alternatively, you can click the step in the **Value** column in the Keyword View and then click the **Output Properties** button.

For more information on the options available in the different Output Value Properties dialog boxes, see:

- "Defining Standard Output Values" on page 679
- "Defining Text and Text Area Output Values" on page 692
- "Outputting Table Content" on page 703
- "Outputting Table Properties" on page 709
- "Defining Database Output Values" on page 715
- "Understanding the XML Output Properties Dialog Box" on page 727

Outputting Property Values

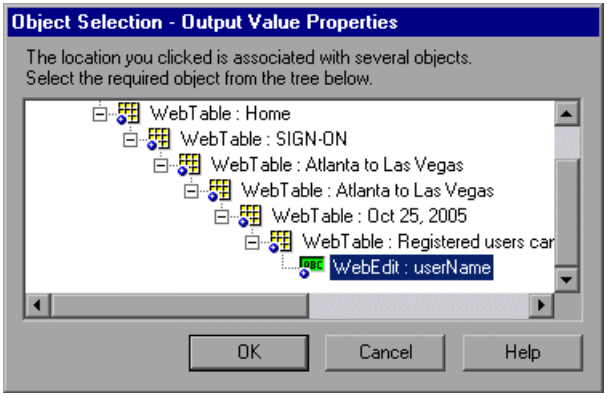
You can use standard output values to output the property values of most objects. You can also use standard output values to output the contents of table cells.

You can create standard output values while recording or editing your test.

To create standard output values while recording:

1 Select **Insert** > **Output Value** > **Standard Output Value**. Alternatively, you can click the arrow beside the **Insert Checkpoint or Output Value** button in the toolbar and select **Standard Output Value**. The pointer changes into a pointing hand. For more information on using the pointing hand feature, see “Tips for Using the Pointing Hand” on page 677.

2 In your application, click the object for which you want to specify an output value. If the location you clicked is associated with more than one object, the Object Selection – Output Value Properties dialog box opens.



3 In the Object Selection dialog box, select the object for which you want to specify an output value, and click **OK**. The Output Value Properties dialog box opens for the selected object. If you select a **Table** item, the Table Output Value Properties dialog box opens.

4 Specify the property values to output and their settings. For more information, see "Defining Standard Output Values" on page 679. If you selected a **Table** item, see "Outputting Table Content" on page 703 and "Outputting Table Properties" on page 709.

5 When you finish defining the output value details, click **OK**. QuickTest inserts an output value step in your test.

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.

To create standard output values while editing your test:

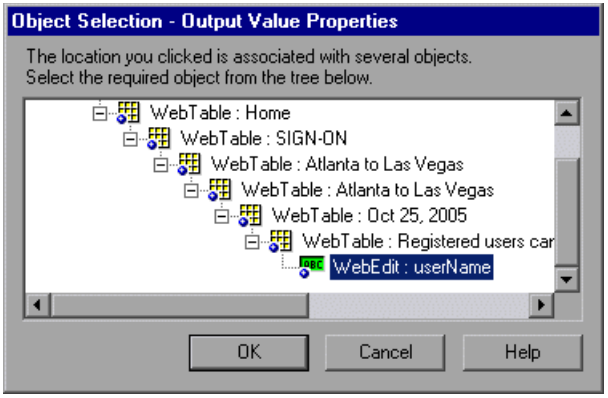
1 Make sure the **Active Screen** button is selected.

2 Click a step whose Active Screen contains the object for which you want to specify an output value. The Active Screen displays the captured bitmap or HTML source corresponding to the highlighted step.

For Windows-based applications, make sure that the Active Screen contains property data for the object for which you want to specify an output value. For more information, see "Setting Active Screen Options" on page 1240.

3 In the Active Screen, right-click the object for which you want to specify an output value and select **Insert Output Value**. Alternatively, you can right-click the step in your test and select **Insert Output Value**.

4 If the location you clicked is associated with more than one object, the Object Selection – Output Value Properties dialog box opens.



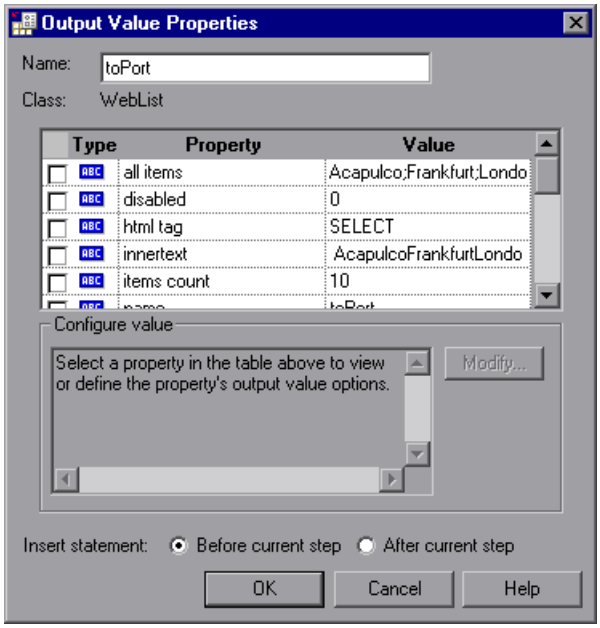
5 Select the object for which you want to specify an output value, and click **OK**. The Output Value Properties dialog box opens for the selected object. If you select a **Table** item, the Table Output Value Properties dialog box opens.

6 Specify the property values to output and their settings. For more information, see "Defining Standard Output Values" on page 679. If you selected a **Table** item, see "Outputting Table Content" on page 703 and "Outputting Table Properties" on page 709.

7 When you finish defining the output value details, click **OK**. QuickTest inserts an output value step in your test.

Defining Standard Output Values

The Output Value Properties dialog box enables you to choose which property values to output and to define the settings for each value that you select.



Note: If you insert an output value on a Web page, the Page Output Value Properties dialog box opens. This dialog box is identical to the Output Value Properties dialog box, except that it contains two additional option areas, **HTML verification** and **All objects in page**. These options are relevant only for checkpoints and are disabled when defining output values.

You can select a number of properties to output for the same object and define the output settings for each property value before closing the dialog box. When the output value step is reached during the run session, QuickTest retrieves all of the specified property values.

Identifying the Output Value

The top part of the dialog box displays information on the output value:

Item	Description
Name	The name that QuickTest assigns to the output value. By default, the output value name is the name of the test object for which you are performing the output value step. You can specify a different name for the output value or accept the default name. If you rename the output value, make sure that the name: ➤ is unique ➤ does not begin or end with a space ➤ does not contain " (double quotation mark) ➤ does not contain the following character combinations: := @@
Class	The type of test object. In this example, the WebList class indicates it is a list object in a Web application.
Find in Repository button  (Located to the right of the **Name** box)	Displays the output value in its repository. **Note:** This option is not available when creating a new output value. It is available only when editing an existing output value.

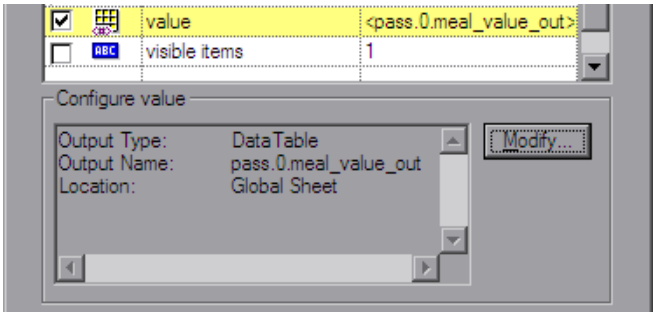
Selecting the Property Values to Output

The upper part of the dialog box contains a pane that lists the properties of the selected object, with their values and types. This pane contains the following items:

Pane Element	Description
Check box	To specify a property to output, select the corresponding check box. You can select more than one property for the object and specify the output options for each property value you select.
Type	The  icon indicates that the value of the property is currently a constant. The  icon indicates that the value of the property is currently stored in a test or action parameter. The  icon indicates that the value of the property is currently stored in the run-time Data Table. The  icon indicates that the value of the property is currently stored in an environment variable.
Property	The name of the property.
Value	The current value of the property. For more information, see “Specifying the Output Settings for a Property Value” on page 682.

Specifying the Output Settings for a Property Value

When you select a check box for a property, the property details are highlighted and the current output definition for the selected property value is displayed in the **Configure value** area.



When a property value is first selected for output, the default output definition for the value is displayed in the **Configure value** area. For more information on default output definitions, see "Understanding Default Output Definitions" on page 683.

When you select a property value to output, you can:

- Change the output type and/or settings for the selected value by clicking the **Modify** button. The Output Options dialog box opens and displays the current output type and settings for the value. For more information, see "Specifying the Output Type and Settings" on page 683.
- Accept the displayed output definition by selecting another property value or by clicking **OK**.

Specifying the Location for the Output Value Step

If the **Insert statement** area is displayed at the bottom of the dialog box, you can specify where the new output value step should be inserted in your test. For more information, see "Selecting the Location for the Output Value Step" on page 687.

Specifying the Output Type and Settings

The output type and settings that you define for each value determine where it is stored and how it can be used during the run session. When the output value step is reached, QuickTest retrieves each value selected for output and stores it in the specified location for use later in the run session.

When you create a new output value step, QuickTest assigns a default definition to each value selected for output. For more information, see "Understanding Default Output Definitions" on page 683.

You can change the current output definition for the selected value by selecting a different output type and/or changing the output settings in the Output Value Properties dialog box.

Understanding Default Output Definitions

When you initially select a value for output, QuickTest generates a default output definition for the value.

When you output a value for a step in a test action:

- If at least one output parameter is defined in the action, the default output type is **Test/action parameter** and the default output name is the first output parameter displayed in the Action Properties dialog box.
- If no output parameters are defined in the action, the default output type is Data Table and QuickTest creates a new Data Table output name based on the selected value.

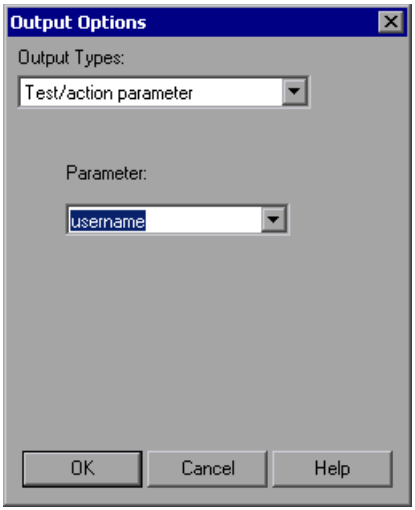
  The output value is created in the Global sheet of the Data Table. For more information on creating output parameters for actions, see "Removing Actions from a Test" on page 460.

  For more information on Data Table sheets, see Chapter 42, "Working with Data Tables."

Outputting a Value to an Action Parameter

You can output a value to an action parameter, so that the values can be used later in the run session, or the values can be passed back to the external application that ran (called) the test. You can only output a value to an action parameter if the parameter has been defined as an output parameter for the calling action. You open the Output Options dialog box by clicking the **Modify** button in any Output Value Properties dialog box.

When **Test/action parameter** is selected as the output type, the Output Options dialog box enables you to select the parameter in which to store the selected value for the duration of the run session.

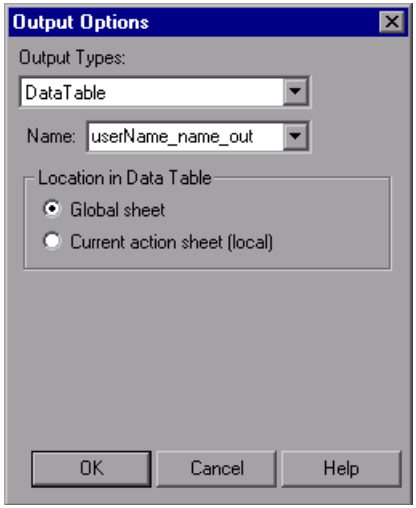


Tip: When you open the Output Options dialog box, QuickTest may display **Test/action parameter** as the default output type. This occurs if at least one output parameter is defined in the action.

The **Parameter** box specifies the name of the parameter in which to store the output value. The read-only list of available parameters contains the names and full descriptions of the currently defined output parameters for the action. You can resize the display, as needed, and, if the list of parameters is long, you can scroll through the list.

Outputting a Value to the Data Table

When **Data Table** is selected as the output type, the Output Options dialog box enables you to specify where to store the selected value within the run-time Data Table. You open the Output Options dialog box by clicking the **Modify** button in any Output Value Properties dialog box.



Tip: When you open the Output Options dialog box, QuickTest may display **Data Table** as the default output type. For more information, see "Understanding Default Output Definitions" on page 683.

The following options are available when outputting a value to the Data Table:

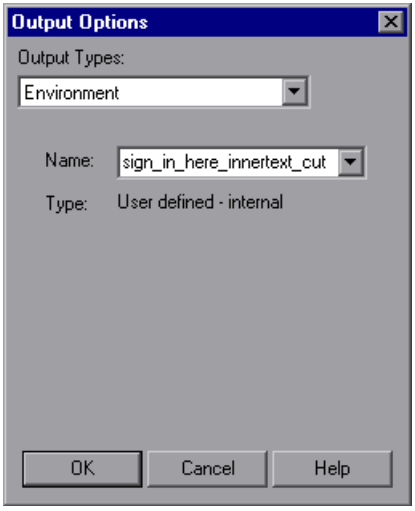
➤ **Name.** Specifies the name of the column in the Data Table in which to store the value. QuickTest suggests a default name for the output. You can select an existing output name from the list, or create a new output name by using the default output name or entering a valid descriptive name.

You can define a new name containing letters, numbers, periods, and underscores. The first character of the output name must be a letter or an underscore. The output name must be unique in the Data Table sheet.

➤ **Location in Data Table.** When outputting values for a test, specifies whether to add the Data Table column name in the global or current action sheet in the Data Table. For more information on the use of data in the global and current action sheets, see "Using Global and Action Data Sheets" on page 429. For more information on actions, see Chapter 15, "Working with Actions" and Chapter 16, "Working with Advanced Action Features."

Outputting a Value to an Environment Variable

When you select **Environment** as the output type, the Output Options dialog box enables you to specify the internal user-defined environment variable in which to store the selected value for the duration of the run session. You open the Output Options dialog box by clicking the **Modify** button in any Output Value Properties dialog box.



The following options are available when outputting a value to an Environment variable:

➤ **Name.** Specifies the name of the internal user-defined environment variable in which to store the value. The list contains all currently defined internal user-defined environment variables with the corresponding type. You can select an existing variable from the list, or you can create a new internal environment variable by modifying the displayed name or by entering a new, descriptive name.

Note: If you edit the name displayed in the **Name** box for an existing variable, you create a new internal user-defined environment variable. The original environment variable is not modified.

Alternatively, you can output the value to an existing environment variable. If you select an existing variable from the list, QuickTest prompts you to choose whether to overwrite its current value with the new value when the output value step runs.

If you choose not to overwrite the current value of the selected variable, a new environment variable is created with the original variable name and an identifying suffix.

➤ **Type.** Displays the environment variable type. Since it is not possible to output values to external or built-in environment variables, the type is always **User-defined - internal**.

For more information on environment variables, see "Using Environment Variable Parameters" on page 645.

Selecting the Location for the Output Value Step

When you create output values while editing a test, the **Insert statement** area is displayed at the bottom of the dialog box.

By default, QuickTest inserts the new output value step before the current step (the step you selected when you chose the **Output Value** option). You can instruct QuickTest to insert the new output value step after the current step, by selecting the **After current step** option.

Note: This option is not available while recording. QuickTest automatically inserts the new output value step after the previously recorded step. It is also not available when modifying an existing output value step.

Outputting Text Values

You can create a text output value from a text string displayed in an application. You can define the output value as part of the displayed text, and specify the text before and/or after the output text.

You can also create a text output value from defined text areas. For more information, see "Creating Text Area Output Values" on page 690.

Note: Before you create a text / text area output value, make sure you configure the required capture settings in the General > Text Recognition pane (**Tools** > **Options** > **Text Recognition** node). For more information, see "The Options Dialog Box: General > Text Recognition Pane" on page 742 and "About Working with Text Recognition for Windows-Based Objects" on page 742.

Creating Text Output Values

You can create a text output value while recording or editing your test.

Note: Before you create a text output value, make sure you configure the required capture settings in the General > Text Recognition pane (**Tools** > **Options** > **Text Recognition** node). For more information, see "The Options Dialog Box: General > Text Recognition Pane" on page 742 and "About Working with Text Recognition for Windows-Based Objects" on page 742.

To create a text output value while recording:

1. Highlight or display the text string you want to use for an output value.
2. Select **Insert** > **Output Value** > **Text Output Value**. The pointer changes into a pointing hand. For more information on using the pointing hand feature, see "Tips for Using the Pointing Hand" on page 689.
3. In your application, click the text string for which you want to specify a text output value. The Text Output Value Properties dialog box opens.

4 Specify the settings for the output value. For more information, see "Defining Text and Text Area Output Values" on page 692.

5 When you finish defining the text output value details, click **OK**. QuickTest inserts an output value step in your test.

To create a text output value when editing your test:

1 Make sure the **Active Screen** is displayed.

2 Click a step in your test where you want to create an output value. The Active Screen displays the screen corresponding to the highlighted step.

3 In the Active Screen, highlight or display the text string you want to specify as an output value.

4 Right-click and select **Insert Text Output**. The Text Output Value Properties dialog box opens.

5 Specify the settings for the output value. For more information, see "Defining Text and Text Area Output Values" on page 692.

6 When you finish defining the output value details, click **OK**. QuickTest inserts an output value step in your test.

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.

Creating Text Area Output Values

You can create a text area output value from a text string displayed in a defined area of a screen in a Windows-based application. You can define the output value as part of the displayed text, and you can specify the text before and/or after the output text. You can create a text area output value only while recording on Windows-based applications.

When you use text-area selection to capture text displayed in a Windows application, it is often advisable to define a text area larger than the actual text you want QuickTest to use as an output value. When QuickTest runs your test, it outputs the selected text, within the defined area, according to the settings you configured.

Because text may change its position during test runs, you must make sure that the area defined is large enough so that the output text is always within its boundaries. For more information, see "About Working with Text Recognition for Windows-Based Objects" on page 742.

Note: Before you create a text area output value, make sure you configure the required capture settings in the General > Text Recognition pane (**Tools** > **Options** > **Text Recognition** node). For more information, see "The Options Dialog Box: General > Text Recognition Pane" on page 742.

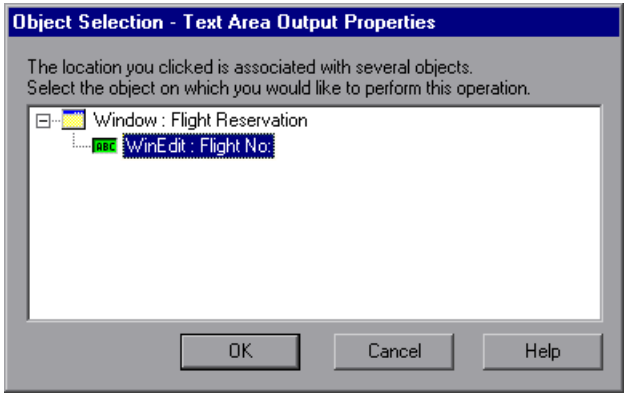
To create a text area output value:

1 While recording, select **Insert** > **Output Value** > **Text Area Output Value**. The QuickTest window is hidden, and the mouse pointer turns into a crosshairs pointer.

2 Define the area containing the text you want QuickTest to use as an output value by clicking and dragging the crosshairs pointer. Release the mouse button after outlining the required area.

Tip: Hold down the left mouse button and use the arrow keys to make precise adjustments to the defined area.

If the area you defined is associated with more than one object, the Object Selection – Text Area Output Properties dialog box opens.



3 Select the object for which you are creating the output value. The Text Area Output Value Properties dialog box opens.

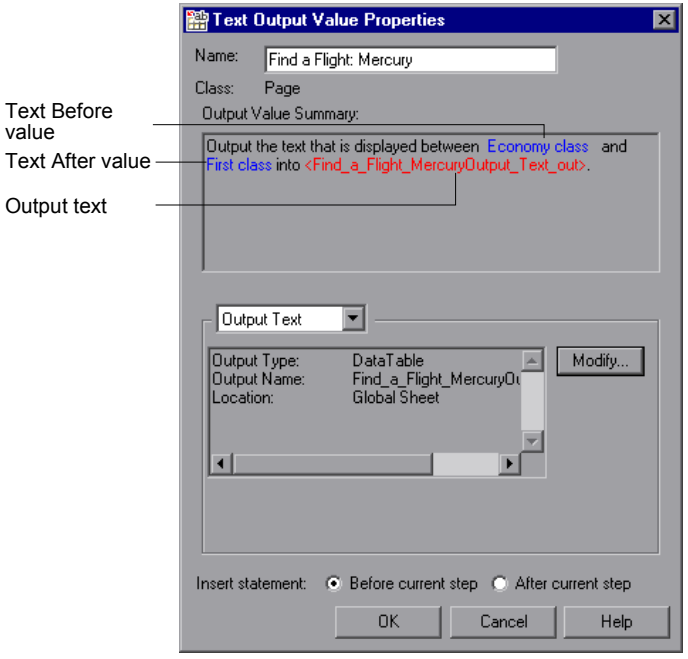
4 Specify the settings for the output value. For more information, see "Defining Text and Text Area Output Values" on page 692.

5 When you have finished defining the output value details, click **OK**. QuickTest inserts an output value step in your test.

Defining Text and Text Area Output Values

You can specify a text string as an output value. You can also specify the text that is displayed before and after the output value text string. This is helpful when the text string you want to specify as an output value is displayed several times in the defined screen area or when the text could change in a predictable way during a run session.

The Text Output Value Properties and Text Area Output Value Properties dialog boxes enables you to define the output value settings for the selected text string, and to define the options for the text displayed before and after the output value.



The top of the Text Output Value Properties dialog box displays the name of the output value and the class of the test object on which the output value check is being performed. You can modify the output value name, if required. For more information, see “Identifying the Output Value” on page 680.

The **Output Value Summary** pane at the top of the dialog box describes the text string for the output value. The text string is the string displayed between the **Text Before** value and the **Text After** value. This pane also shows the output name assigned to the text string. QuickTest automatically displays the text output in red, and the text before and after the text output in blue. For example, in the dialog box displayed above, the output value is the text displayed between **Economy class** (the **Text Before** value) and **First class** (the **Text After** value).

For a text area output value, the output value string contains all the text in the selected area. Although the Text Output Value Properties and Text Area Output Value Properties dialog boxes are identical, when you create a text area output value, the **Text Before** and **Text After** values are not captured.

When you create a text or text area output value, you can specify the captured text as an output value. You can also specify options for **Text Before** and **Text After** values. For example, you can define these values as parameters. If the specified text is displayed more than once in the selected object or area, you can specify the exact occurrence that relates to the output value. If you are editing your test, you can also specify the location for the output value step.

Identifying the Output Value

The top part of the Table Output Value Properties dialog box contains the following options:

Name	The name that QuickTest assigns to the output value. By default, the output value name is the name of the test object for which you are performing the output value step. You can specify a different name for the output value or accept the default name. If you rename the output value, make sure that the name: ➤ is unique ➤ does not begin or end with a space ➤ does not contain " (double quotation mark) ➤ does not contain the following character combinations: := @@
Class	Specifies the type of object (read-only). This may be a table-type object or a list view-type object.
Find in Repository button 	Displays the output value in its repository. **Note:** This option is not available when creating a new output value. It is available only when editing an existing output value.

Specifying the Captured Text as an Output Value

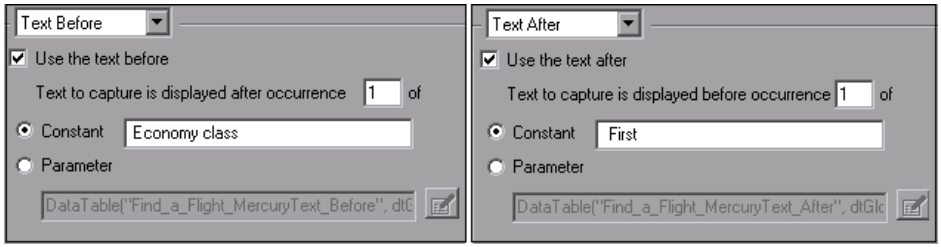
By default, **Output Text** is selected in the list box in the middle of the dialog box. The area below the list box displays the current output value settings for the selected text.

When you create a new output value, the default output definition is displayed for the value. For more information, see "Understanding Default Output Definitions" on page 683.

You can accept the displayed output definition, or you can click **Modify** to specify the output settings for the selected text. For more information, see "Specifying the Output Type and Settings" on page 683.

Specifying Options for the Text Before/Text After Values

When you select **Text Before** or **Text After** from the list box, you can define the options for the text displayed before or after the output value string.



Option	Description
Use the text before / Use the text after	When selected, the current **Text Before / Text After** value is displayed in the **Constant** box. When cleared, QuickTest retrieves the value of the first occurrence of the defined output string, regardless of the text displayed before it (if you chose **Text Before**) or after it (if you chose **Text After**). **Note:** When this check box is cleared, the options below it are not available.

Option	Description
Text to capture is displayed before occurrence / **Text to capture is displayed after occurrence**	Specifies the exact occurrence of the value specified in the **Constant** or **Parameter** box, if the value is displayed more than once in the object or area. If you accept the default text that QuickTest recommends, the number in this box is correct. For example, if the selected output string is displayed before the first occurrence of the string First (as shown in the dialog box above). When **Text After** is selected, the number 1 is displayed in the **Text to capture is displayed before occurrence** box. If you modify the recommended value, you must confirm that the occurrence number is accurate. If you choose text that is not unique in the defined object or area, change the occurrence number appropriately. For example, if you want to output the text displayed after the third occurrence of the string Mercury Tours, select **Text Before** and enter 3 in the **Text to capture is displayed after occurrence** box. **Note:** QuickTest starts counting occurrences of the specified **Text After** value from the beginning of the text string you selected to output, and includes any occurrences within the output value string itself.

Option	Description
Constant	Sets the **Text Before** or **Text After** value as a constant. A constant is a value that is defined directly within the test. It remains set for the duration of the test. When you are creating a text output value with **Text Before** selected, the **Constant** box displays the captured **Text Before** value. When you are creating a text output value with **Text After** selected, the **Constant** box displays the captured **Text After** value. You can change the value by typing in the text box. When you are creating a text area output value, the **Text Before** and **Text After** values are not captured. You can enter the text by typing or copying it into the **Constant** box. **Tip:** It is recommended to specify a text string that is unique within the object or area whenever possible, to ensure that the occurrence number is 1.
Parameter	Sets the **Text Before** or **Text After** value as a parameter. For more information on specifying parameter values, see "Configuring a Parameter Value" on page 758.

Specifying the Location for the Output Value Step

If the **Insert statement** area is displayed at the bottom of the dialog box, you can specify where the new output value step should be inserted in your test. For more information, see "Selecting the Location for the Output Value Step" on page 687.

Outputting Table Values

You can output the values of table cells and table properties while recording or editing your test. You specify the values to output using the Table Output Value Properties dialog box.

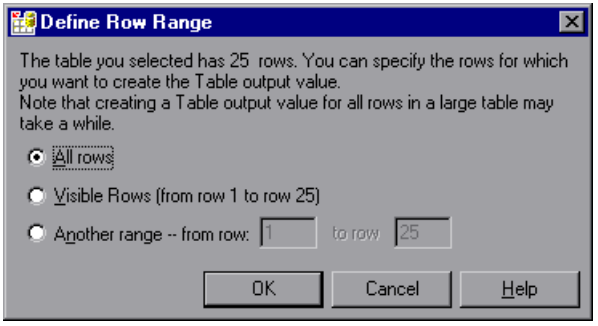
To output table values while recording:

1 Select **Insert** > **Output Value** > **Standard Output Value** or select **Standard Output Value** from the **Insert Checkpoint or Output Value** button. The QuickTest window is hidden, and the pointer changes into a pointing hand. For more information on using the pointing hand feature, see "Tips for Using the Pointing Hand While Recording" on page 702.

2 Click the table from which you want to output cell values. The Object Selection - Output Value Properties dialog box opens.

3 Select a table item from the displayed object tree and click **OK**. If the Table Output Value Properties dialog box opens, skip to step 4.

Otherwise, for certain objects in certain environments, such as WinList View objects, the Define Row Range dialog box opens.



Select the range of rows you want to include in your output value. You can include:

- **All rows.** Includes all of the rows in the table. Note that capturing all of the data for large table or list view objects may take some time.
- **Visible Rows (from row X to row Y).** Includes only the rows visible on the screen. Note that this option may not be available for some environments or object types.
- **Another range -- from row _ to row _.** You can specify any row range in the table.

Click **OK**. The Define Row Range dialog box closes, and the Table Output Value Properties dialog box displays the rows you specified (above the grid area).

4 In the Table Output Value Properties dialog box, specify the settings for the output value. For information on specifying the table content to output, see "Outputting Table Content" on page 703. For information on specifying the object properties to output, see "Outputting Table Properties" on page 709.

Note: For some environments, the Table Output Value Properties dialog box contains two tabs: Table Content and Properties. For other environments, the Table Output Value Properties dialog box displays only the options available in the Table Content tab, but does not contain any tabs.

---

5 Click **OK** to close the dialog box. An output value statement is added for the selected object in the Keyword View and Expert View.

To add a table output value while editing:

1 Depending on whether the object from which you want to output a value is already in a step, do one of the following:

- If you have already recorded a step on the object from which you want to output values, right-click the step and select **Insert Output Value**. Alternatively, select the step and select **Insert** > **Output Value** > **Standard Output Value**.

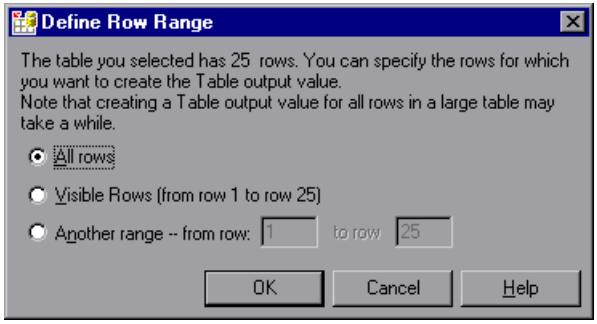
- If you have not recorded a step on the object from which you want to output values, make sure the **Active Screen** button is selected and the Active Screen is visible. Click a step in your test where you want to add an output value. The Active Screen displays the application screen corresponding to the highlighted step. Right-click the table in the Active Screen and select **Insert Output Value**. The Object Selection - Output Value Properties dialog box opens. Select a table item from the displayed object tree and click **OK**.

Note: In some environments, you must have the table open in your application to output a value from it.

---

2 If the Table Output Value Properties dialog box opens, skip to step 3.

Otherwise, for certain objects in certain environments, the Define Row Range dialog box opens.



Select the range of rows you want to include in your output value. You can include:

- **All rows.** Includes all of the rows in the table. Note that capturing all of the data for large table or list view objects may take some time.
- **Visible Rows (from row X to row Y).** Includes only the rows visible on the screen. Note that this option may not be available for some environments or object types.
- **Another range -- from row X to row Y.** You can specify any row range between 1 and the number of rows listed in the table.

Click **OK**. The Define Row Range dialog box closes, and the Table Output Value Properties dialog box displays the rows you specified (above the grid area).

3 In the Table Output Value Properties dialog box, specify the settings for the output value. For information on specifying the table content to output, see "Outputting Table Content" on page 703. For information on specifying the object properties to output, see "Outputting Table Properties" on page 709.

Note: For some environments, the Table Output Value Properties dialog box contains two tabs: Table Content and Properties. For other environments, the Table Output Value Properties dialog box displays only the options available in the Table Content tab, but does not contain any tabs.

---

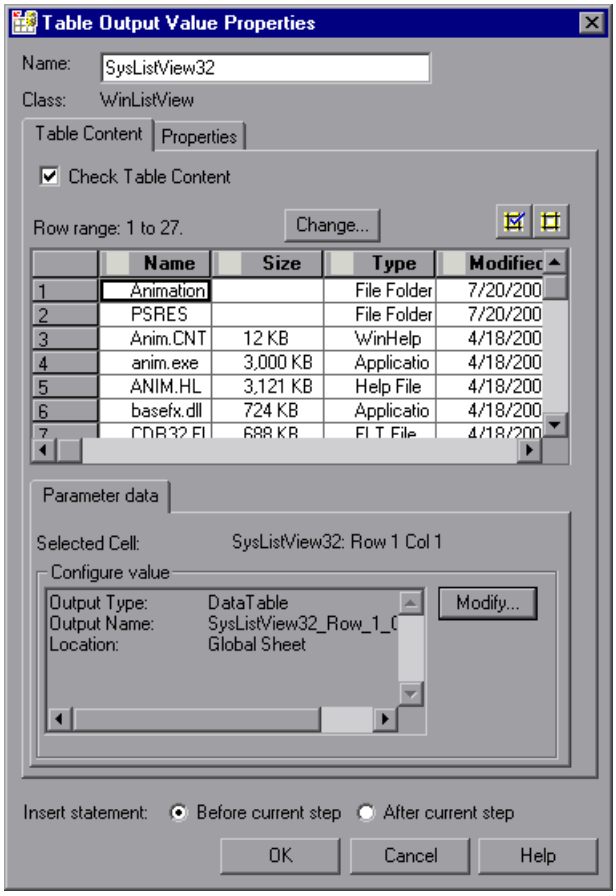
4 Click **OK** to close the dialog box. An output value statement is added for the selected object.

Tips for Using the Pointing Hand While Recording

- 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.

Outputting Table Content

You can specify the table cells whose content you want to output. Depending on the environment, you do this either in the Table Content tab of the Table Output Value Properties dialog box—or directly in the Table Output Value Properties dialog box, if the dialog box does not contain any tabs.



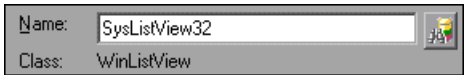
Notes:

- Some of the options shown in this example are available only in certain environments and only for certain objects.
- For some environments, you can also specify object properties (using the Properties tab).

This section describes the general settings and options displayed in the Table Output Value Properties dialog box. Most of the options described in this section are available regardless of whether the Table Output Value Properties dialog box contains tabs.

Identifying the Output Value

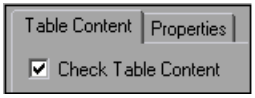
The top part of the Table Output Value Properties dialog box contains the following options:

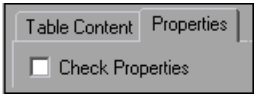


Name	The name that QuickTest assigns to the output value. By default, the output value name is the name of the test object for which you are performing the output value step. You can specify a different name for the output value or accept the default name. If you rename the output value, make sure that the name: ➤ is unique ➤ does not begin or end with a space ➤ does not contain " (double quotation mark) ➤ does not contain the following character combinations: := @@
Class	Specifies the type of object (read-only). This may be a table-type object or a list view-type object.
Find in Repository button	Displays the output value in its repository. **Note:** This option is not available when creating a new output value. It is available only when editing an existing output value.

Tabs (If Available)

If the Table Output Value Properties dialog box contains tabs, each tab displays a check box. You can select one or both of these check boxes to specify the type of data to output.



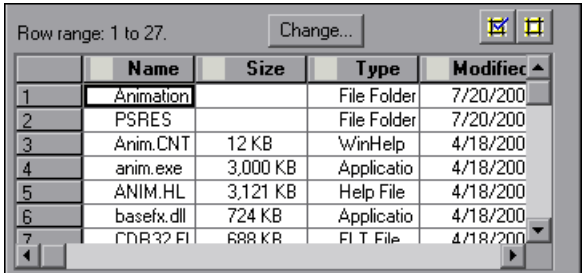


Check Table Content check box	(Table Content tab) Selecting the **Check Table Content** check box instructs QuickTest to output the values of the selected cells in the table object. (Selected by default)
Check Properties check box	(Properties tab) Selecting the **Check Properties** check box instructs QuickTest to output the property values of the selected cells in the table object. (Cleared by default)

Note: These check boxes are displayed only if the Table Output Value Properties dialog box contains tabs. If the Table Output Value Properties dialog box does not contain tabs, QuickTest automatically outputs the values of the selected cells as defined in the dialog box.

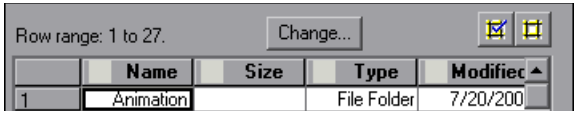
Choosing Cells for Output Values

The top part of the dialog box displays a grid representing the cells in the captured table. The column header names are captured from the table you selected for your output value step. You can output the values for one or more cells in the grid.



Tip: You can change the column widths and row heights of the grid by dragging the column and row header dividers.

Some environments and objects support selecting a row range. This enables you to specify which rows are displayed in the grid area. If row range selection is supported, the row range you specify when creating the output value is displayed above the grid:



Clicking the **Change** button enables you to modify the row range. (Depending on the environment, your application may need to be open and the relevant table displayed to modify the row range.) For more information, see “Modifying a Table Output Value” on page 711.

To choose a cell for a value to output:

Double-click the cell or select it and click the **Add Output Value** button (located above the grid, on the right). An output value icon is added to the cell.

To remove a cell from an output value:

Double-click the cell again or select it and click the **Remove Output Value** button (located above the grid, on the right). The output value icon is removed from the cell.

Specifying the Settings for the Output Value

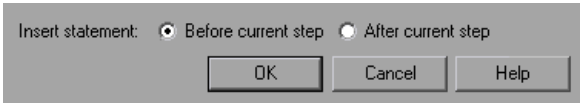
When a value in a table cell is first selected for output, the default output definition for the value is displayed in the **Configure value** area. For more information on default output definitions, see "Understanding Default Output Definitions" on page 683.

When you select a value in a table cell, you can:

- accept the displayed output definition by selecting another cell or by clicking **OK**.
- change the output type and/or settings for the selected value by clicking the **Modify** button. The Output Options dialog box opens and displays the current output type and settings for the value. For more information, see "Specifying the Output Type and Settings" on page 683.

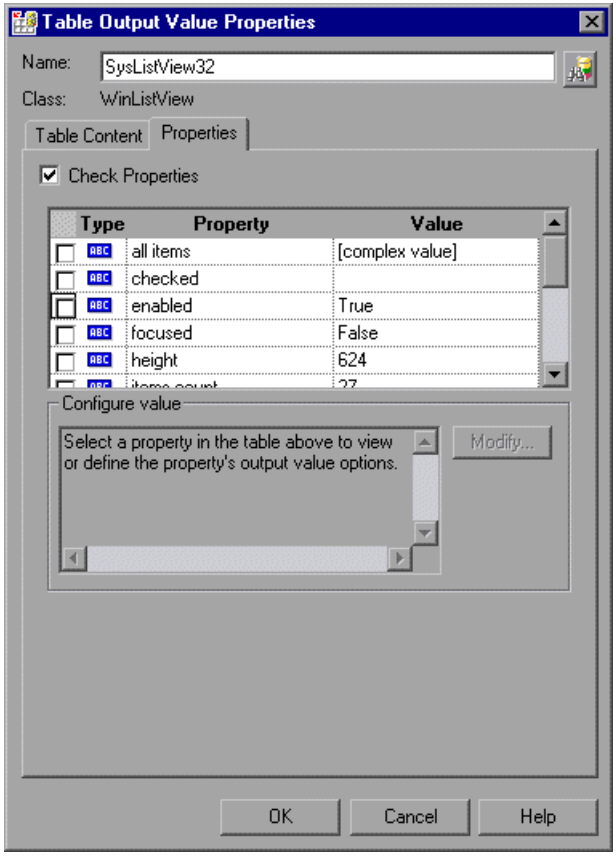
Specifying the Location for the Output Value Step

If the **Insert statement** area is displayed at the bottom of the dialog box, you can specify where the new output value step should be inserted in your test. For more information, see "Selecting the Location for the Output Value Step" on page 687.



Outputting Table Properties

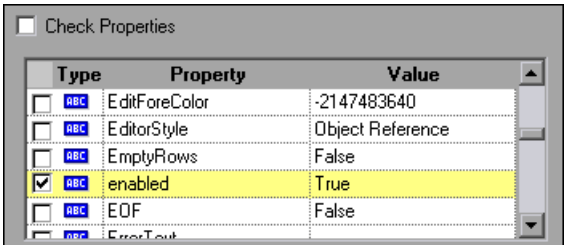
For certain environments, you can specify which object property values you want to output. By default, when you create a table output value on an object, QuickTest captures all the object's properties, but does not select any properties to output.



Note: For information on general table output value options, such as **Name** and **Class**, and on the options available in the Table Content tab, see "Outputting Table Content" on page 703.

Selecting Properties to Output

When you create a table output value, the Properties pane displays the object's default properties, including the properties, their values, and their types.



You instruct QuickTest to output specific properties by selecting the **Check Properties** check box. (This check box is cleared by default.)

The Properties pane for the object contains the following:

Check box	To output a property, select the corresponding check box. To remove a property from the output, clear the corresponding check box.
Type	The icon indicates that the value of the property is currently a constant. The icon indicates that the value of the property is currently a test or action parameter. The icon indicates that the value of the property is currently a Data Table parameter. The icon indicates that the value of the property is currently an environment variable parameter. The icon indicates that the value of the property is currently a random number parameter.
Property	The name of the property.
Value	The expected value of the property. For more information on modifying the value of a property, see "Setting Values in the Configure Value Area" on page 757.

Modifying a Table Output Value

You can modify the table output value options, which specify where an output value is stored and how it can be used during the run session. You can also modify the number of rows for which QuickTest can output the values of specific table cells.

To modify the table output value options:

1 In the Keyword View or Expert View, right-click the Output CheckPoint step for the table whose output options you want to modify and select **Output Value Properties**. Alternatively, select the step containing the Output CheckPoint and select **Edit > Step Properties > Output Value Properties**. The Table Output Value Properties dialog box opens.

2 Perform one of the following:

- If the Table Output Value Properties dialog box does not contain tabs, click the **Modify** button. The Output Options dialog box opens.
- If the Table Output Value Properties dialog box contains tabs:
  - To modify the output options for the table content, make sure the Table Content tab is displayed and click the **Modify** button.
  - To modify the output options for the object properties, select the Properties tab and click the **Modify** button.

  The Output Options dialog box opens.

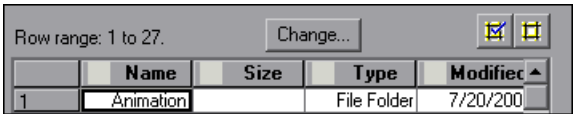
3 Modify the output value, as needed. For more information, see "Specifying the Output Type and Settings" on page 683.

4 You can also rename the output value, if needed. For more information, see "Identifying the Output Value" on page 680.

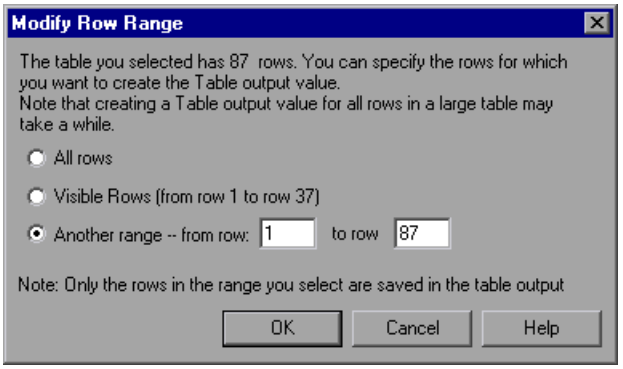
To modify the range or number of rows in an existing table output value:

1 Open the application containing the table or list view object from which you want to output a value and display the object in the application.

2 In the Keyword View or Expert View, right-click the Output CheckPoint step for the table whose row range you want to modify and select **Output Value Properties**. Alternatively, select the step containing the Output CheckPoint and select **Edit** > **Step Properties** > **Output Value Properties**. The Table Output Value Properties dialog box opens, displaying the currently selected row range.



3 In the Table Content tab, click the **Change** button at the top of the dialog box (above the grid area). The Modify Row Range dialog box opens.



4 Select the range of rows you want to include in your output value step. You can include all the rows, only the visible rows, or another range that you specify.

Note: The **Visible Rows** option may not be available for some environments or object types.

---

5 Click **OK**. The Modify Row Range dialog box closes, and the Table Output Value Properties dialog box displays the rows you specified in the Modify Row Range dialog box.

- If your modified row range includes new rows, you can select the cells from which you want to output values from the newly selected rows. The cells whose values you select will be outputted during the run session.
- If your modified row range includes some or all of the rows that you specified previously, the cells whose values you selected to output will be outputted during the run session.
- If your modified row range excludes some or all of the rows that were selected previously, the values of any previously selected cells in those rows will not be outputted during the run session.

Note: You can output values only from cells that are included in the specified row range.

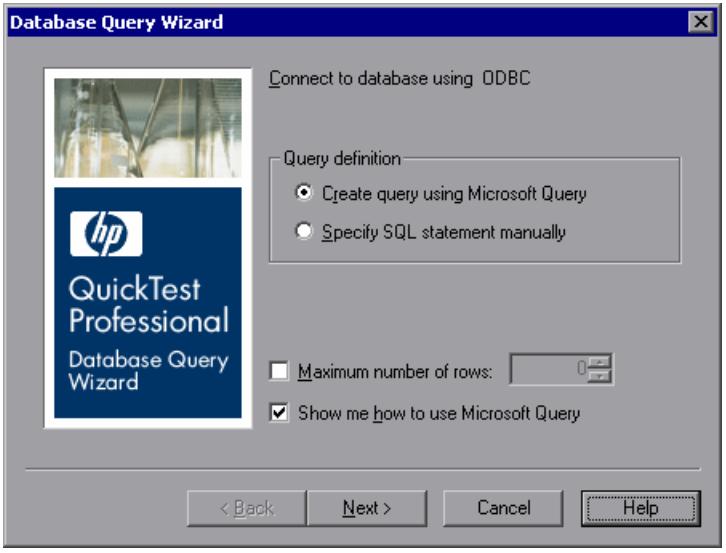
Outputting Database Values

You can create database output values by defining a query to retrieve data from the database and selecting the values you want to output from the query result set. You can then specify the output settings for the selected values. During the run session, QuickTest captures the current data from the database and outputs the values according to the specified settings.

You can create database output values while recording or editing your test.

To create database output values:

1 Select **Insert** > **Output Value** > **Database Output Value**. The Database Query Wizard opens.



2 Use the wizard to define a query to retrieve the data that you want to output. Follow the instructions for creating a database checkpoint in "Creating a Check on a Database" on page 576.

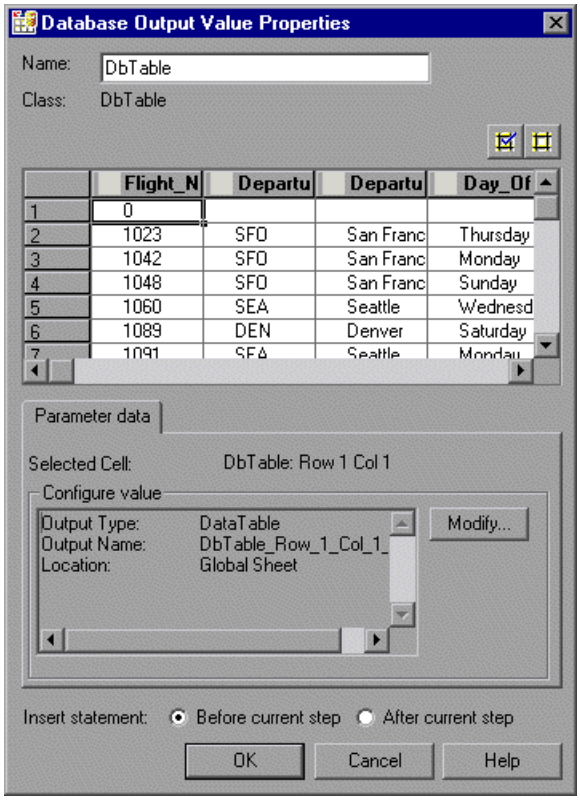
After you finish defining your query, the Database Output Value Properties dialog box opens.

3 Specify the values to output and their settings. For more information, see "Defining Database Output Values" on page 715.

4 When you finish defining the output value details, click **OK**. QuickTest inserts an output value step in your test.

Defining Database Output Values

The Database Output Value Properties dialog box enables you to select the database cells for the values you want to output. You can define the output settings for each value that you select.



Identifying the Database Output Value

The top part of the Database Output Value Properties dialog box contains the following options:

Name	The name that QuickTest assigns to the output value. By default, the output value name is the name of the test object for which you are performing the output value step. You can specify a different name for the output value or accept the default name. If you rename the output value, make sure that the name: ➤ is unique ➤ does not begin or end with a space ➤ does not contain " (double quotation mark) ➤ does not contain the following character combinations: := @@
Class	Specifies the type of test object (read-only) for which you are performing the output value step.
Find in Repository button 	Displays the output value in its repository. **Note:** This option is not available when creating a new output value. It is available only when editing an existing output value.

Choosing Cells for Output Values

The top part of the dialog box displays a grid representing the cells in the captured database query results set. You can output the values for one or more cells in the grid.

Tip: You can change the width of the columns and the height of the rows in the grid by dragging the boundaries of the column and row headers.

To choose a cell for a value to output:

Double-click the cell or select it and click the **Add Output Value** button (located above the grid, on the right). An output value icon is added to the cell.

To remove a cell from an output value:

Double-click the cell again or select it and click the **Remove Output Value** button (located above the grid, on the right). The output value icon is removed from the cell.

Specifying the Settings for the Output Value

When a value in a database cell is first selected for output, the default output definition for the value is displayed in the **Configure value** area. For more information on default output definitions, see "Understanding Default Output Definitions" on page 683.

When you select a value in a database cell, you can:

- accept the displayed output definition by selecting another cell or by clicking **OK**.
- change the output type and/or settings for the selected value by clicking the **Modify** button. The Output Options dialog box opens and displays the current output type and settings for the value. For more information, see "Specifying the Output Type and Settings" on page 683.

Specifying the Location for the Output Value Step

If the **Insert statement** area is displayed at the bottom of the dialog box, you can specify where the new output value step should be inserted in your test. For more information, see "Selecting the Location for the Output Value Step" on page 687.

Outputting XML Values

You can create XML output value steps from any XML document contained in an XML Web page or frame, directly from an XML file, or from test objects that support XML. You can output element and/or attribute values in an XML output value step.

You can insert XML Web page or frame output value steps only while you are recording. You can create XML output value steps from an XML file or from a test object while recording or editing your test.

Note: XML Output Values are compatible with namespace standards and a change in namespace between nodes stored in the Output Properties dialog box XML tree and the actual values will result in a failed output value step.

For more information on XML standards, see http://www.w3.org/XML/

For more information on namespace standards, see http://www.w3.org/TR/1999/REC-xml-names-19990114/

To create XML output values from an XML Web page or frame:

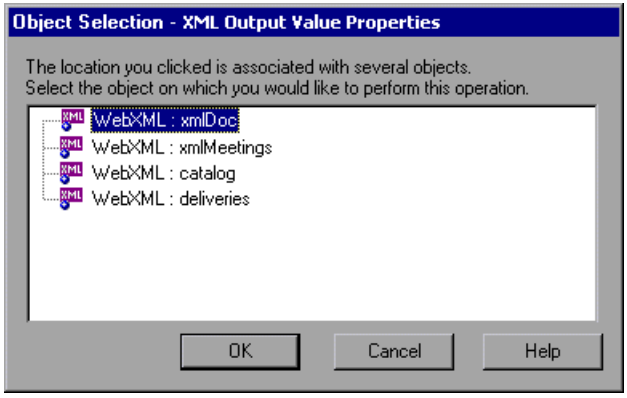
1 While recording, select **Insert** > **Output Value** > **XML Output Value (From Application)**, or click the **Insert Checkpoint or Output Value** button and select **XML Output Value (From Application)**. The pointer changes into a pointing hand. For more information on using the pointing hand, see "Tips for Using the Pointing Hand" on page 726.

Note: The **XML Output Value (From Application)** option is available only when the Web Add-in is installed and loaded. For more information on loading add-ins, see the section on working with QuickTest add-ins in the *HP QuickTest Professional Add-ins Guide*.

You can also insert a Web page or frame output value step using the **XML (From Resource)** option by selecting an existing WebXML test object. For more information, see creating XML output values from a test object that supports XML on page 723.

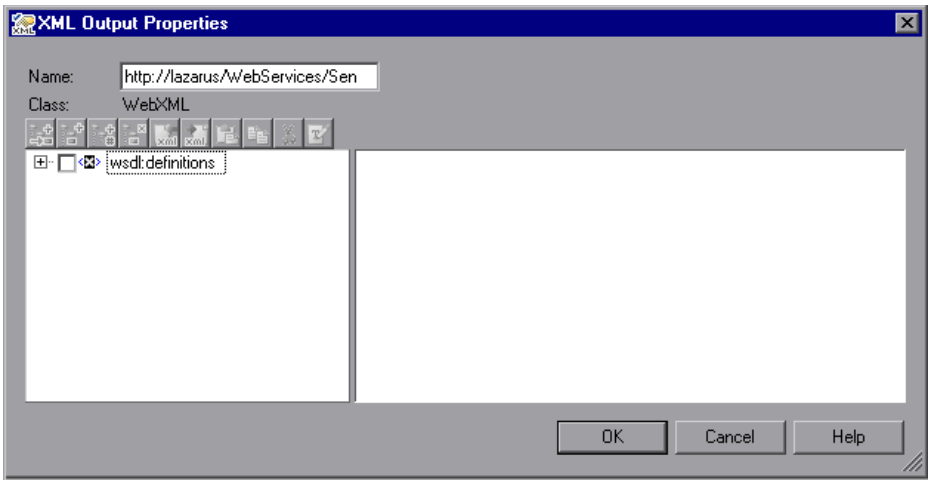
---

2 Click the XML object for which you want to specify an output value. If the location you clicked is associated with more than one object, the Object Selection - XML Output Value Properties dialog box opens.



3 Select the XML item you want to specify for the output value step.

4 Click **OK**. The XML Output Properties dialog box opens.



The XML Output Properties dialog box displays the element hierarchy and values (character data) of the selected XML document.

In the **Name** box, either accept the name that QuickTest assigns to the output value step or specify another name for it. By default, the output value name is the name of the test object on which the output value step is being performed.

If you rename it, make sure that the name:

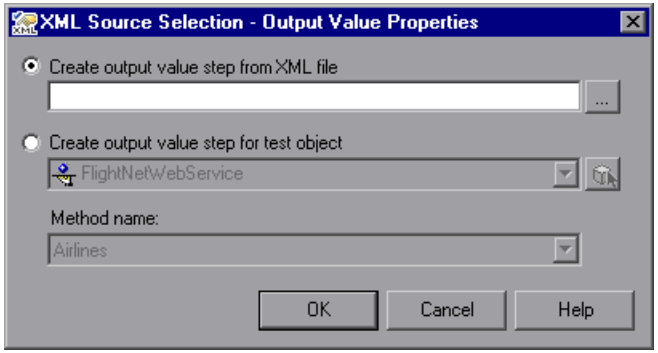
- is unique
- does not begin or end with a space
- does not contain " (double quotation mark)
- does not contain the following character combinations:
  :=
  @@

5 Select the items to output. For more information, see "Understanding the XML Output Properties Dialog Box" on page 727.

6 When you finish defining the output value details, click **OK**. QuickTest inserts an output value step in your test.

To create an XML output value step from an XML file:

1 Select **Insert** > **Output Value** > **XML Output Value (From Resource)**, or click the **Insert Checkpoint or Output Value** button and select **XML Output Value (From Resource)**. The XML Source Selection - Output Value Properties dialog box opens.



Tip: You can also insert an XML File output value step by selecting an existing XMLFile test object. For more information, see creating XML output values from a test object that supports XML on page 723.

---

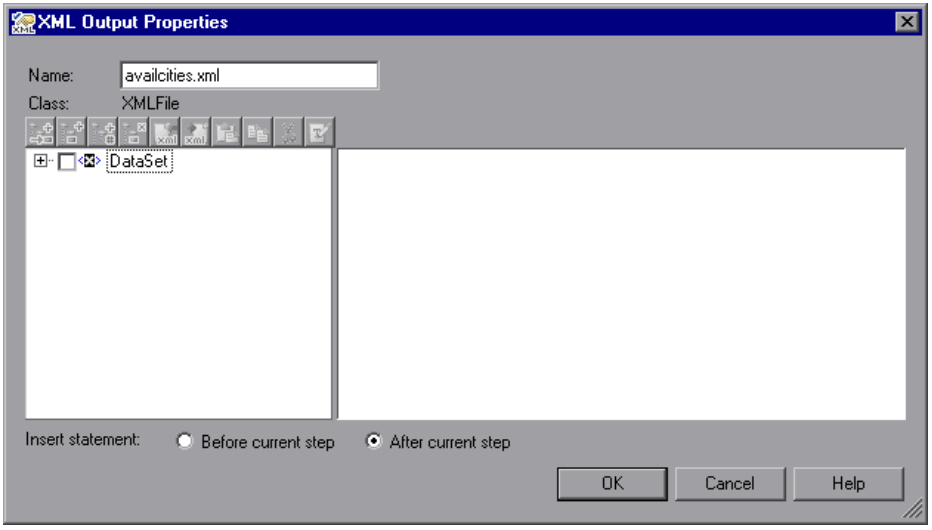
2 Select **Create output value step from XML file**. Enter the Internet address or file path of the XML file.

Alternatively, click the browse button to open the Open XML File dialog box. In the sidebar, select the location of the XML file, and then navigate to the XML file for which you want to create an output value. You can specify an XML file either from your file system or from Quality Center. Select the file and click **Open**. The file path and name are entered in the box.

Note: You can enter a relative path and QuickTest will search for the XML file in the folders listed in the Folders pane of the Options dialog box. Once QuickTest locates the file, it saves it as an absolute path and uses the absolute path during the test run. For more information, see "Setting Folder Testing Options" on page 1237.

---

3 Click **OK**. The XML Output Properties dialog box opens.



The XML Output Properties dialog box displays the element hierarchy and values (character data) of the selected XML document.

In the **Name** box, either accept the name that QuickTest assigns to the output value step or specify another name for it. By default, the output value name is the name of the test object on which the output value step is being performed.

If you rename it, make sure that the name:

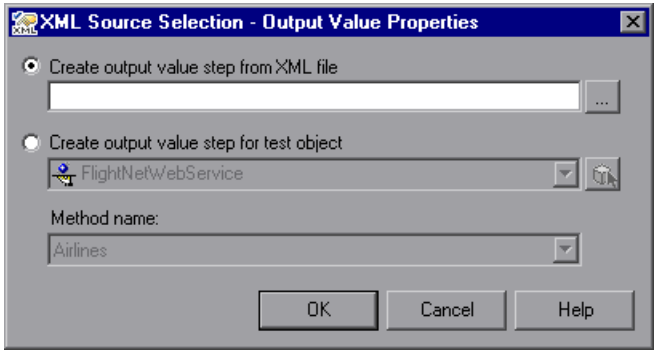
- is unique
- does not begin or end with a space
- does not contain " (double quotation mark)
- does not contain the following character combinations:
  :=
  @@

4 Select the items to output and the location for the output value step. For more information, see "Understanding the XML Output Properties Dialog Box" on page 727.

5 When you finish defining the output value details, click **OK**. QuickTest inserts an output value step in your test.

To create an XML output value step from a test object that supports XML:

1 Select **Insert** > **Output Value** > **XML Output Value (From Resource)**, or click the **Insert Checkpoint or Output Value** button and select **XML Output Value (From Resource)**. The XML Source Selection - Output Value Properties dialog box opens.



2 Select **Create output value step for test object** and select the test object from which you want to output values.

To select an object that is not displayed in the list, click **Object from Repository**. Then select an XML test object from the object repository on which to create a new output value step. The selected object must support XML.

You can select an existing WebXML or XMLFile test object type or you can select a WebService test object.

Note: Selecting a WebXML or XMLFile test object is identical to using the **XML Output Value (From Application)** or **XML Output Value (From Resource)** options, but may be faster than browsing to these objects and can be inserted while recording or editing. However, to use this option, the XML source must be available when you select the test object (the Web page must be open or the file must exist in the same location as when the test object was defined).

---

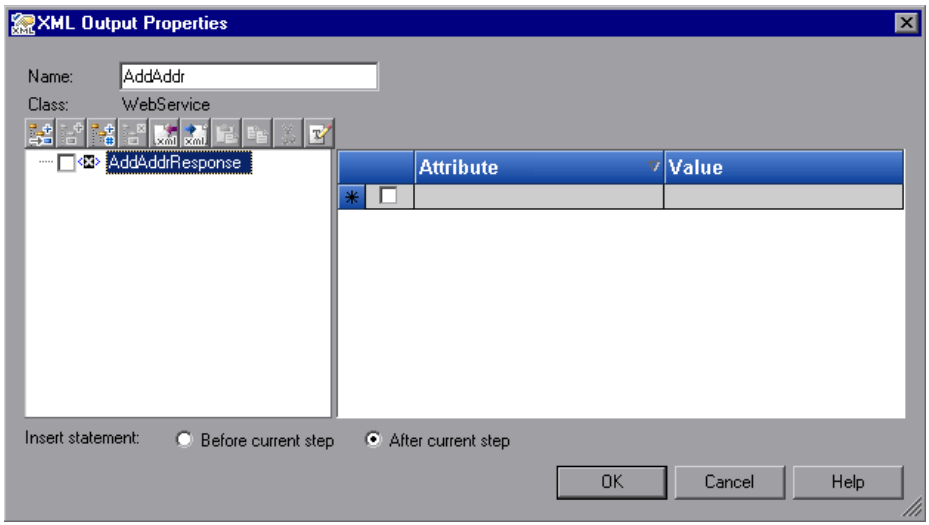
3 If you select a WebService test object, then the **Method name** box is enabled. Select the Web service operation whose return values you want to check.

Notes:

- The **Method name** box is available only if the Web Services Add-in is installed and loaded. The **Method name** box is enabled only if you select a WebService test object.
- XML output value steps on Web service operations retrieve the values returned from the last Web service operation performed on the test object. If a different Web service operation step is performed prior to the output value step, then the output value step will fail.

---

4 Click **OK**. The XML Output Properties dialog box opens.



The XML Output Properties dialog box displays the element hierarchy in an XML tree, and the attributes and values (if any) of the selected XML output.

When you create an XML output value for an operation return value, only a generic XML tree is created and shown in the XML Output Properties dialog box. Before you can select which element or attribute values you want to output, you must populate the XML tree with the actual elements, attributes, and values. For more information, see "Updating the XML Hierarchy for XML Test Object Operation Output Value Steps (For WebService Test Objects Only)" on page 732.

5 In the **Name** box, either accept the name that QuickTest assigns to the output value step or specify another name for it. By default, the output value name is the name of the test object on which the output value step is being performed.

If you rename it, make sure that the name:

- is unique
- does not begin or end with a space
- does not contain " (double quotation mark)
- does not contain the following character combinations:
  :=
  @@

Select the items to output and the location for the output value step. For more information, see "Understanding the XML Output Properties Dialog Box" on page 727.

6 When you finish defining the output value details, click **OK**. QuickTest inserts an output value step in your test.

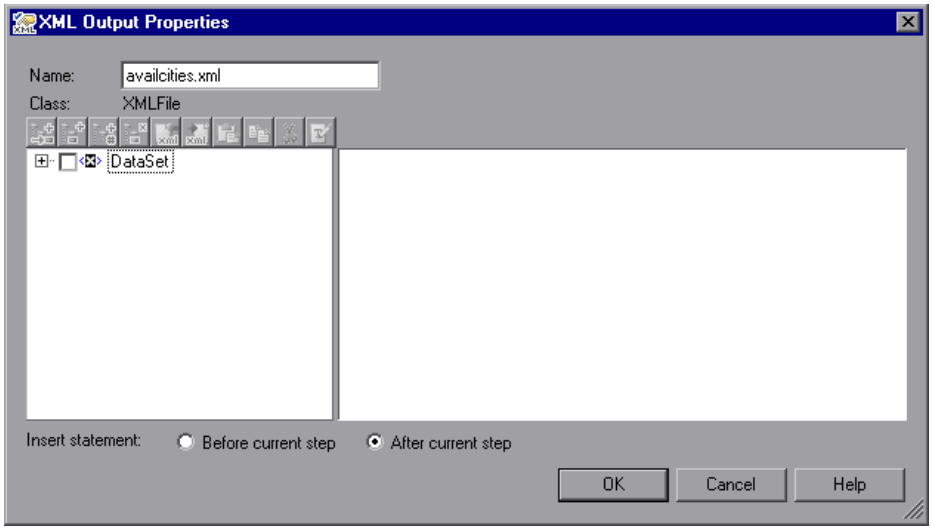
Tips for Using the Pointing Hand

- 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.

Understanding the XML Output Properties Dialog Box

The XML Output Properties dialog box enables you to choose which element and/or attribute values to output and to define the output settings for each value that you select.



Identifying the Object

The top part of the XML Output Properties dialog box displays information on the test object for which you are creating an output value step:

Item	Description
Name	The name that QuickTest assigns to the output value step. By default, the output value name is the name of the test object for which you are performing the output value step. You can specify a different name for the output value or accept the default name. If you rename it, make sure that the name: ➤ is unique ➤ does not begin or end with a space ➤ does not contain " (double quotation mark) ➤ does not contain the following character combinations: := @@
Class	The test object class on which you are creating the output value step. This can be: **XMLFile** (for files), **WebXML** (for Web pages or frames) or **WebService** (for a Web service).
Find in Repository button	Displays the output value in its repository. **Note:** This option is not available when creating a new output value. It is available only when editing an existing output value.

Modifying the XML Tree

The following commands are available according to the node you select in the tree:

Command	Icon	Description
Add Child		Adds a child node below the selected node in the tree.
Insert Sibling		Adds a sibling node at the same level as the selected node in the tree.
Add Value		Enables you to assign a constant or parameterized value to the selected element.
Delete		Deletes the selected node. Note that you cannot delete the root node of the output value step.
Import XML		Enables you to browse to and select a file structure from an existing XML file. The new file overrides the selected node's current sub-tree.
Export XML		Enables you to save the file structure of the selected node to an XML file.
Paste		Pastes a cut or copied node as a child node below the selected node in the XML tree. **Note:** You cannot paste an XML element node as its own descendant.
Copy		Makes a copy of the selected node, which you can then paste in another location in the XML tree.
Cut		Prepares the selected node to be cut and copies it to the clipboard. When you paste the node in the new location, it is removed from the original location in the XML tree.
Edit XML as Text		Opens the Edit XML as Text dialog box, enabling you to modify the XML text of the selected node and it's subnodes in a test editor. For more information, see "Understanding the Edit XML as Text Dialog Box" on page 613.

Command	Icon	Description
Duplicate		Adds a new node, identical to the selected one, as a sibling node at the same level as the selected node in the XML tree. **Note:** This command is available only from the context menu (right-click menu).

XML Tree

The XML tree displays the hierarchical relationship between each element and value in the XML tree, enabling you to select the element and/or attribute values that you want to output. Each element node is displayed with a  icon. Each value node is displayed with a  icon.

Note: When you create an XML output value for an operation return value, only a generic XML tree is created and shown in the XML Output Properties dialog box. Before you can select which element or attribute values you want to output, you must populate the XML tree with the actual elements, attributes, and values. For more information, see "Updating the XML Hierarchy for XML Test Object Operation Output Value Steps (For WebService Test Objects Only)" on page 732.

Select an element node in the XML tree to display or set output options for its attributes and values on the right of the XML Output Properties dialog box. Select a value node in the XML tree to display or set output options for its value on the right of the XML Output Properties dialog box.

Tip: The XML tree pane and the **Attribute** and **Value** columns in the right pane are resizable.

To set output XML options:

1 Select the check box for an element or value node in the XML tree to indicate that you want to output a value for that node.

2 Select the element or value node to display or set output options for its attributes and/or values.

3 If you are outputting an element attribute, select the check box of the attributes for which you want output values.

4 Click in the **Value** column of an attribute, or click in the cell of an element value, and then click the **Output Options** button  to display the Value Configuration Options dialog box, which enables you to select or define the parameter in which you want to store the retrieved value.

5 In the Value Configuration Options dialog box, select the parameter type. Additional options are available for the output parameter type that you select. For more information on the options available for each parameter type, see:

- **Data Table.** "Using Data Table Parameters" on page 639.
- **Environment.** "Using Environment Variable Parameters" on page 645.
- **Random Number.** "Using Random Number Parameters" on page 655.

Insert Statement Options

If you are inserting an output value step while editing your test, the bottom part of the XML Output Properties dialog box displays **Insert statement** options, enabling you to choose whether you want to insert the output value step before or after the step that you selected. Select **Before current step** if you want to insert the step before the highlighted step is performed. Select **After current step** if you want to insert the step after the highlighted step is performed.

Note: The **Insert statement** options are not available if you are adding a new output value step while recording or if you are modifying an existing output value step. They are available only if you are adding a new output value step while editing steps.

Updating the XML Hierarchy for XML Test Object Operation Output Value Steps (For WebService Test Objects Only)

This section is relevant only when working with XML output value steps on WebService test object operations (with the QuickTest Professional Web Services Add-in).

When you create an XML output value step for a test object operation (for a WebService test object), the XML tree of the operation return value data cannot be generated. Therefore, only a generic XML tree is created. To select the elements and attributes to output, you must first populate the XML tree with the actual elements, attributes, and values that the operation is expected to return.

You can use one of the three methods below to populate the XML tree:

- Updating the XML Tree Manually
- Importing an XML Tree from a File
- Updating the XML Tree Using Update Run Mode

Updating the XML Tree Manually

You can update the XML tree by adding elements, attributes, and values manually in the XML Output Properties dialog box.

To update the XML tree manually:

1 In the Keyword View, select the output value step whose XML tree you want to update. Click in the **Value** cell.

2 Click the **Output Properties** button or right-click and select **Output Value Properties**. The XML Output Properties dialog box opens.

3 Select a node in the XML tree and then click a toolbar button or choose an option from the context (right-click) menu to:

- Add an element at the same level as the selected node

- Add an element below the selected node

- Add a value to the selected node

- Edit the XML text of the selected node

➤ Copy the selected node

➤ Cut the selected node (the selected node is removed only after you paste it in another location)

➤ Paste a cut or copied node as a child node below the selected node

➤ Delete the selected node

➤ Duplicate the selected node, adding an identical node as a sibling node at the same level (this command is available only from the right-click menu)

For more information on the available tools in the XML Output Properties dialog box, see “Understanding the XML Output Properties Dialog Box” on page 727.

Importing an XML Tree from a File

You can import an XML tree from an existing file for a specific element in the XML tree hierarchy or for the whole tree.

To import an existing XML tree from a file:

1 In the Keyword View, select the output value step whose XML tree you want to update.

2 Click in the **Value** cell and then click the **Output Properties** button. The XML Output Value Properties dialog box opens.

3 If you want to import an XML hierarchy for the whole XML tree, select the root node. If you want to import an XML hierarchy for a specific element, select the element in the XML tree hierarchy.

4 Click the **Import XML** button. A message warns you that the imported hierarchy overwrites the selected node and its sub-tree. Click **Yes** to close the message.

5 In the Import XML from File dialog box, browse to the required XML file and click **Open**. The XML hierarchy is imported from the file.

Updating the XML Tree Using Update Run Mode

QuickTest cannot generate the return values of an operation when you insert an XML output value step on a Web service operation, but it can generate this information after it runs the operation. Therefore, you can run your Web service test in Update Run mode to automatically populate or update the elements, attributes and values in your XML tree.

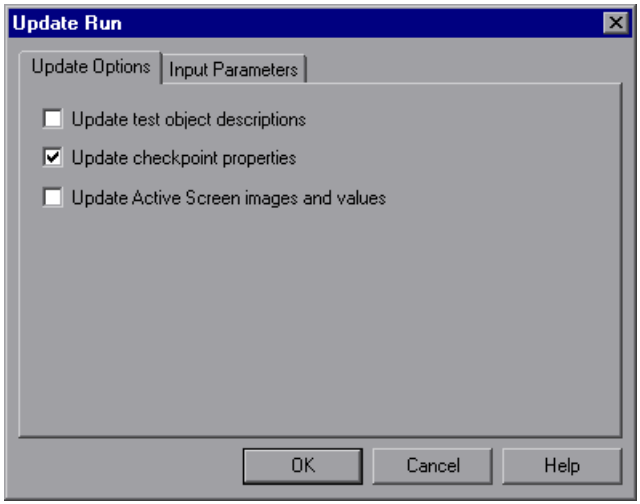
To generate a new XML tree based on the current return values of the Web service operation, ensure that none of the node, attribute, or value check boxes are selected in the XML tree of the Output Value Properties dialog box.

Note: XML Output Value steps on Web service operations retrieve the values returned from the last Web service operation performed on the test object. If a different Web service operation step is performed prior to the output value step, then the output value step will fail.

To update an XML tree using Update Run mode:

1 Open a test containing an XML test object output value step for a Web service operation.

2 Click the down arrow next to the **Run** button in the toolbar and select **Update Run Mode**, or select **Automation** > **Update Run Mode**. The Update Run dialog box opens.



3 Select **Update checkpoint properties** and click **OK**. QuickTest runs the test and updates the XML hierarchy and values for each blank XML checkpoint and XML output value step in the test. It updates values only for XML checkpoints or output value steps that have one or more nodes selected.

4 If you want to confirm that QuickTest successfully updated your output value step, expand the tree in the Test Results window and select the XML output value step. Then check that Update done is displayed in the pane on the right. (If the Test Results window did not open automatically at the end of the run, click the **Results** button or select **Automation** > **Results**.)

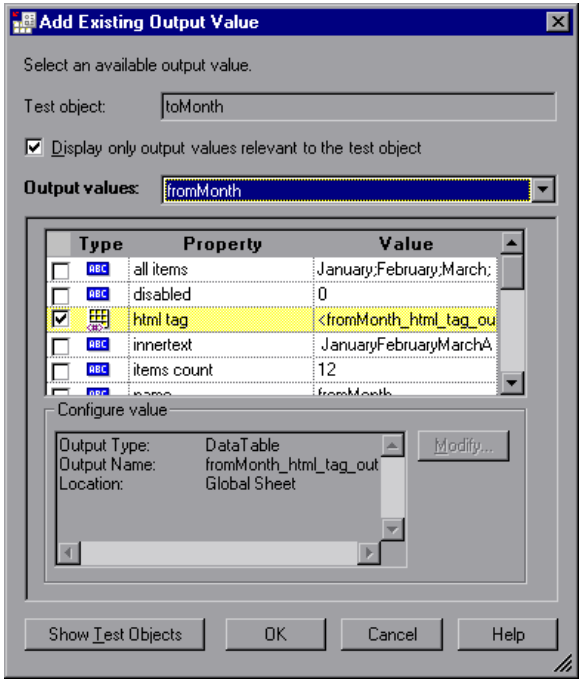
Adding Existing Output Values to a Test

QuickTest enables you to insert existing output values into your test.

When you insert an existing output value in your test, consider which output values should be used in multiple locations in your test. Each time an output value step is performed, the value contained in the output value is overwritten with the new output value. You should insert an existing output value into your test only if the stored value will no longer be needed by your test when the output value object is used again.

Understanding the Add Existing Output Value Dialog Box

You open the Add Existing Output Value dialog box by selecting **Insert** > **Output Value** > **Existing Output Value**. This option is available only if at least one of the object repositories associated with the current action (including the local object repository) contains at least one Output object.



If a step is highlighted in the Keyword View or the cursor is located in a step in the Expert View, the Add Existing Output Value dialog box opens with the **TestObjects** tree hidden.

The test object displayed in the **Test object** box is the object from the highlighted step in the Keyword View or the specific object where the cursor is located in the Expert View.

You can display or hide the **TestObjects** tree by clicking the **Show/Hide Test Objects** button.

The Add Existing Output Value dialog box contains the following options:

Option	Description
Test object	Specifies the test object for which you are adding an output value.
TestObjects tree	Displays the objects stored in the object repositories associated with the current action.
Show/Hide Test Objects	Shows or hides the **TestObjects** tree.
Display only output values relevant to the selected test object	When selected, QuickTest determines which output value objects from the current action's object repositories are relevant for the selected object (based on the output value type and the properties selected to output in the output value object) and displays only those output value objects in the Output Values list. When using this option, it is recommended to open your application and display the selected object so that QuickTest can accurately determine all of the checkpoints that can apply to that object.
Output values	Lists the checkpoints available for insertion. If the **Display only output values relevant to the selected test object** option is cleared, this list includes all output value objects from all object repositories associated with the current action. If the **Display only output values relevant to the selected test object** option is selected, this list displays only the relevant output value objects as described above.
Output Value Properties Area	Displays the output value options for the selected output value object in read-only format.

To insert an existing output value in your test:

1. Select the step after which you want to insert the checkpoint.
2. Select **Insert** > **Output Value** > **Existing Output Value**. The Add Existing Output Value dialog box opens.
3. If the **TestObjects** tree is displayed, select the object for which you want to insert an Output Value. Otherwise proceed to step 4.
4. From the **Output values** list, select the output value that you want to insert for the object displayed in the **Test object** box.
5. Click **OK**. The output value step is inserted after the current step.

26

Working with Text Recognition for Windows-Based Objects

QuickTest uses various mechanisms to identify the text strings in your Windows-based objects. This chapter describes how to configure QuickTest to optimize the results of text recognition for your Window-based objects.

This chapter includes:

About Working with Text Recognition for Windows-Based Objects

QuickTest identifies text in your application using either a Windows API-based mechanism or an OCR (optical character recognition) mechanism. You can use the text and text area checkpoint or output value commands to verify or retrieve text in your Windows-based objects. Alternatively, you can use the *testobject*.**GetText** (for Terminal Emulator objects), *testobject*.**GetVisibleText**, or *testobject*.**GetTextLocation** test object methods, or the **TextUtil.GetText** or **TextUtil.GetTextLocation** reserved object methods to capture the text you need.

By default, QuickTest tries to retrieve the text directly from the object using a Windows API-based mechanism. If QuickTest cannot capture the text this way (for example, because the text is part of a picture), it tries to capture the text using an OCR (optical character recognition) mechanism. You use the Text Recognition pane to specify the preferred text recognition mechanism and OCR-specific settings.

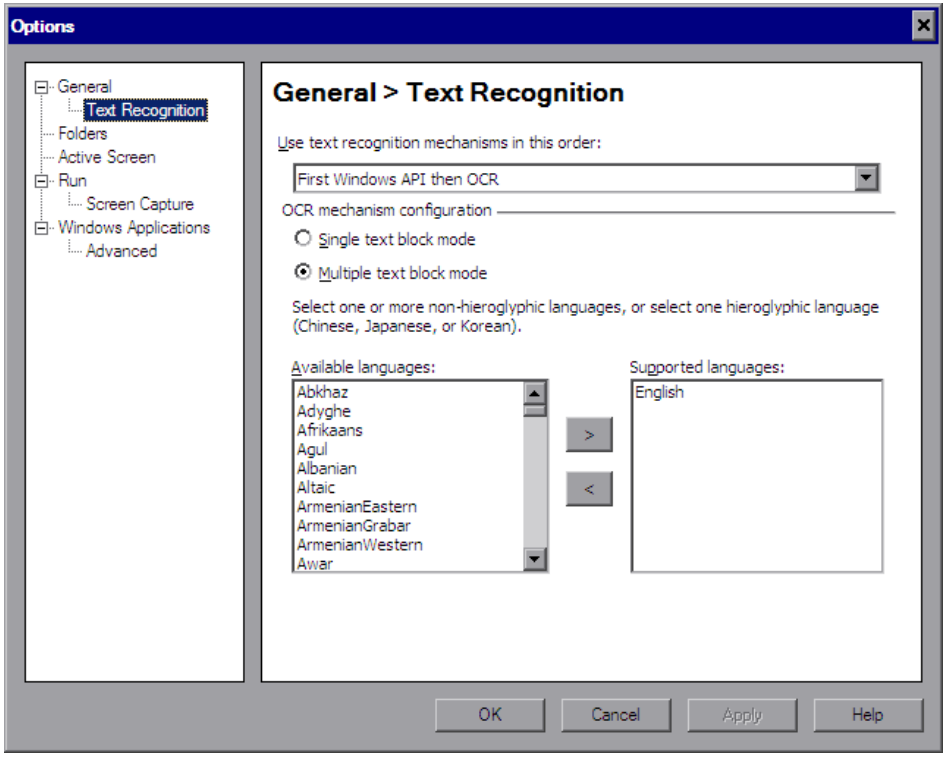
Before you insert a text / text area checkpoint or output value, review the "Guidelines for Text Recognition" on page 746.

The Options Dialog Box: General > Text Recognition Pane

Description	Enables you to configure how QuickTest identifies text in your application. You can use this pane to modify the default text capture mechanism, OCR (optical character recognition) mechanism mode, and the language dictionaries the OCR mechanism uses to identify text.
Accessed by	**Tools** menu > **Options** item > **General** node > **Text Recognition** node

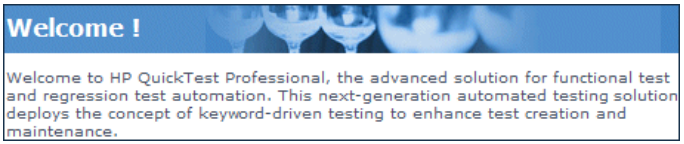
Important Information	The General > Text Recognition options are relevant only for Windows-based objects, such as Standard Windows, .NET WinForms, WPF, SAP Gui for Windows, Visual Basic, and ActiveX.
Learn More	**Conceptual Overview:** "About Working with Text Recognition for Windows-Based Objects" on page 742 **Additional related topics**: "Additional References" on page 746

Below is an image of the General > Text Recognition pane of the Options dialog box:



The General > Text Recognition pane options include:

Option	Description
Use text recognition mechanisms in this order	Specifies the text recognition mechanism that QuickTest uses when capturing text. **Possible values:** **First Windows API then OCR.** (Default) Instructs QuickTest to first try to retrieve text directly from the object using the Windows API-based mechanism. If no text can be retrieved (for example, because the text is part of a picture), QuickTest tries to retrieve text using the OCR (optical character recognition) mechanism. (This setting is highly recommended when working with CJK (Chinese, Japanese, Korean) languages.) **First OCR then Windows API.** Instructs QuickTest to first try to retrieve text from the object using the OCR mechanism. If no text can be retrieved, then QuickTest uses its Windows API-based mechanism to retrieve text from the object. **Use Only Windows API.** Instructs QuickTest to use only the Windows API-based mechanism (and not the OCR mechanism) to retrieve text from the object. **Use Only OCR.** Instructs QuickTest to use only the OCR mechanism (and not the Windows API-based mechanism) to retrieve text from the object. (Required when working with Windows Vista.) For more information on text recognition support in Windows-based environments, see the *HP QuickTest Professional Readme*.
Single text block mode	Select this radio button if the text on the object is uniform in font, size, color, and background. For example:  The single text block mode instructs the OCR mechanism to focus on the area and treat it as a single text block. This is especially useful when trying to capture text on small objects or in a small text area.

Option	Description
Multiple text block mode	Select this radio button only if the text on the object comprises different fonts, font sizes, colors, and/or backgrounds. For example:  The multiple text block mode instructs the OCR mechanism to handle each text area in the object that has a different background font and size. The OCR mechanism decides where to divide the text blocks according to an internal algorithm.
Available languages	Lists all of the language dictionaries that the OCR mechanism can potentially use when retrieving text from the object. **To specify the language dictionaries used by the OCR mechanism:** Move a language to the **Supported languages** list box by selecting a language and clicking the right arrow button (>).
Supported languages	Lists the language dictionaries that the OCR mechanism uses when capturing text. The **Supported languages** list box can contain either: ➤ One CJK (Chinese, Japanese, Korean) language. (**Note:** By default, English is also supported when capturing text in CJK languages.) ➤ One or more non-CJK languages. **To remove a language dictionary from the Supported languages list:** Select the language and click the left arrow button (<).

Additional References

Related Use Case Scenarios	"Use-Case Scenario: Checking Text in an Image" on page 750
Related Tasks	➤ "Creating a Text Checkpoint" on page 552 ➤ "Creating a Text Area Checkpoint" on page 554 ➤ "Outputting Text Values" on page 688
Related Concepts	➤ "Guidelines for Text Recognition" on page 746 ➤ "Checking Text" on page 551

Guidelines for Text Recognition

- When using the OCR mechanism, the larger the text, the better the text recognition.
- Try to keep the dimensions of the selected text area as small as possible, as this helps prevent additional unwanted characters in recognized text.

  At the same time, consider the potential movement (change of coordinates) of the object within the window. For example, the screen resolution is often different on different computers, and this can affect the coordinates of the object in the application. Also, during the design and development stages of an application, an object may be moved to make room for other objects or for aesthetic purposes.

  Consider that the operating system, installed service packs, installed toolkits, and so on, can all affect the size and location of an object in an application. Make sure that the dimensions of the selected text area are large enough for different system configurations.

  The dimensions of the selected text area need to be large enough to take these issues into account.
- If you are not sure which text block mode to use, first use the single text block mode, as text captures performed on single text blocks are generally more accurate than text captures on multiple text blocks. If the results are not what you expect, then try using the multiple text block mode. For an example of when to use different text block modes, see "Use-Case Scenario: Checking Text in an Image" on page 750.

Tip: If you want to use the text recognition mechanism for a large area containing different fonts and backgrounds, it is recommended to create several steps to capture the text for each single text block instead of creating one step to capture a multiple text block.

- Windows provides various themes. When working with text recognition, try to apply themes in the following order:
  - Windows Vista theme (for best results)
  - Windows XP theme
  - Windows Classic theme
- If the text recognition mechanism retrieves unwanted text information (such as hidden text and shadowed text that appears as multiple copies of the same string), when using the multiple text block mode, use the single text block mode option.
- If your text recognition options are set to use the Windows API mechanism, then, when running a step that uses text recognition, the Windows API may cause a "blinking effect" in your application as it captures the text. If your test contains consecutive steps that utilize the text recognition mechanism, the "blinking effect" in one step may cause the subsequent text recognition step (or other step that relies on the appearance of the application, such as a bitmap checkpoint) to fail.

  To address this, you can insert a **Wait** statement prior to each such step. This enables you to delay the performance of the next text recognition step until the Windows API capture of the previous step is complete.
- It is highly recommended to check text from your application window by inserting a standard checkpoint for the object containing the desired text, using its **text** (or similar) property.

Note: If you are creating text area checkpoints, see “Considerations for Defining the Text Area” on page 556 for additional guidelines.

Text Recognition and Development Environments

The following table lists the development environments supported by QuickTest (via its add-ins), and specifies what is supported for text recognition.

Development Environment	Text Recognition	
	Supported	**Not Supported**
ActiveX	Full text recognition support	N/A
Delphi	Full text recognition support	N/A
Java	➤ Text checkpoints ➤ Text output values ➤ Text area checkpoints ➤ Text area output values	➤ **GetTextLocation** method ➤ **GetVisibleText** method
.NET WebForms	➤ Text checkpoints for Page object only ➤ Text output values for Page object only	➤ Text checkpoints for all other objects ➤ Text output values for all other objects ➤ Text area checkpoints for all other objects ➤ Text area output values for all other objects ➤ **GetTextLocation** method ➤ **GetVisibleText** method
.NET WinForms	Full text recognition support	N/A
Oracle	N/A	No text recognition support

Development Environment	Text Recognition	
	Supported	**Not Supported**
PeopleSoft	➤ Text checkpoints for PSFrame object only ➤ Text output values for PSFrame object only	➤ Text checkpoints for all other objects ➤ Text output values for all other objects ➤ Text area checkpoints for all other objects ➤ Text area output values for all other objects ➤ **GetTextLocation** method ➤ **GetVisibleText** method
PowerBuilder	Full text recognition support	N/A
SAP Gui for Windows	N/A	No text recognition support
SAP Web	➤ Text checkpoints ➤ Text output values	➤ Text area checkpoints ➤ Text area output values ➤ **GetTextLocation** method ➤ **GetVisibleText** method
Siebel	N/A	No text recognition support
Standard Windows	Full text recognition support	N/A
Stingray	Full text recognition support	N/A
Terminal Emulators	Text output values for TeScreen and TeTextScreen objects only	➤ Other text checkpoints ➤ Other text output values ➤ Text area checkpoints ➤ Text area output values ➤ **GetTextLocation** method ➤ **GetVisibleText** method
VisualAge	Full text recognition support	N/A
Visual Basic	Full text recognition support	N/A

Development Environment	Text Recognition	
	Supported	**Not Supported**
Web	➤ Text checkpoints for Page object only ➤ Text output values for Page object only	➤ Text checkpoints for all other objects ➤ Text output values for all other objects ➤ Text area checkpoints for all other objects ➤ Text area output values for all other objects ➤ **GetTextLocation** method ➤ **GetVisibleText** method
Web Services	N/A	No text recognition support
WPF	Full text recognition support	N/A

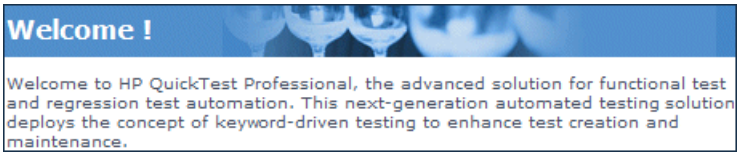
Use-Case Scenario: Checking Text in an Image

Ben and George are quality assurance engineers who are experienced QuickTest users. George is also familiar with text recognition and has a basic understanding of how text recognition mechanisms work.

Ben often uses bitmap checkpoints to test the appearance of different icons or pictures in the user interface he is testing.

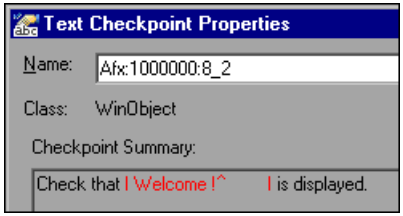
For one of his projects, Ben also needed to verify the text in the graphics, so he decided to use text checkpoints.

Ben decided to begin the verification process by inserting a text checkpoint to check that the text Welcome ! was displayed correctly in the following graphic.



Before inserting the text checkpoint, Ben opened the Text Recognition pane and configured the text recognition settings. Ben set the text recognition mechanism to **Use Only OCR** because the text was part of a graphic. Ben also knew that single text block mode usually works best, so he selected the **Single text block mode** option.

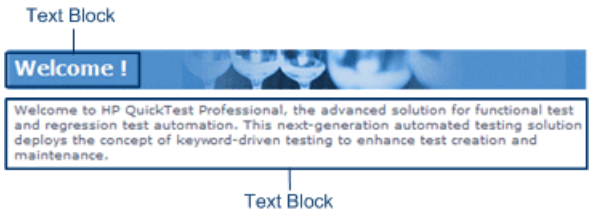
Ben then inserted a text checkpoint on the entire area shown above and received the following results in the Text Checkpoint Properties dialog box:



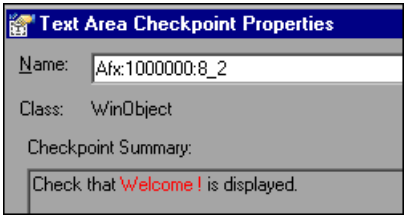
Ben noticed that there were extra characters in the **Checkpoint Summary** area of the text checkpoint, but he did not know why.

Ben asked his colleague, George, for help. George explained to him that the text recognition mechanism sometimes adds extra characters to the text checkpoint when it does not recognize the text correctly.

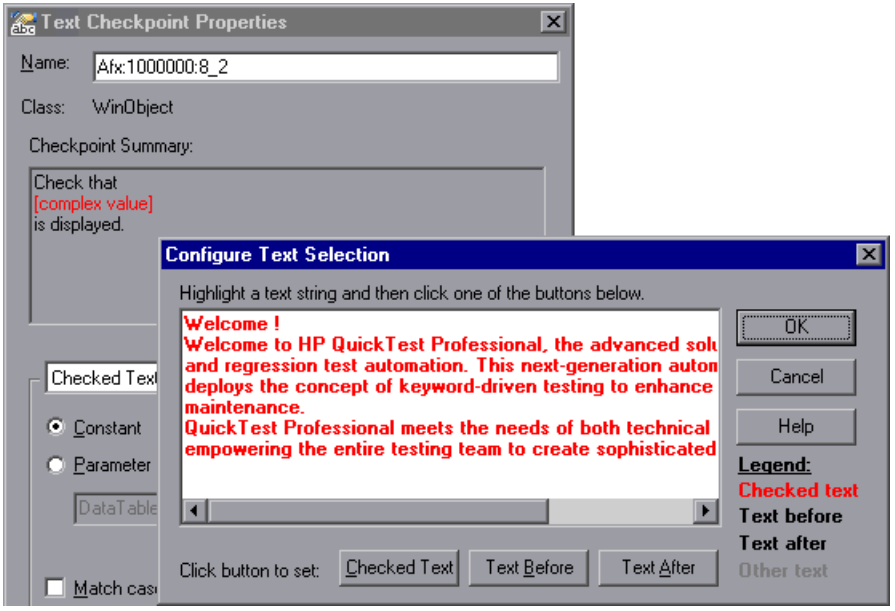
George also pointed out that the area Ben defined for the text checkpoint consisted of multiple text blocks because the text was not uniform in font size, color, or background. The title area consisted of white characters on a blue-gray background, while the remaining text was smaller and consisted of blue text on a white background.



Ben remembered that he had selected the **Single text block mode** option in the General > Text Recognition pane and understood that if he wanted to use single text block mode, he would have to create a text checkpoint only on the Welcome ! area of the graphic, and not on the entire graphic. Ben tried this, and the OCR mechanism correctly identified the text, as shown below:



Ben was pleased with the results, but he wanted to explore other possibilities, so he inserted another text checkpoint—this time on the entire graphic. He selected the **Multiple text block mode** option in the Text Recognition pane, which resulted in the following:



Ben was pleased that the OCR mechanism correctly recognized all of the text in the graphic. But he needed to test only the title, Welcome !, so he finalized this checkpoint by marking all of the text after Welcome ! as **Text After**.

Even though both checkpoints passed, Ben needed only one text checkpoint. He decided to keep the first checkpoint (that used **Single text block mode**), and he deleted the second one. He selected the **Single text block mode** option in the Text Recognition pane to help ensure that the checkpoint would pass in future test runs.

27

Configuring Values

QuickTest enables you to configure the values for properties and other items by defining a value as a constant or a parameter. You can also use regular expressions in values to increase the flexibility and adaptability of your tests.

This chapter includes:

About Configuring Values

Some dialog boxes, such as the Checkpoint Properties dialog boxes, include a **Configure value** area, in which you can define the value for a selected item as a constant or a parameter. In other contexts, such as the Keyword View, Step Generator, and Object Repository window, you can select a value directly and parameterize it or define it as a constant.

- **Constant.** A value that is defined directly in the step and remains unchanged for the duration of the test.
- **Parameter.** A value that is defined or generated separately from the step and is retrieved when the specific step runs. For example, a parameter value may be defined in an external file or generated by QuickTest.

When you define a value as a parameter, you can also specify other settings according to the parameter type. For more information on using parameters in your tests, see Chapter 24, "Parameterizing Values."

You can edit a constant value in the **Configure value** area. In certain contexts, you can define a constant value using a regular expression.

A regular expression is a string that specifies a complex search phrase. Regular expressions are used to identify objects and text strings with varying values. For example, if the name of a window's title bar changes according to a file name, you can use a regular expression in a test object description to identify a window whose title bar has the specified product name, followed by a hyphen, and then any other text.

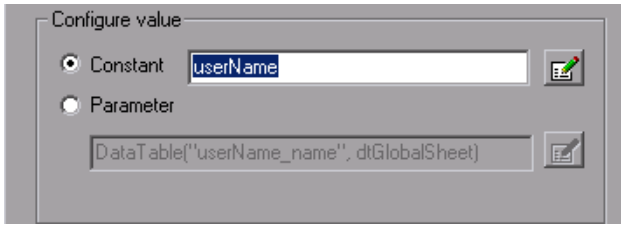
Configuring Constant and Parameter Values

You can define a value as a constant or a parameter in several ways:

- In the Value Configuration Options dialog box, you can click the parameterization button  for a selected value, for example, in the Keyword View, Step Generator, or Object Repository window. For more information, see "Configuring a Selected Value" on page 760.
- In the **Configure value** area of a dialog box, you can select a property or argument, for example, in the Checkpoint Properties dialog box.

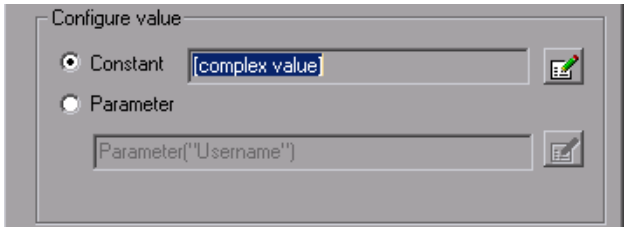
Setting Values in the Configure Value Area

When you select an item in a dialog box containing a **Configure value** area, such as the Checkpoint Properties dialog box, you can select **Constant** or **Parameter** to set the value. The default is **Constant**.



If you select **Constant**, you can edit a single-line value directly in the **Constant** box. If it is a **string** value, you can also click the **Constant Value Options** button to define the value as a regular expression. For information on regular expressions, see "Understanding and Using Regular Expressions" on page 762.

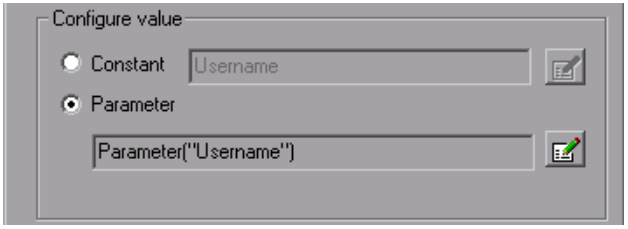
If the entire value cannot be displayed in the **Constant** box, it is shown as **[complex value]**. For example, the value of a list's **all items** property is a multi-line value, where each line contains the value of an item in the list.



You can view or edit a complex value by clicking the **Constant Value Options** button. You can also define a complex value as a regular expression. For more information on editing constant values, see "Setting Constant Value Options" on page 759.

Configuring a Parameter Value

If you select **Parameter** for a value that is already parameterized, the **Parameter** box displays the current parameter definition for the value. If you select **Parameter** for a value that is not yet parameterized, the **Parameter** box displays the default parameter definition for the value.



For more information on default parameter definitions, see "Understanding Default Parameter Values" on page 634.

You can click the **Parameter Options** button to select a different parameter type or modify the parameter settings for the value.

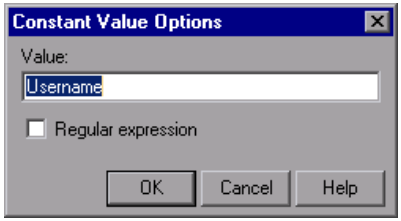
The Parameter Options dialog box opens for the displayed parameter type. For more information on defining values for specific parameter types, see:

- "Setting Test and Action Parameter Options" on page 636
- "Setting Data Table Parameter Options" on page 641
- "Setting Environment Variable Parameter Options" on page 652
- "Using Random Number Parameters" on page 655

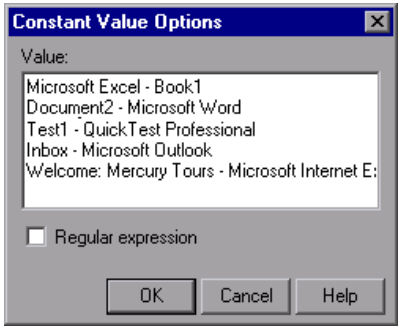
For more information on using parameters in your tests, see Chapter 24, "Parameterizing Values."

Setting Constant Value Options

When you click the **Constant Value Options** button in the **Configure value** area, the Constant Value Options dialog box opens.



For a complex value (a value that cannot be displayed entirely in the **Constant** box), the Constant Value Options dialog box expands to show the entire contents of the value.

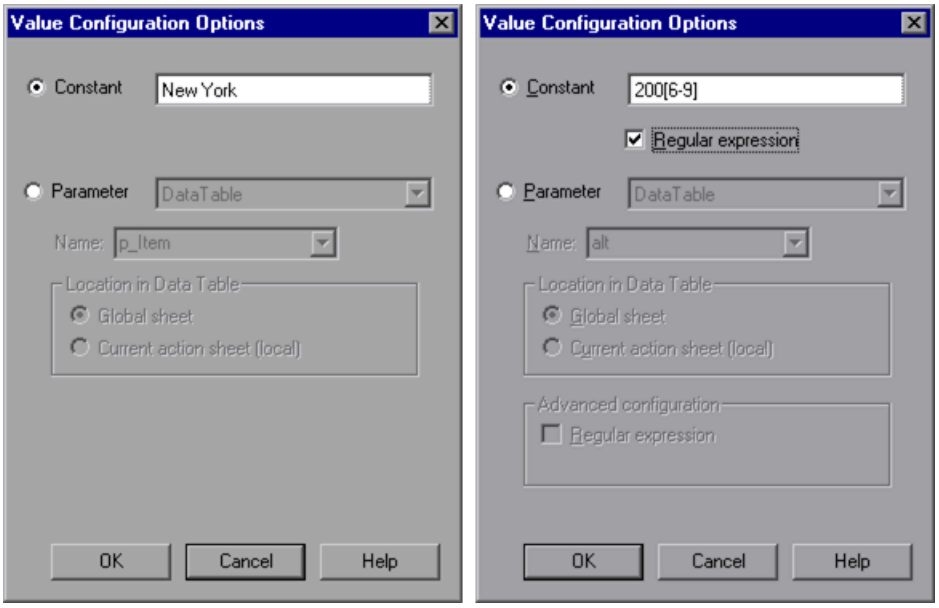


You can update the following options to edit the value of the constant:

- **Value.** Specifies the value for the constant.
- **Regular expression.** Sets the defined value as a regular expression:
  - For general information on regular expressions, see “Understanding and Using Regular Expressions” on page 762.
  - For information on defining a regular expression, see “Defining Regular Expressions” on page 765.

Configuring a Selected Value

When you click the parameterization button for a selected value, the Value Configuration Options dialog box opens. In some situations, you can also define the constant or parameter using a regular expression. (The following examples illustrate the Value Configuration Options dialog box with and without the **Regular expression** check box.)



Note: The parameter options shown in this dialog box change according to the parameter type selected in the **Parameter** box.

You can select one of the following options:

- **Constant.** Defines a value that remains unchanged for the duration of the test. You can edit the value directly in the **Constant** box. This box offers the same editing options as the **Value** cell in which you clicked the parameterization button to open this dialog box. For more information on these options, see "Defining Values for Your Step Arguments" on page 404.

  In some situations, for example, when parameterizing an object identification property value, you can also specify a constant value using a regular expression (by using a regular expression in the **Constant** box and selecting the **Regular expression** check box). For information on regular expressions, see "Understanding and Using Regular Expressions" on page 762.

- **Parameter.** Specifies a value that is defined or generated separately from the step and is retrieved when the specific step runs.

  If you select **Parameter** for a value that is already parameterized, the **Parameter** section displays the current parameter type and details for the value. If you select **Parameter** for a value that is not yet parameterized, the **Parameter** section displays the default parameter type and details for the value.

  For more information on default parameter definitions, see "Understanding Default Parameter Values" on page 634.

  You can change the default definition by selecting a different parameter type or modifying the parameter settings for the value. The options in the **Parameter** section change according to the parameter type you select.

Note: If you are using an environment variable to parameterize an argument that receives a predefined constant or number, only the environment variable parameters whose value is of type **integer** are shown in the **Name** list.

The **Parameter** section of the Value Configuration Options dialog box is very similar to the Parameter Options dialog box. For more information on configuring values for specific parameter types, see:

- "Defining the Settings for a Test or Action Parameter" on page 637
- "Defining the Settings for a Data Table Parameter" on page 642
- "Defining the Settings for an Environment Variable Parameter" on page 652
- "Defining Settings for a Random Number Parameter" on page 656

For more information on using parameters in your tests, see Chapter 24, "Parameterizing Values."

Understanding and Using Regular Expressions

Regular expressions enable QuickTest to identify objects and text strings with varying values. You can use regular expressions in the following situations:

- When defining the property values of an object in dialog boxes or in programmatic descriptions
- When parameterizing a step
- When creating checkpoints with varying values

For example, you can use a regular expression if the text property of an object is a date value, but the displayed date changes according to the current date. If you define the date as a regular expression, QuickTest can identify the object that contains text with the expected date format, rather than the exact date value.

A regular expression is a string that specifies a complex search phrase. By using special characters, such as a period (.), asterisk (*), caret (^), and brackets ([ ]), you can define the conditions of a search.

Notes:

- You can use regular expressions only for values of type **string**.
- When any special character in a regular expression is preceded by a backslash (\), QuickTest searches for the literal character.

For more information and examples of the use of regular expressions, see:

- "Using Regular Expressions for Property Values" on page 763
- "Using Regular Expressions in Checkpoints" on page 764

For information on defining regular expressions, including regular expression syntax, see "Defining Regular Expressions" on page 765.

Using Regular Expressions for Property Values

If you expect the value of a property to change in a predictable way during each run session, you can use regular expressions when defining or parameterizing property values, for example, in the Object Repository window, or in programmatic descriptions. For more information on programmatic descriptions, see "Using Programmatic Descriptions" on page 863.

For example, your Web site may include a form in which the user inputs data and clicks the **Send** button to submit the form. When a required field is not completed, the form is displayed again for the user to complete. When resubmitting the form, the user clicks the **Resend** button. You can define the value of the button's **name** property as a regular expression, so that QuickTest ignores variations in the button name when clicking the button.

Using Regular Expressions in Checkpoints

When creating a standard checkpoint to verify the property values of an object, you can set the expected value of an object's property as a regular expression so that an object with a varying value can be verified.

For example, suppose you want to check that every window and dialog box in your application contains the name of your application followed by a hyphen (-) and a descriptive title. You can add a checkpoint to each dialog box object in your test to check that the first part of the title contains the name of your application followed by a hyphen.

When creating a text checkpoint to check that a varying text string is displayed on your application, you can define the text string as a regular expression.

For example, when booking a flight in the Mercury Tours sample Web site, the total cost charged to a credit card number should not be less than $300. You define the amount as a regular expression, so that QuickTest will ignore variations in the text string as long as the value is not less than $300.

You can apply the same principles to any checkpoint type whose dialog box contains a **Configure Value** area similar to that described in "Configuring Constant and Parameter Values" on page 756.

For example, for table checkpoints you can set cell values as regular expressions, and for XML checkpoints you can set attribute or element values as regular expressions. For more information on specific checkpoint types, see the relevant chapter for the checkpoint type.

Defining Regular Expressions

You can define a regular expression for a constant value, a Data Table parameter value, an Environment parameter value, or a property value in a programmatic description. For more information on defining property values, see "Configuring Constant and Parameter Values" on page 756.

You can define a regular expression by entering the regular expression syntax for the string in the **Value** box in the Constant Value Options dialog box or the Parameter Options dialog box. You instruct QuickTest to treat the value as a regular expression by selecting the **Regular Expression** check box.

All programmatic description property values are automatically treated as regular expressions. For more information on programmatic descriptions, see "Using Programmatic Descriptions" on page 863.

Note: You can use regular expressions only for values of type **string**.

By default, QuickTest treats all characters in a regular expression literally, except for the period (.), hyphen (-), asterisk (*), caret (^), brackets ([ ]), parentheses (()), dollar sign ($), vertical line (|), plus sign (+), question mark (?), and backslash (\). When one of these special characters is preceded by a backslash (\), QuickTest treats it as a literal character.

If you enter a special character in the **Value** box of the Constant Value Options or the Parameter Options dialog box, QuickTest asks you if you want to add a backslash (\) before each special character. If you click **Yes**, a backslash (\) is added before the special character to instruct QuickTest to treat the character literally. If you click **No**, QuickTest treats the special character as a regular expression character.

This section describes some of the more common options that can be used to create regular expressions:

- Using the Backslash Character (\ )
- Matching Any Single Character (**.**)
- Matching Any Single Character in a List (**[***xy***]**)
- Matching Any Single Character Not in a List (**[^***xy***]**)
- Matching Any Single Character within a Range (**[***x*-*y***]**)
- Matching Zero or More Specific Characters (*****)
- Matching One or More Specific Characters (**+**)
- Matching Zero or One Specific Character (**?**)
- Grouping Regular Expressions (**( )**)
- Matching One of Several Regular Expressions (**|**)
- Matching the Beginning of a Line (**^**)
- Matching the End of a Line (**$**)
- Matching Any AlphaNumeric Character Including the Underscore (**\w**)
- Matching Any Non-AlphaNumeric Character (**\W**)
- Combining Regular Expression Operators

Note: For a complete list and explanation of supported regular expressions characters, see the Regular Expressions section in the Microsoft VBScript documentation (select **Help** > **QuickTest Professional Help** to open the QuickTest Professional Help. Then select **VBScript Reference** > **VBScript** > **User's Guide** > **Introduction to Regular Expressions**).

Using the Backslash Character

A backslash (\) can serve two purposes. It can be used in conjunction with a special character to indicate that the next character be treated as a literal character. For example, \. would be treated as period (.) instead of a wildcard. Alternatively, if the backslash (\) is used in conjunction with some characters that would otherwise be treated as literal characters, such as the letters n, t, w, or d, the combination indicates a special character. For example, \n stands for the newline character.

For example:

- w matches the character w
- \w is a special character that matches any word character including underscore
- \\ matches the literal character \
- \(matches the literal character (

For example, if you were looking for a Web site called:

```
newtours.demoaut.com
```

the period would be mistaken as an indication of a regular expression. To indicate that the period is not part of a regular expression, you would enter it as follows:

```
newtours\.demoaut\.com
```

Note: If a backslash character is used before a character that has no special meaning, the backslash is ignored. For example, \z matches z.

Matching Any Single Character

A period (.) instructs QuickTest to search for any single character (except for \n). For example:

```
welcome.
```

matches welcomes, welcomed, or welcome followed by a space or any other single character. A series of periods indicates the same number of unspecified characters.

To match any single character including \n, enter:

```
(.|\n)
```

For more information on the **( )** regular expression characters, see "Grouping Regular Expressions" on page 770. For more information on the **|** regular expression character, see "Matching One of Several Regular Expressions" on page 771.

Matching Any Single Character in a List

Square brackets instruct QuickTest to search for any single character within a list of characters. For example, to search for the date 1967, 1968, or 1969, enter:

```
196[789]
```

Matching Any Single Character Not in a List

When a caret (^) is the first character inside square brackets, it instructs QuickTest to match any character in the list except for the ones specified in the string. For example:

```
[^ab]
```

matches any character except a or b.

Note: The caret has this special meaning only when it is displayed first within the brackets.

Matching Any Single Character within a Range

To match a single character within a range, you can use square brackets ([ ]) with the hyphen (-) character. For instance, to match any year in the 1960s, enter:

```
196[0-9]
```

A hyphen does not signify a range if it is displayed as the first or last character within brackets, or after a caret (^).

For example, [-a-z] matches a hyphen or any lowercase letter.

Note: Within brackets, the characters ".", "*", "[" and "\" are literal. For example, [.*] matches . or *. If the right bracket is the first character in the range, it is also literal.

Matching Zero or More Specific Characters

An asterisk (*) instructs QuickTest to match zero or more occurrences of the preceding character. For example:

```
ca*r
```

matches car, caaaaaar, and cr.

Matching One or More Specific Characters

A plus sign (+) instructs QuickTest to match one or more occurrences of the preceding character. For example:

```
ca+r
```

matches car and caaaaaar, but not cr.

Matching Zero or One Specific Character

A question mark (?) instructs QuickTest to match zero or one occurrences of the preceding character. For example:

```
ca?r
```

matches car and cr, but nothing else.

Grouping Regular Expressions

Parentheses (()) instruct QuickTest to treat the contained sequence as a unit, just as in mathematics and programming languages.

Using groups is especially useful for delimiting the argument(s) to an alternation operator (|) or a repetition operator (* , + , ? , { }).

Matching One of Several Regular Expressions

A vertical line (|) instructs QuickTest to match one of a choice of expressions. For example:

```
foo|bar
```

causes QuickTest to match either foo or bar.

```
fo(o|b)ar
```

causes QuickTest to match either fooar or fobar.

Matching the Beginning of a Line

A caret (^) instructs QuickTest to match the expression only at the start of a line, or after a newline character.

For example:

```
book
```

matches book within the lines—book, my book, and book list, while

```
^book
```

matches book only in the lines—book and book list.

Matching the End of a Line

A dollar sign ($) instructs QuickTest to match the expression only at the end of a line, or before a newline character. For example:

```
book
```

matches book within the lines—my book, and book list, while a string that is followed by ($), matches only lines ending in that string. For example:

```
book$
```

matches book only in the line—my book.

Matching Any AlphaNumeric Character Including the Underscore

\w instructs QuickTest to match any alphanumeric character and the underscore (A-Z, a-z, 0-9, _).

For example:

\w* causes QuickTest to match zero or more occurrences of the alphanumeric characters—A-Z, a-z, 0-9, and the underscore (_). It matches Ab, r9Cj, or 12_uYLgeu_435.

For example:

\w{3} causes QuickTest to match 3 occurrences of the alphanumeric characters A-Z, a-z, 0-9, and the underscore (_). It matches Ab4, r9_, or z_M.

Matching Any Non-AlphaNumeric Character

\W instructs QuickTest to match any character other than alphanumeric characters and underscores.

For example:

\W matches &, *, ^, %, $, and #

Combining Regular Expression Operators

You can combine regular expression operators in a single expression to achieve the exact search criteria you need.

For example, you can combine the '.' and '*' characters to find zero or more occurrences of any character (except \n).

For example,

start.*

matches start, started, starting, starter, and so forth.

You can use a combination of brackets and an asterisk to limit the search to a combination of non-numeric characters. For example:

```
[a-zA-Z]*
```

To match any number between 0 and 1200, you need to match numbers with 1 digit, 2 digits, 3 digits, or 4 digits between 1000-1200.

The regular expression below matches any number between 0 and 1200.

```
([0-9]?[0-9]?[0-9]|1[01][0-9][0-9]|1200)
```

28

Adding Steps Containing Programming Logic

After creating a test, you can use special QuickTest tools to enhance it with programming statements, even if you choose not to program manually in the Expert View.

This chapter includes:

About Adding Steps Containing Programming Logic

When you design tests, you usually begin by adding steps that represent the operations an end-user would perform as part of the business process you want to test. Then, to increase the power and flexibility of your test, you can add steps that contain programming logic to the basic framework.

Programming statements can contain:

- **Test object operations.** These are methods and properties defined by QuickTest. They can be operations that a user can perform on an object, operations that can retrieve or set information, or operations that perform operations triggered by an event.
- **Native operations.** These are methods and properties defined within the object you are testing, and therefore are retrieved from the run-time object in the application.
- VBScript programming commands that affect the way the test runs, such as conditions and loops. These are often used to control the logical flow of a test.
- Supplemental statements, such as comments, to make your test easier to read, and messages that appear in the test results, to alert you to a specified condition.

This chapter shows you how to insert different types of statements, mostly from the Keyword View, aided by the Step Generator and other dialog boxes.

The Step Generator dialog box helps you add steps that use test object operations, Utility object operations, and function calls, so that you do not need to memorize syntax or to be proficient in high-level VBScript. You can use the Step Generator from the Keyword View and also from the Expert View.

For information on how to insert statements in the Expert View, see Chapter 29, "Working in the Expert View and Function Library Windows."

You can incorporate decision-making into your test and define messages for the test results by using the appropriate dialog boxes.

In addition, you can improve the readability of your test using **With** statements. You can instruct QuickTest to automatically generate **With** statements as you record. But even after your basic test is recorded, you can convert its statements, in the Expert View, to **With** statements—by selecting a menu command.

You can handle synchronization issues between the run session and your application, using synchronization points.

When working with tests, you can also measure how long it takes certain parts of your test to run, using transaction statements.

Inserting Steps Using the Step Generator

The Step Generator enables you to add steps by selecting from a range of context-sensitive options and entering the required values. In the Step Generator dialog box you can define steps that use:

- test object operations (tests only)
- Utility object operations
- calls to library functions (tests only), VBScript functions, and internal script functions

For example, you can add a step that checks that an object exists, or that stores the returned value of a method as an output value or as part of a conditional statement. You can parameterize any of the values in your step.

Note: You can use the Step Generator to insert steps in tests and function libraries. However, in function libraries, you cannot use the Step Generator to access test object names or collections, or to access the list of library functions.

Before you open the Step Generator to define a new step, you first select where in your test the new step should be inserted. For more information on the hierarchy of steps and objects, see "Understanding the QuickTest Object Hierarchy" on page 391.

After you open the Step Generator, you first select the category for the step operation (test object, Utility object or function) and the required object or the function library source (for example, built-in or local script functions). You can then select the appropriate operation (method, property, or function) and define the arguments and return values, parameterizing them if required.

The Step Generator then inserts a step with the correct syntax into your test. You can continue to add further steps at the same location without closing the Step Generator.

You can open the Step Generator from the Keyword View, Expert View, or Active Screen.

To open the Step Generator from the Keyword View or Expert View:

1 While recording or editing, click the step which you want the new step to follow. (When you finish defining the new step, QuickTest will insert it after this step.)

2 Select **Insert** > **Step Generator** or right-click the step and select **Insert Step** > **Step Generator**. Alternatively, press F7.

The Step Generator dialog box opens and displays the object from the selected step in the **Object** box. For more information, see "Defining Steps in the Step Generator Dialog Box" on page 780.

To open the Step Generator from a function library:

1 In the function library, click the location in which you want to insert the new step.

2 Select **Insert** > **Step Generator**, or right-click and select **Step Generator**. Alternatively, press F7.

The Step Generator dialog box opens. For more information, see "Defining Steps in the Step Generator Dialog Box" on page 780.

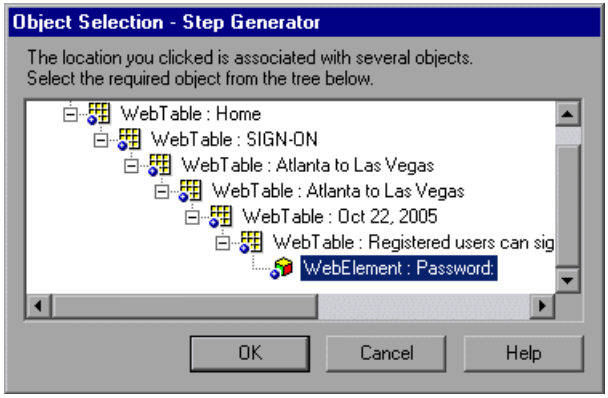
To open the Step Generator from the Active Screen while editing:

1 Confirm that the Active Screen is displayed. If it is not, select **View** > **Active Screen** or toggle the **Active Screen** toolbar button.

2 In the Keyword View or Expert View, click the step which you want the new step to follow. (When you finish defining the new step, QuickTest will insert it after this step.) The Active Screen displays the captured bitmap or HTML source corresponding to the selected step.

3 In the Active Screen, right-click the object for which you want to insert a step, and select **Step Generator**.

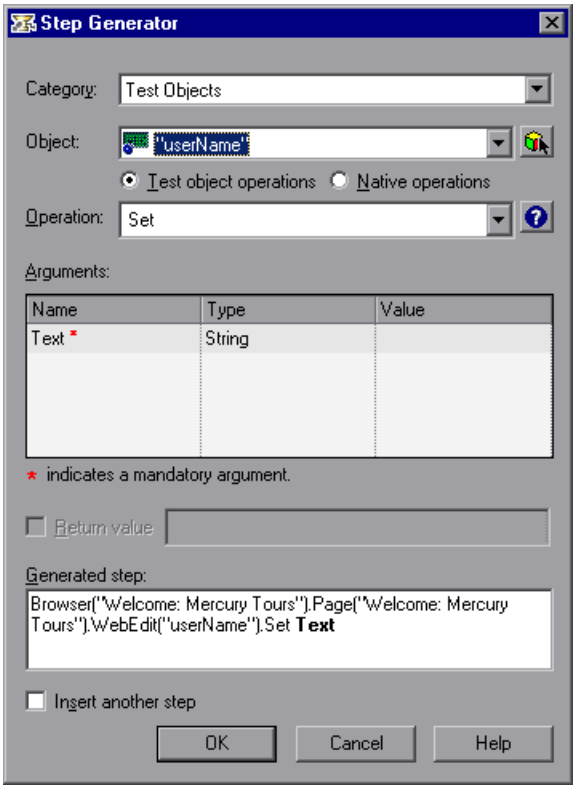
If the location you clicked is associated with more than one object, the **Object Selection - Step Generator** dialog box opens.



4 Select an object and click **OK**. The Step Generator dialog box opens and displays the object from the selected step in the **Object** box. For more information, see “Defining Steps in the Step Generator Dialog Box” on page 780.

Defining Steps in the Step Generator Dialog Box

The Step Generator dialog box enables you to add steps that perform operations, using test object methods (for tests only), Utility object methods, or function calls.



Note: The Step Generator dialog box that opens from the Expert View and from a function library is similar to the dialog box that opens from the Keyword View (shown in the example above).

In the Expert View, the Step Generator contains additional Utility objects and the box at the bottom of the dialog box shows a preview of the step that will be inserted in the Expert View. For more information, see "Viewing the Generated Step in the Expert View" on page 785.

In a function library, the Step Generator has a different title, contains only Utility objects and built-in and local script functions, and the box at the bottom of the dialog box shows a preview of the statement that will be inserted in the function library. For more information, see "Viewing the Generated Step in a Function Library" on page 785.

When the Step Generator dialog box opens, the object from the selected step is displayed in the **Object** box and the default method for the object is shown in the **Operation** box.

Defining a New Step

When you define a new step, you first select the type of step that you want to add to your test. You can then select the specific object and operation for the step, or the function that you want the step to use.

After you select the operation for the step, you can specify the relevant argument values and the location for the return value, if applicable. These values can be parameterized if required.

Finally, you can view the step documentation or statement syntax and add your new step or statement to your test or function library.

Note: Although the Step Generator shows information regarding the currently selected step, selections that you make in the Step Generator add a new step to your test; they do not modify the existing step.

Selecting the Type of Step to Add

In the **Category** list box, you can select one of the following options:

- **Test Objects.** Enables you to select a test object and operation for the step (for tests only). For more information, see "Specifying a Test Object and Operation for the Step" on page 786.
- **Utility Objects.** Enables you to select a Utility object and operation for the step. For more information, see "Specifying a Utility Object and Operation for the Step" on page 791.
- **Functions.** Enables you to select a function for the step from the available library functions (tests only), VBScript functions, and internal script functions. For more information, see "Specifying a Function for the Step" on page 793.

Specifying Argument Values

After you select the object and the operation (method, property, or function) for the step, you can specify the relevant argument values. These values can be parameterized if required.

If the selected operation has arguments, the **Arguments** area displays the name and type of each argument.

In the **Value** column, you can define the values for the arguments, as follows:

- **Mandatory arguments.** If the name of the argument is followed by a red asterisk (*), you must specify a value for the argument. You cannot insert the step or view the step documentation if the values have not been defined for all mandatory arguments.
- **Optional arguments.** If the name of the argument is not followed by a red asterisk (*), you can specify a value for the argument or leave the cell blank. If you do not specify a value, QuickTest uses the default value for the argument. (You can view the default value by moving the pointer over the cell).
- **Required arguments.** If you specify a value for an optional argument, then you must also specify the values for any optional arguments that are listed before this argument. If you do not specify these values, QuickTest uses the default values for all required arguments. You can see the default value for each argument in a tooltip, by moving the pointer over the **Value** column.
- **Parameterized arguments.** You can use a parameter for any argument value by clicking the parameterization button  . For more information, see "Configuring a Selected Value" on page 760.
- **Predefined constants.** If an argument has a predefined list of values, QuickTest provides a drop-down list of possible values. If a list of values is provided, you cannot manually type a value in this box.

Specifying the Location for the Return Value

If the selected operation returns a value, you can specify that you want to store the value by selecting the **Return Value** check box. When this check box is selected, a default variable is displayed as the return value location.

You can supply a different variable definition by editing the value. You can select a different storage location for the return value by clicking the displayed value and then the output storage button  . For more information, see "Storing Return Values and Action Output Parameter Values" on page 794.

Viewing the Step Documentation in the Keyword View

If you open the Step Generator from the Keyword View, the **Step documentation** box at the bottom of the Step Generator dialog box can display summary information on the current step in an easy-to-read sentence.

If you select either the **Test Object** or **Utility Object** category and you define all the mandatory and required values for the current operation, the **Step documentation** box describes the operation performed by the step. When the step is inserted into your test, this description is displayed in the **Documentation** column in the Keyword View.

If all the mandatory and required argument values have not been defined for the operation, the **Step documentation** box displays a warning message.

Note: If you select the **Functions** category, step documentation is available for user-defined functions, if you provided this information when defining them. For more information, see "Documenting the Function" on page 934.

Viewing the Generated Step in the Expert View

If you open the Step Generator from the Expert View, the **Generated step** box displays the defined statement for the step.

If all the mandatory and required argument values have not been defined for the operation, the names of the undefined arguments are highlighted in bold text. If you attempt to insert the step, an error message is displayed.

Viewing the Generated Step in a Function Library

If you open the Step Generator from a function library, the **Generated step** box displays the defined statement for the step.

If all the mandatory and required argument values have not been defined for the statement, the names of the undefined arguments are highlighted in bold text. If you attempt to insert the step, an error message is displayed.

Inserting Steps

After you define all mandatory argument values for the current operation, the following options are available:

- To insert the current step and close the Step Generator, make sure the **Insert another step** check box is cleared. When you click **OK**, the step is added to your test and the Step Generator dialog box closes.
- To insert the current step and continue adding steps at the same location, select the **Insert another step** check box. The **OK** button changes to **Insert**. When you click **Insert**, the current step is added to your test and the Step Generator dialog box remains open, enabling you to define another step.

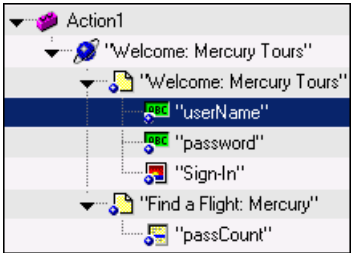
  When you insert a new step using the Step Generator, it is added to your test after the selected step, and the new step is selected. For more information on the hierarchy of steps and objects and the positioning of new steps, see "Understanding the QuickTest Object Hierarchy" on page 391.

Specifying a Test Object and Operation for the Step

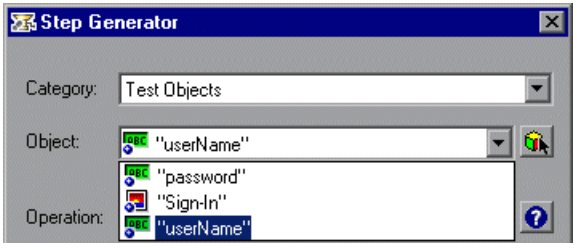
If you select **Test Objects** in the **Category** list in the Step Generator dialog box, you can select the object for the new step in the context of the currently selected step in your test. Alternatively, you can select any object from the object repository or from your application.

The list in the **Object** box contains all the objects in the object repository that are at the same hierarchical level and location as the currently selected step. You can select any of these objects for your new step.

For example, suppose that you selected the step for the **userName** object in the **Welcome: Mercury Tours** Web page, as shown below:



When you open the Step Generator, **Test Objects** is selected in the **Category box**, and the **Object** box lists the **userName**, **password** and **Sign-in** objects.



Note: The objects are listed by name in alphabetical order.

You can select an object from the object repository or from your application, by clicking the **Select Object** button. For more information, see "Selecting an Object from the Repository or Application" on page 788.

After you select the object for the step, you can select the required operation type (test object operation or, if available, native (run-time object) operation) and then you can select the operation for the step.

Selecting the Operation for a Test Object

If QuickTest can retrieve native (run-time object) operations for the selected test object, you can select the operation type—**Test object operation** or **Native operation**. (If QuickTest cannot retrieve native operations for the selected object, the **Native operations** option is not available.)

The **Operation** box displays the default operation for the selected object. You can select a different operation from the **Operation** box list, which contains all the operations that are available for the selected object.

For detailed information on a test object operation and its syntax, you can click the **Operation Help** button to open the *HP QuickTest Professional Object Model Reference* for the selected operation.

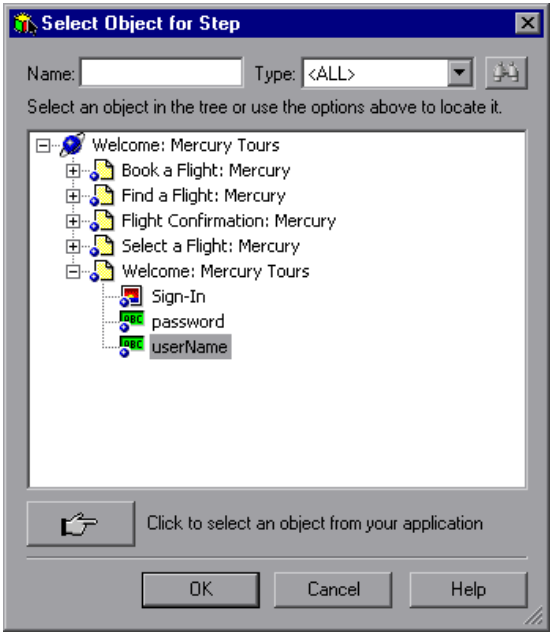
If you click the **Operation Help** button when a native operation is selected, the *HP QuickTest Professional Object Model Reference* opens for the selected test object. For more information on specific native operations, see the documentation for the environment or application you are testing.

Note: If you select a native operation, the Step Generator inserts a step using **.Object** syntax. For information on using the **Object** property, see "Accessing Native Properties and Operations" on page 887.

After you select the operation for your test object, you can define the relevant argument values. For more information, see "Specifying Argument Values" on page 783.

Selecting an Object from the Repository or Application

The Select Object for Step dialog box displays the object repository tree and enables you to select an object from the object repository or from your application.



You can select any object in the object repository tree for your new step. For more information on the object repository, see Chapter 5, "Managing Test Objects in Object Repositories."

If the object that you want to use in the new step is not in the object repository, you can select an object in your application.

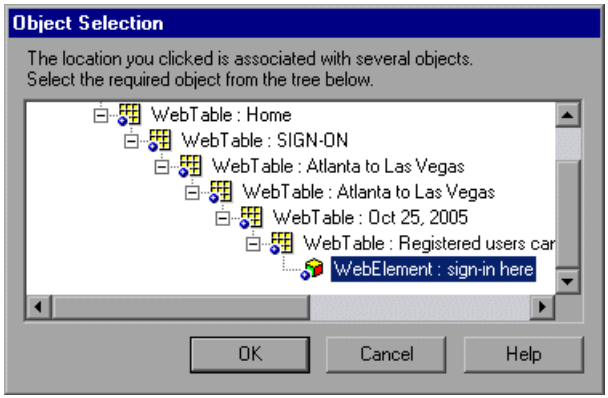
When you click **OK**, the selected object is displayed in the dialog box from which you opened the Select Object for Step dialog box.

To select an object in your application for the new step:

1 Click the pointing hand button. QuickTest is hidden, and the pointer changes to a pointing hand.

2 Use the pointing hand to click on the required object in your application. For more information about using the pointing hand feature, see "Tips for Using the Pointing Hand" on page 790.

If the location you clicked is associated with more than one object, the Object Selection dialog box opens.



3 Select the object for the new step and click **OK**. The object is displayed in the dialog box from which you opened the Select Object for Step dialog box.

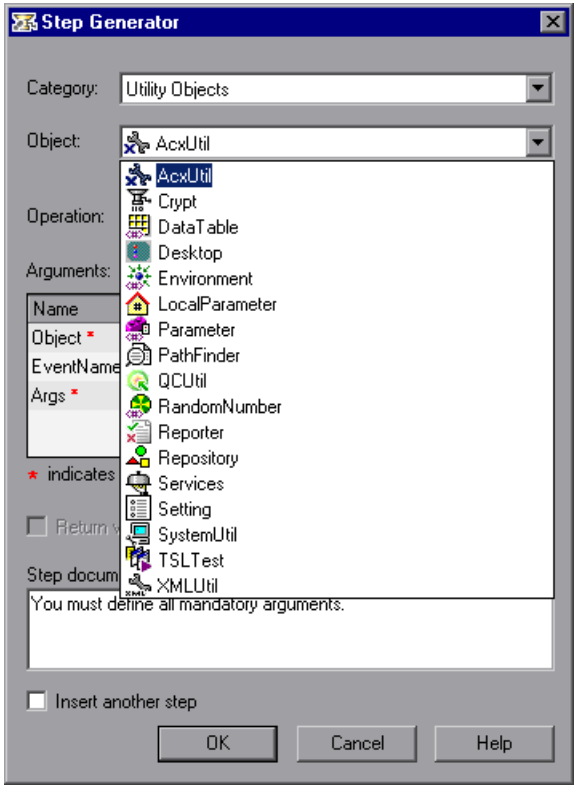
Tip: If you select an object in your application that is not in the object repository, a test object is added to the object repository when you insert the new step.

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.

Specifying a Utility Object and Operation for the Step

If you select **Utility Objects** in the **Category** box list, you can select the required Utility (reserved) object from the **Object** box list.



Tip: The above example shows the list of Utility objects that are available when you open the Step Generator from the Keyword View. When you open the Step Generator from the Expert View or a function library, the list includes a number of additional Utility objects. If you have one or more add-ins installed, the list may include additional Utility objects for those add-ins.

For more information on Utility objects, see the Utility Objects section of the *HP QuickTest Professional Object Model Reference*.

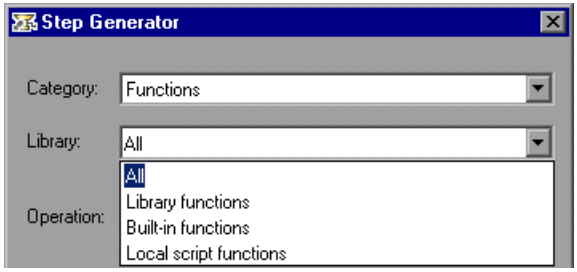
The **Operation** box displays the default operation for the selected Utility object. You can select a different operation from the **Operation** box list, which contains all the operations that are available for the selected object.

For detailed information on a Utility object operation and its syntax, you can click the **Operation Help** button to open the *HP QuickTest Professional Object Model Reference* for the selected operation.

After you select the operation for your Utility object, you can define the relevant argument values. For more information, see "Specifying Argument Values" on page 783.

Specifying a Function for the Step

If you select **Functions** in the **Category** box list, you can select one of the following options from the **Library** box list:



- **All.** Enables you to select a function from all the available functions and types.
- **Library functions.** Enables you to select a function from any function library associated with your test (for tests only). For more information on defining and using associated function libraries, see "Working with Associated Function Libraries" on page 919.
- **Built-in functions.** Enables you to select any standard VBScript function supported by QuickTest. For more information on working with VBScript, you can open the VBScript documentation from the QuickTest **Help** menu (**Help** > **QuickTest Professional Help** > **VBScript Reference**).
- **Local script functions.** Enables you to select any local function defined directly in the current action or function library.

You can select the required function from the **Operation** box list, which displays the functions available for the selected function type in alphabetical order.

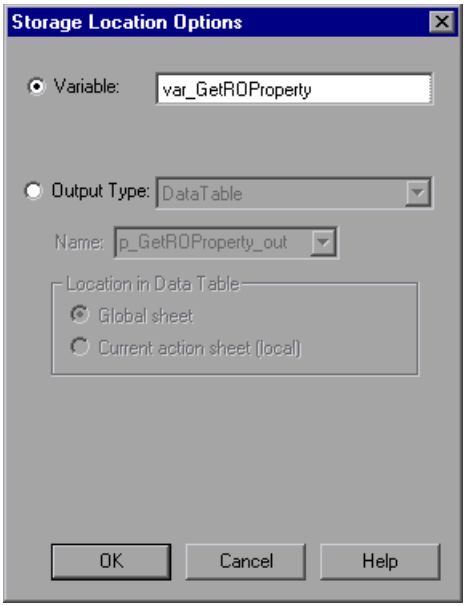
For detailed information on a selected built-in VBScript function, you can click the **Operation Help** button to open Microsoft's VBScript Reference or the *HP QuickTest Professional Object Model Reference*. This option is not available for library and local script functions.

After you select the function for the operation, you can define the relevant argument values. For more information, see "Specifying Argument Values" on page 783.

Storing Return Values and Action Output Parameter Values

The Storage Location Options dialog box enables you to specify how and where to store a return value for an operation that you have selected in the Step Generator dialog box. When you click the displayed return value and then the output storage button  , the Storage Location Options dialog box opens.

The Storage Location Options dialog box also enables you to specify how and where to store the value for an output parameter for an action. When you select an output parameter in the Parameter Values tab of the Action Call Properties dialog box and click the output storage button  in the **Store in** column, the Storage Location Options dialog box opens.



You can select one of the following options to specify where to store the value:

- **Variable.** Stores the value in a run-time variable for the duration of the run session. You can accept the default name assigned to the variable (if any) or enter a different variable name.
- **Output Type.** Stores the value in an test or action output parameter, Data Table column or environment variable. You can specify the output type and settings as for any other output value.

  When a return value or a test or action output parameter is first selected, the default output definition for the value is displayed. For more information on default output definitions for a return value, see "Understanding Default Output Definitions" on page 683.

  For more information on default output definitions for output action parameter values, see "Understanding Default Output Definitions for Action Parameter Values" on page 796.

  You can accept the default output definition by clicking **OK** or you can change the output type and/or settings.

  The options for changing the output type and settings are identical to those in the Output Options dialog box. For more information, see:

- "Outputting a Value to an Action Parameter" on page 684
- "Outputting a Value to the Data Table" on page 685
- "Outputting a Value to an Environment Variable" on page 686

Understanding Default Output Definitions for Action Parameter Values

When you select **Output Type** for an output action parameter value for a nested action:

- If at least one output action parameter is defined in the action calling the nested action, the default output type is **Test/action parameter** and the default output name is the first output parameter displayed in the Action Properties dialog box of the calling action.
- If no output action parameters are defined in the calling action, the default output type is Data Table and QuickTest creates a new Data Table output name based on the selected value. The value is created in the Global sheet of the Data Table.

When you select **Output Type** for an output action parameter value for a top-level action:

- If at least one output action parameter is defined in the test, the default output type is **Test/action parameter** and the default output name is the first output parameter displayed in the Test Properties dialog box.
- If no output action parameters are defined in the test, the default output type is **Data Table** and QuickTest creates a new Data Table output name based on the selected value. The value is created in the Global sheet of the Data Table.

Using Conditional Statements

You can control the flow of your test with conditional statements. Using conditional **If...Then...Else** statements, you can incorporate decision-making into your tests.

The **If...Then...Else** statement is used to evaluate whether a condition is true or false and, depending on the result, to specify one or more statements to run. Usually the condition is an expression that uses a comparison operator to compare one value or variable with another. The following comparison operators are available: less than <, less than or equal to <=, greater than >, greater than or equal to >=, not equal <>, and equal =.

Your **If...Then...Else** statement can be nested to as many levels as you need. It has the following syntax:

If *condition* **Then** *statements* [**Else** *elsestatements*] **End If**

Or, you can use the block form syntax:

If *condition* **Then**
    [*statements*]
[**ElseIf** *condition-n* **Then**
    [*elseifstatements*] . . .
[**Else**
    [*elsestatements*]
End If

For example:

```
'Set the focus on (activate) the Open Order dialog box
Window("Flight Reservation").Dialog("Open Order").Activate

'Insert a check mark in the Order No. check box
Window("Flight Reservation").Dialog("Open Order").WinCheckBox("Order No.").
    Set "ON"


Insert an order number in the displayed box and save the value as "OrderNo" for
'use later in the script. If the value is less than or equal to 0, generate a message
'box. (If the value is illegal and a message box is generated, end the run session
'when the user clicks OK.)
OrderNo = InputBox("Enter Order Number")

If OrderNo <= 0 Then
    Msgbox "You entered an invalid order number."
    ExitAction
End If

'Insert the saved order number value in the Order No. box
Window("Flight Reservation").Dialog("Open Order").WinEdit("OrderNumber
    Edit").Set OrderNo

'Click OK to close the Open Order dialog box
Window("Flight Reservation").Dialog("Open Order").WinButton("OK").Click

'Check if an error message opens and send a report to the test results
If Window("Flight Reservation").Dialog("Open Order").Dialog("Flight
    Reservations").Exist Then
        Reporter.ReportEvent micFail, "Check that the value of the order
            number is legal", "The order number does not exist."
        Window("Flight Reservation").Dialog("Open Order").Dialog("Flight
            Reservations").WinButton("OK").Click
Else
        Reporter.ReportEvent micPass, "Check that the value of the order
            number is legal", "The order number exists."
End If
```

The above example checks whether the application being tested will identify whether a valid order number is being entered in the Order No. box in the Open Order dialog box.

To do this, QuickTest activates the Open Order dialog box (brings it into focus), selects the Order No. check box, and opens a box in which the user inserts a value—the relevant order number—and clicks **OK**. The first conditional statement instructs QuickTest to verify that the value entered by the user is greater than zero. **If** it is not greater than zero, QuickTest opens a message box stating that the value entered is invalid. When the user clicks **OK** to close the message box, QuickTest ends the run session.

Otherwise, if the value entered is greater than zero, QuickTest inserts the above value in the Order No. box.

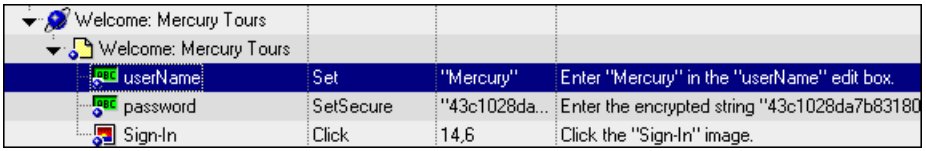
The next **If** statement instructs QuickTest to check whether the order number exists in the application and to send a report to the Test Results indicating that the step passed or failed. If the step failed because the order number was invalid, the Flight Reservations error message opens, and QuickTest clicks **OK** to close this message box before ending the run session.

Note: You can insert conditional statements in the Expert View and in the Keyword View. You can also switch between the views, as needed. For information on working with conditional steps in the Expert View, see "Using Comments, Control-Flow, and Other VBScript Statements" on page 876, and the VBScript documentation (select **Help** > **QuickTest Professional Help** > **VBScript Reference**).

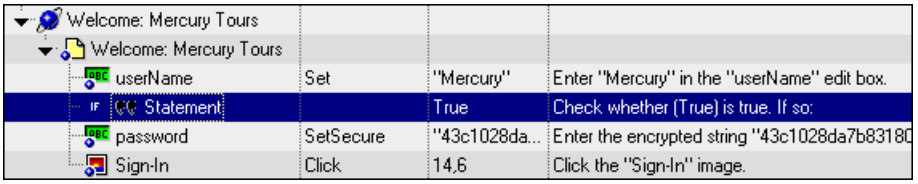
To insert a conditional statement in the Keyword View:

1 In the Keyword View, select the step that you want the conditional statement to follow.

The following example shows the userName row selected:



2 Select **Insert** > **Conditional Statement** and select **If...Then**. The new statement is added to the Keyword View below the selected step. For example:



Note: Each statement type is indicated by one of the following icons:

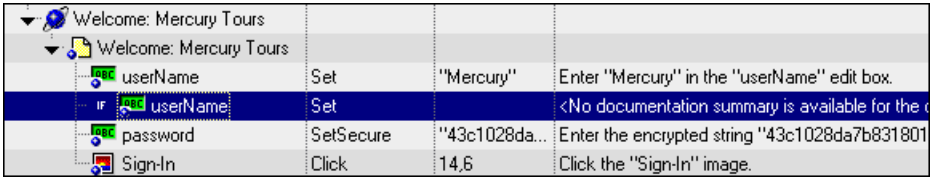
IF (**If...Then** statement)

ELSE IF (**ElseIf...Then** statement)

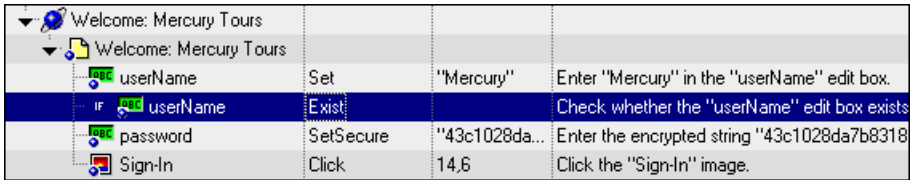
ELSE (**Else** statement)

---

3 Click in the **Item** cell for the **If** statement. Then click the down arrow and select the object on which you want to perform the conditional statement. For example:

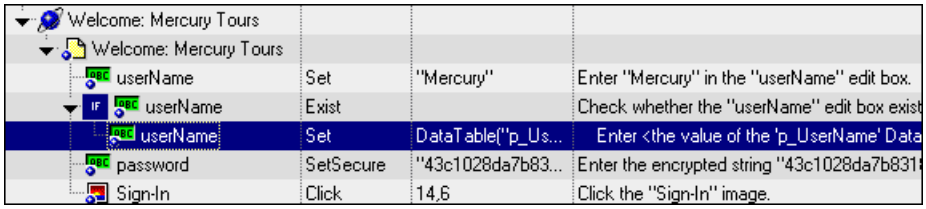


4 Click in the **Operation** cell and select the operation you want to perform. For example:



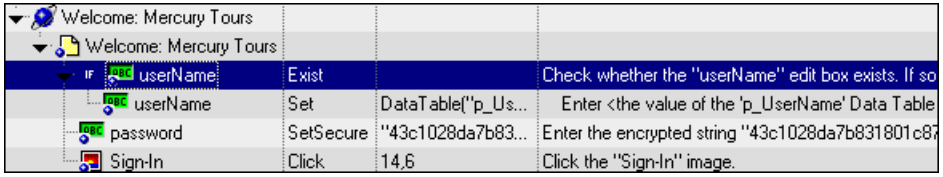
5 If needed, click in the **Value** cell and enter the required condition. (In this example, because we are using the **Exist** property, it is not necessary to add a value to the **Value** cell.)

6 Insert a **Then** statement by selecting the **If** statement step and inserting a new statement (**Insert** > **New Step**) or by recording a new step. For example:



Make sure you set the values for the new step in the **Operation** and **Value** columns.

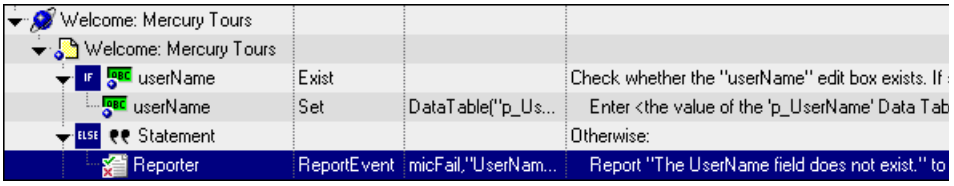
7 Delete the row immediately above the **If** statement. For example:



8 You can now complete the statement with an **Else** statement, or you can nest an additional level in your statement. To do this, select the **If** statement and select one of the following options:

To add:	Select:
an **If** statement	**Insert** > **Conditional Statement** > **If...Then**
an **ElseIf** statement	**Insert** > **Conditional Statement** > **ElseIf...Then**
an **Else** statement	**Insert** > **Conditional Statement** > **Else**

For example, the statements below check that the User Name edit box exists in the Mercury Tours site. **If** the edit box exists, **Then** a user name is entered; **Else** a message is sent to the Test Results.



The same example is displayed in the Expert View as follows:

```
If Browser("Welcome: Mercury").Page("Welcome: Mercury").
    WebEdit("userName").Exist Then
        Browser("Welcome: Mercury").Page("Welcome: Mercury").
            WebEdit("userName").Set DataTable ("p_UserName", dtGlobalSheet)
Else
    Reporter.ReportEvent micFail, "UserName Check", "The User Name field
        does not exist."
End If
```

9 After you have finished creating the conditional statement, use the **Insert Step After Block** option if you want to insert a step outside of the conditional statement block. For more information, see “Adding a Standard Step After a Conditional or Loop Block” on page 409.

Using Loop Statements

You can control the flow of your test with loop statements. Using loop statements, you can run a group of steps repeatedly, either while or until a condition is True. You can also use loop statements to repeat a group of steps a specific number of times.

The following loop statements are available in the Keyword View:

- **While...Wend.** Performs a series of statements as long as a specified condition is True.
- **For...Next.** Uses a counter to perform a group of statements a specified number of times.
- **Do...While.** Performs a series of statements indefinitely, as long as a specified condition is True.
- **Do...Until.** Performs a series of statements indefinitely, until a specified condition becomes True.

Note: For more information on loop statements, see the VBScript documentation (select **Help** > **QuickTest Professional Help** > **VBScript Reference**).

To insert a loop statement in the Keyword View:

1 Select the step that you want the loop statement to follow.

2 Select **Insert** > **Loop Statement** and select the statement type that you want to insert from the sub-menus. The new statement is added to the Keyword View below the selected step.

Each statement type is indicated by one of the following icons:

Icon	Type
	While...Wend statement
	For...Next statement
	Do...While statement
	Do...Until statement

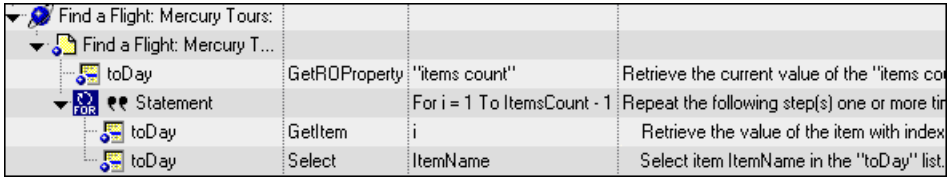
3 In the **Value** column, enter the required condition, for example:
For i = 0 to ItemsCount - 1

4 To complete the loop statement, you can:

- Select the loop statement step and record a new step to add it to your loop statement.
- Select the loop statement step and select **Insert** > **New Step** or press F8 to insert a new step into your loop statement.

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

The following example counts the number of items in a list and then selects them one by one. After each of the items has been selected, the test continues.



The same example is displayed in the Expert View as follows:

```
itemsCount = Browser("Welcome: Mercury").Page("Find a Flight:").
    WebList("toDay").GetROProperty ("items count")
For i = 1 To ItemsCount-1
    ItemName = Browser("Welcome: Mercury").Page("Find a Flight:").
        WebList("toDay").GetItem (i)
    Browser("Welcome: Mercury").Page("Find a Flight:").WebList("toDay").
        Select ItemName
Next
```

5 After you have finished creating the loop statement, use the **Insert Step After Block** option if you want to insert a step outside of the loop statement block. For more information, see "Adding a Standard Step After a Conditional or Loop Block" on page 409.

Generating With Statements for Your Test

You can instruct QuickTest to automatically generate **With** statements when you record a test or to generate **With** statements for any existing action. You can also remove **With** statements from an action.

Note: Using **With** statements in your test has no effect on the run session itself, only on the way your test appears in the Expert View. Generating **With** statements for your test does not affect the Keyword View in any way.

Understanding With Statements

With statements make your script (in the Expert View) more concise and easier to read by grouping consecutive statements with the same parent hierarchy.

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

```
With object
    statement
    statement
    statement
End With
```

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:

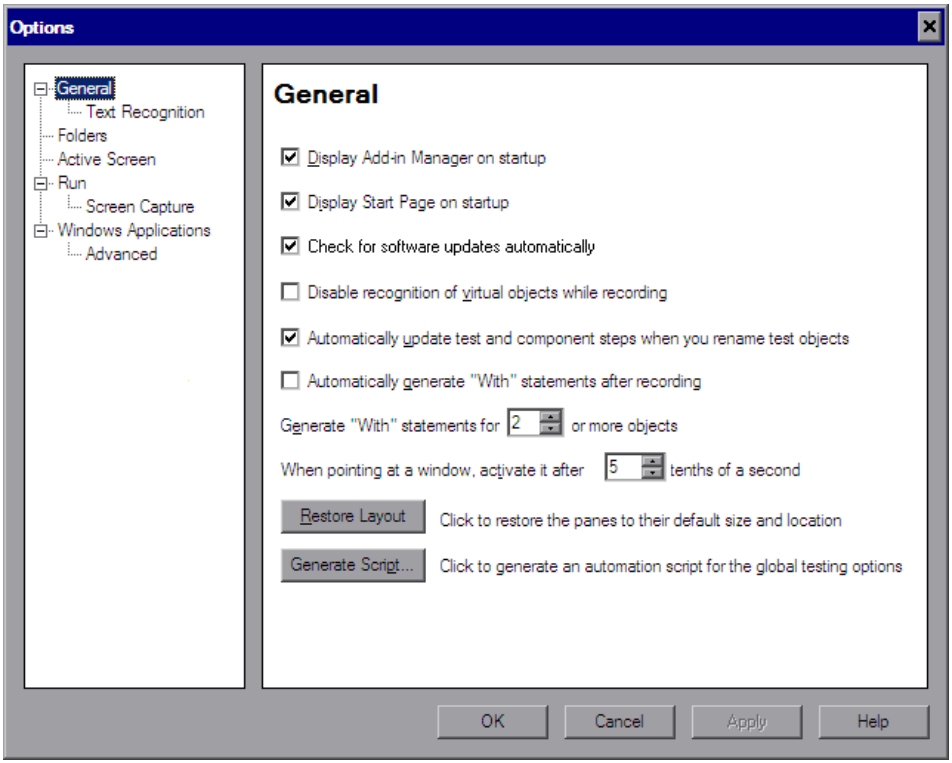
```
With Window("Flight Reservation")
    .WinComboBox("Fly From:").Select "London"
    .WinComboBox("Fly To:").Select "Los Angeles"
    .WinButton("FLIGHT").Click
    With .Dialog("Flights Table")
        .WinList("From").Select "19097   LON "
        .WinButton("OK").Click
    End With 'Dialog("Flights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

Automatically Generating With Statements

You can instruct QuickTest to automatically generate **With** statements for the steps you record. When you select this option, statements are displayed in their normal format while recording. When you stop recording, the statements in all actions recorded during the current recording session are automatically converted to the **With** format.

To generate With statements automatically when you record:

1 Select **Tools** > **Options** or click the **Options** toolbar button. The Options dialog box opens.



2 In the **General** pane, select **Automatically generate "With" statements after recording**.

3 Enter the minimum number of consecutive, identical objects for which you want to apply the **With** statement in the **Generate "With" statements for __ or more objects** box. The default is 2.

Note: This setting is used when you use the **Apply "With" to Script** option (see "Generating With Statements for Existing Actions" on page 809) as well as for the **Automatically generate "With" statements after recording** option.

For example, if you only want to generate a **With** statement when you have three or more consecutive statements based on the same object, enter 3.

4 Begin recording your test. While recording, statements are recorded normally. When you stop recording, the statements in all actions recorded during the current recording session are automatically converted to the **With** format.

Generating With Statements for Existing Actions

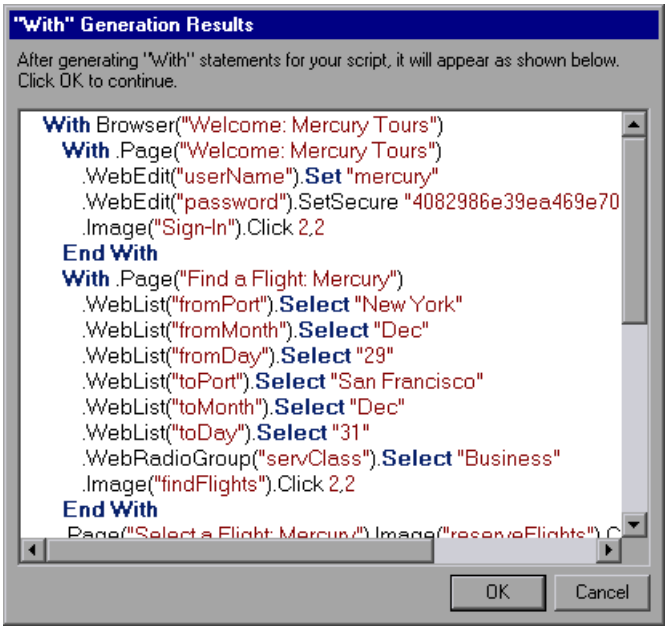
You can instruct QuickTest to generate **With** statements for any action displayed in the Expert View, and to enable IntelliSense within existing **With** statements.

To generate With statements for existing actions:

1 Confirm that the proper number is set for the **Generate "With" statements for __ or more objects** in the General pane of the Options dialog box. (The default is 2.)

2 Display the action for which you want to generate **With** statements.

3 From the Expert View, select **Edit** > **Advanced** > **Apply "With" to Script**. The "With" Generation Results window opens.



Each **With** statement contains only one object

4 To confirm the generated results, click **OK**. The **With** statements are applied to the action.

Tips:

- You can search for text strings in the Generation Results window by pressing CTRL+F. For more information on the Find dialog box, see "Finding Text Strings" on page 847.
- If you type a **With** statement (as opposed to creating it using the procedure described above), select **Edit** > **Advanced** > **Apply "With" to Script** to enable IntelliSense within the **With** statement.

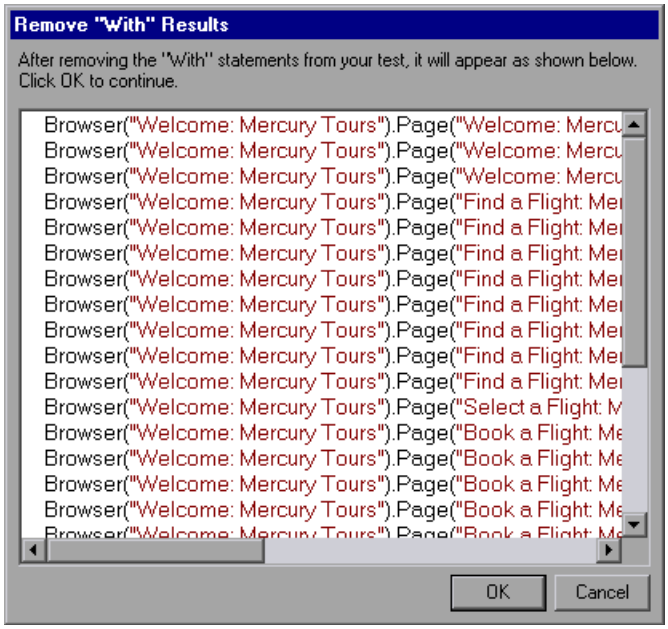
Removing With Statements from an Action

You can remove all the **With** statements in an action displayed in the Expert View.

To remove With statements from an action:

1 Display the action for which you want to remove **With** statements.

2 From the Expert View, select **Edit** > **Advanced** > **Remove "With" Statements**. The Remove "With" Results window opens.



3 To confirm the results, click **OK**. The **With** statements are replaced with the standard statement format.

Generating Messages

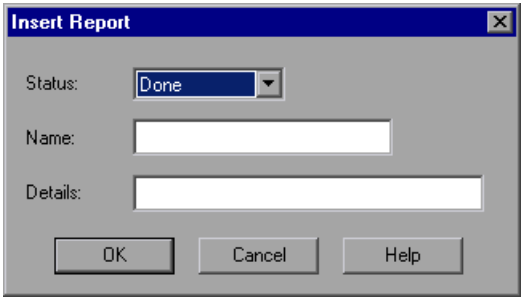
You can generate messages in your test that are displayed in the Test Results window. You can also choose to display messages on screen while running your test.

Sending Messages to the Test Results

You can define a message that QuickTest sends to your test results. For example, suppose you want to check that a password edit box exists in the Mercury Tours site. If the edit box exists, then a password is entered. Otherwise, QuickTest sends a message to the test results indicating that the object is absent.

To send a message to your test results:

1 In the Keyword View, select a step and select **Insert** > **Report** or right-click a step and select **Insert Step** > **Report**. The Insert Report dialog box opens.



2 Select the status that will result from this step from the **Status** list.

Status	Description
Passed	Causes this step to pass. Sends the specified message to the report.
Failed	Causes this step (and therefore the test itself) to fail. Sends the specified message to the report.
Done	Sends a message to the report without affecting the pass/fail status of the step.
Warning	Sends a warning status for the step, but does not cause the test to stop running, and does not affect its pass/fail status.

3 In the **Name** box, type a name for the step, for example, Password edit box.

4 In the **Details** box, type a detailed description of this step to send to your test results, for example, Password edit box does not exist.

5 Click **OK**. A report step is inserted into the Keyword View and a **Reporter.ReportEvent** statement is inserted into your script in the Expert View. For example:

```
Reporter.ReportEvent micFail, "Password edit box", "Password edit box does not exist"
```

In this example, micFail indicates the status of the report (failed), Password edit box is the report name, and Password edit box does not exist is the report message.

For more information on test results, see Chapter 33, "Viewing Run Session Results."

Note: After you add a report step, you can modify it in the Keyword View by right-clicking the step and choosing **Report Properties**, or by modifying any of the arguments in the **Value** column. (You can also modify the **Reporter.ReportEvent** statement directly in the Expert View.)

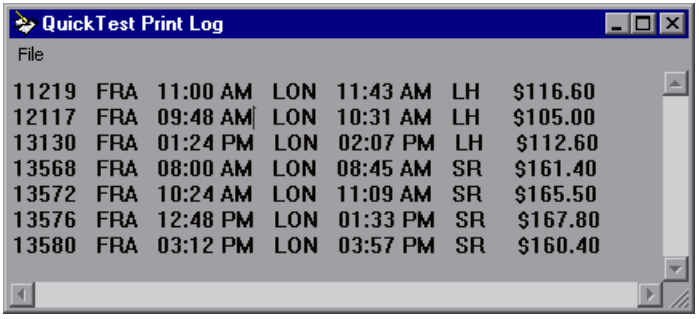
Displaying Messages During the Run Session

In addition to sending messages to the Test Results, you can generate messages in the following ways:

- Use the **MessageBox** VBScript function in your test to display information during the run session. The run session pauses until the message box is closed. For more information, see the VBScript documentation from the QuickTest **Help** menu (**Help** > **QuickTest Professional Help** > **VBScript Reference**).
- Use the **Print** Utility statement in your test to display information in the QuickTest Print Log window while still continuing the run session. For example, the following example iterates all the items in the **Flight Table** dialog (in the sample Flight application) and uses the **Print** Utility statement to print the content of each item to the QuickTest Print Log window.

```
Set FlightsList = Window("Flight Reservation").Dialog("Flights Table").
    WinList("From")
For i = 1 to FlightsList.GetItemsCount
        Print FlightsList.GetItem(i - 1)
Next
```

The QuickTest Print Log window remains open throughout the run session, until you close it.



Adding Comments

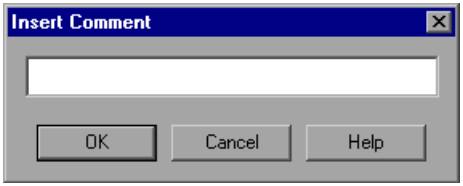
While editing your test, you can add comments in the Keyword View or in the Expert View. You can also add comments to function libraries. A comment is an explanatory remark in a program. When you run a test, QuickTest does not process comments. You can use comments to explain sections of your tests to improve readability and to make them easier to update. You can add comments directly to the Keyword View or the Expert View, or you can use the Insert Comment dialog box. You can also modify comments at any time directly in the Keyword View or the Expert View, or using the Comment Properties dialog box.

To add a comment in the Keyword View:

1 If the **Comment** column is not visible, right-click any column header and select **Comment**.

2 Add a comment in one of the following ways:

- To add a comment on the same line as a step, select the step and type your comment in the **Comment** column.
- To add a comment on a separate line, select a step and select **Insert** > **Comment**, or right-click a step and select **Insert Step** > **Comment**. The Insert Comment dialog box opens. Type a comment and click **OK**. A comment statement is added to your test.



In the Keyword View, the  icon indicates a comment.

To add a comment in the Expert View or a function library:

Type an apostrophe (') and then type your comment. You can add a comment at the end of a line or at the beginning of a separate line.

To modify a comment:

- In the Keyword View, you can modify the comment text directly in the **Comment** column, or you can right-click any column in the step and select **Comment Properties** to open the Comment Properties dialog box (which is similar to the Insert Comment dialog box).
- In the Expert View, you can overwrite any existing comment.

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

Synchronizing Your Test

When you run a test, your application may not always respond with the same speed. For example, it might take a few seconds:

- for a progress bar to reach 100%
- for a status message to appear
- for a button to become enabled
- for a window or pop-up message to open

You can handle these anticipated timing problems by synchronizing your test to ensure that QuickTest waits until your application is ready before performing a certain step.

There are several options that you can use to synchronize your test:

- You can insert a **synchronization point**, which instructs QuickTest to pause the test until an object property achieves the value you specify. When you insert a synchronization point into your test, QuickTest generates a **WaitProperty** statement in the Expert View.

- You can insert **Exist** or **Wait** statements that instruct QuickTest to wait until an object exists or to wait a specified amount of time before continuing the test.
- You can modify the default amount of time that QuickTest waits for a Web page to load.
- When working with tests, you can increase the default timeout settings for a test to instruct QuickTest to allow more time for objects to appear.

Creating Synchronization Points

If you do not want QuickTest to perform a step or checkpoint until an object in your application achieves a certain status, you should insert a synchronization point to instruct QuickTest to pause the test until the object property achieves the value you specify (or until a specified timeout is exceeded).

For example, suppose you record a test on a flight reservation application. You insert an order, and then you want to modify the order. When you click the **Insert Order** button, a progress bar is displayed and all other buttons are disabled until the progress bar reaches 100%. Once the progress bar reaches 100%, you record a click on the **Update Order** button.

Without a synchronization point, QuickTest may try to click the **Update Order** button too soon during a test run (if the progress bar takes longer than the test's object synchronization timeout), and the test will fail.

You can insert a synchronization point that instructs QuickTest to wait until the **Update Order** button's **enabled** property equals 1.

Tip: QuickTest must be able to identify the specified object to perform a synchronization point. To instruct QuickTest to wait for an object to open or appear, use an **Exist** or **Wait** statement. For more information, see "Adding Exist and Wait Statements" on page 821.

To insert a synchronization point:

1 Start a recording session.

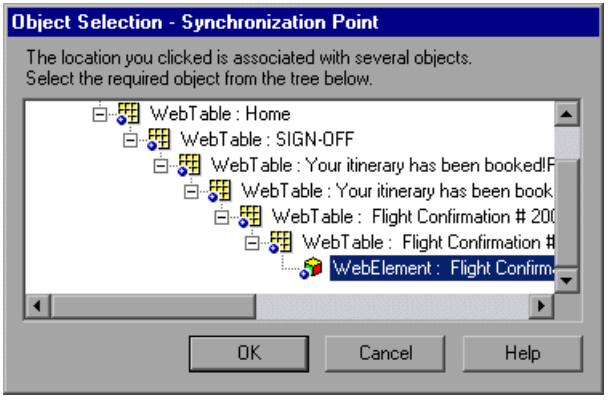
2 Display the screen or page in your application that contains the object for which you want to insert a synchronization point.

3 In QuickTest, select **Insert > Synchronization Point**. The pointer changes to a pointing hand. For more information about using the pointing hand feature, see "Tips for Using the Pointing Hand" on page 820.

4 Click the object in your application for which you want to insert a synchronization point.

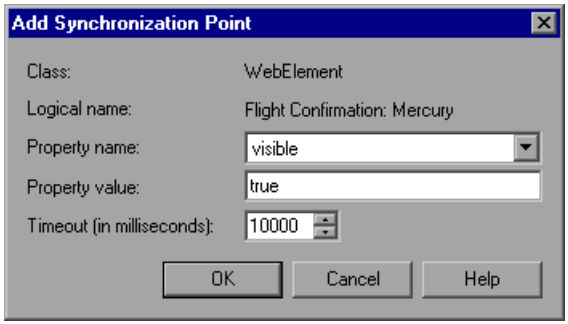
Note: It does not matter what property values the object has at the time that you insert the synchronization point.

If the location you click is associated with more than one object in your application, the Object Selection - Synchronization Point dialog box opens.



Select the object for which you want to insert a synchronization point, and click **OK**.

The Add Synchronization Point dialog box opens.



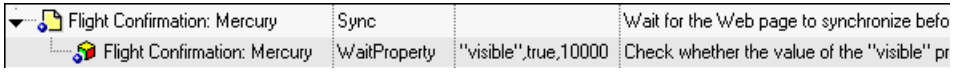
5 The **Property name** list contains the identification properties associated with the object. Select the property name you want to use for the synchronization point.

6 Enter the property value for which QuickTest should wait before continuing to the next step in the test.

7 Enter the synchronization point timeout (in milliseconds) after which QuickTest should continue to the next step, even if the specified property value was not achieved.

8 Click **OK**. A **WaitProperty** step is added to your test.

Because the WaitProperty step is a method of the selected object, it is displayed in the Keyword View with the icon for the selected object. For example, if you insert a synchronization point for the **Update Order** button, it may look something like this:



In the Expert View, this appears as:

```
Browser("Welcome: Mercury Tours").Page("Flight Confirmation: Mercury").Sync
Browser("Welcome: Mercury Tours").Page("Flight Confirmation: Mercury").
    WebElement("Flight Confirmation #").WaitProperty "visible", true, 10000
```

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

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.

Adding Exist and Wait Statements

You can enter **Exist** and/or **Wait** statements to instruct QuickTest to wait for a window to open or an object to appear. Exist statements return a boolean value indicating whether or not an object currently exists. Wait statements instruct QuickTest to wait a specified amount of time before proceeding to the next step. You can combine these statements within a loop to instruct QuickTest to wait until the object exists before continuing with the test.

For example, the following statements instruct QuickTest to wait up to 20 seconds for the Flights Table dialog box to open.

```
blnDone=Window("Flight Reservation").Dialog("Flights Table").Exist
counter=1
While Not blnDone
    Wait (2)
    blnDone=Window("Flight Reservation").Dialog("Flights Table").Exist
    counter=counter+1
    If counter=10 then
        blnDone=True
    End if
Wend
```

For more information on **While**, **Exist**, and **Wait** statements, see the *HP QuickTest Professional Object Model Reference*.

Modifying Timeout Values

If you find that, in general, the amount of time QuickTest waits for objects to appear or for a browser to navigate to a specified page is insufficient, you can increase the default object synchronization timeout values for your test and the browser navigation timeout values for your test.

Alternatively, if you insert synchronization points and **Exist** and/or **Wait** statements for the specific areas in your test where you want QuickTest to wait a longer time for an event to occur, you may want to decrease the default timeouts for the rest of your test.

- When working with tests, to modify the maximum amount of time that QuickTest waits for an object to appear, change the **Object Synchronization Timeout** in the **File** > **Settings** > **Run** pane. For more information, see "Defining Run Settings for Your Test" on page 1270.
- To modify the amount of time that QuickTest waits for a Web page to load, change the **Browser Navigation Timeout** in the **File** > **Settings** > **Web** pane. For more information, see the *HP QuickTest Professional Add-ins Guide*.

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

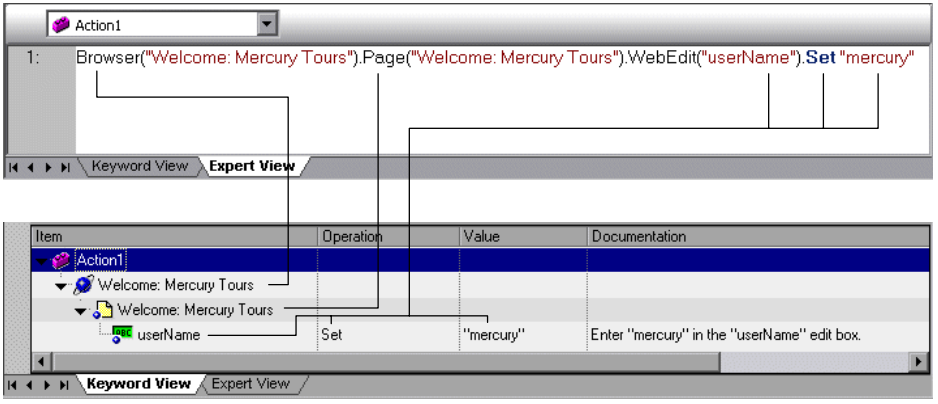
If you prefer to work with VBScript statements, you can choose to work with your tests in the Expert View, as an alternative to using the Keyword View. You can move between the two views as you wish, by selecting the Expert View or Keyword View tab at the bottom of the Test pane in the QuickTest window.

Working in the Expert View

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

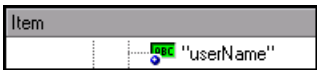
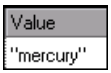
- In the Keyword View, QuickTest displays information about each step and shows the object hierarchy in an icon-based table. For more information, see Chapter 14, "Working with the Keyword View."
- In the Expert View, QuickTest displays each step as a VBScript line or statement. In object-based steps,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

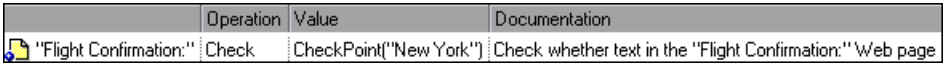
Understanding Checkpoint and Output Statements

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:

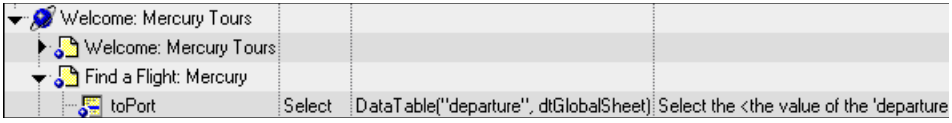
```
Browser("Welcome: Mercury").Page("Find a Flight:").WebList("toPort").
    Select "San Francisco"
```

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.

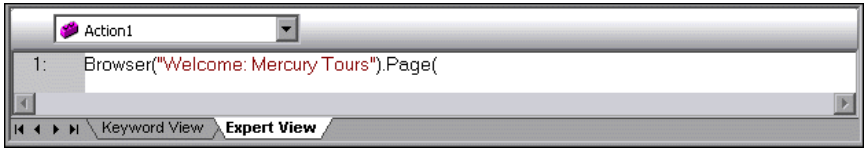
Tip: If you press CTRL+SPACE, QuickTest displays a list of the relevant test objects, operations, properties, collections, VBScript functions, user-defined functions, VBScript constants, and utility objects that you can add. This list is displayed even if you typed an object that has not yet been added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

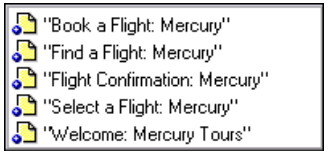
1 Confirm that the **Statement completion** option is selected (**Tools** > **View Options** > **General** tab).

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 **(**



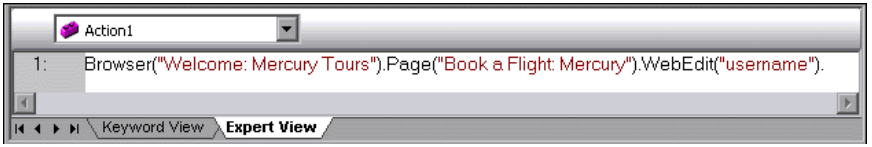
If there is only one object of this type in the object repository, QuickTest automatically enters its name in quotes after the open parenthesis. If more than one object of this type exists in the object repository, QuickTest displays them in a list.



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:

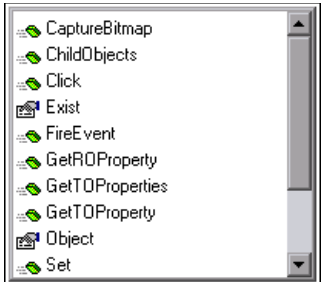
➤ If you are working in the Expert View, type a period (.) after the object on which you want to perform the operation.



➤ 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")
```

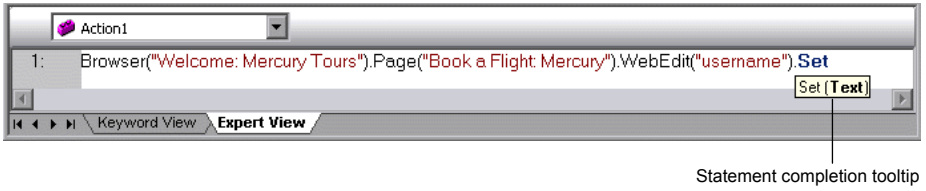
5 Type a period (.) after the object description, for example ("username"). QuickTest displays a list of the available operations and properties for the object.



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.

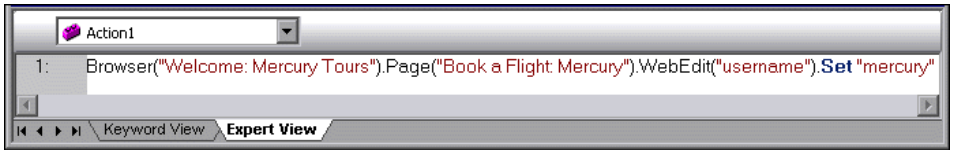
6 Double-click an operation or property in the list or use the arrow keys to choose an operation or property and press ENTER. QuickTest inserts the operation or property into the statement. If the operation or property contains arguments, QuickTest displays the syntax of the operation or property in 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.

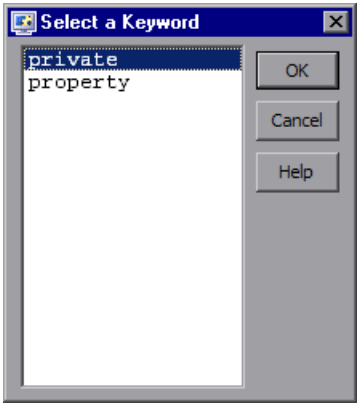
Automatically Completing VBScript Syntax

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

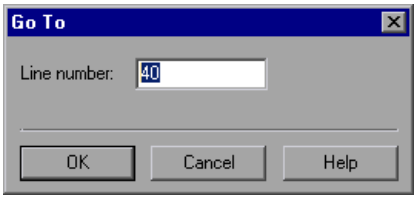
You can use the Go To dialog box to navigate to a specific line in an action or in a function library.

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.



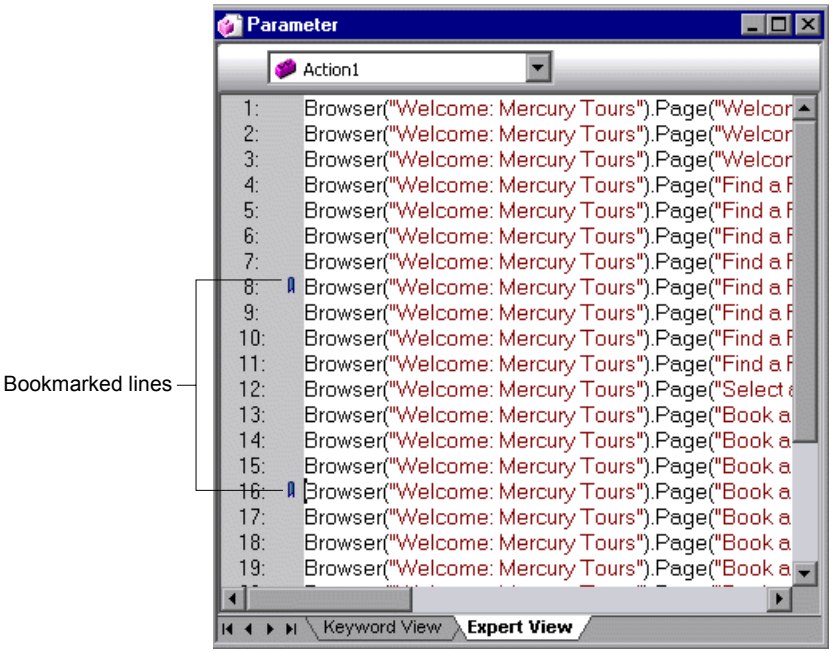
3 Enter the line to which you want to navigate in the **Line number** box and click **OK**. The cursor moves to the beginning of the line you specify.

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.

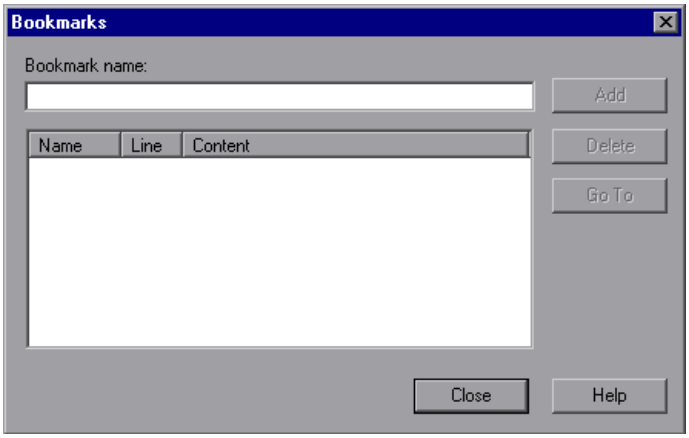


To set bookmarks:

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:

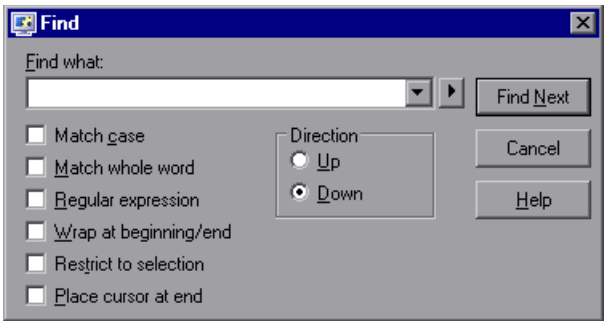
- Click the **Find** button.
- Select **Edit** > **Find**.

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.

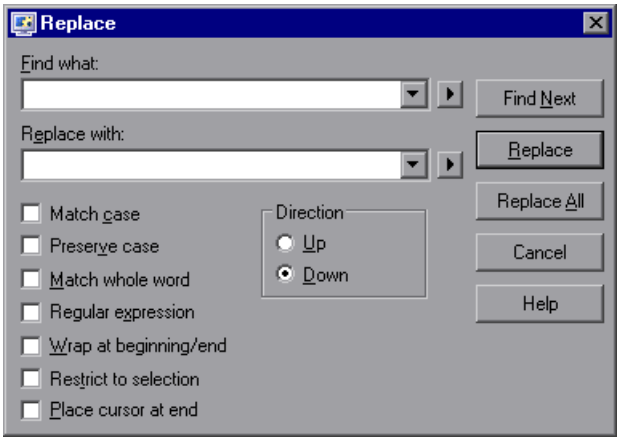
To replace a text string:

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.

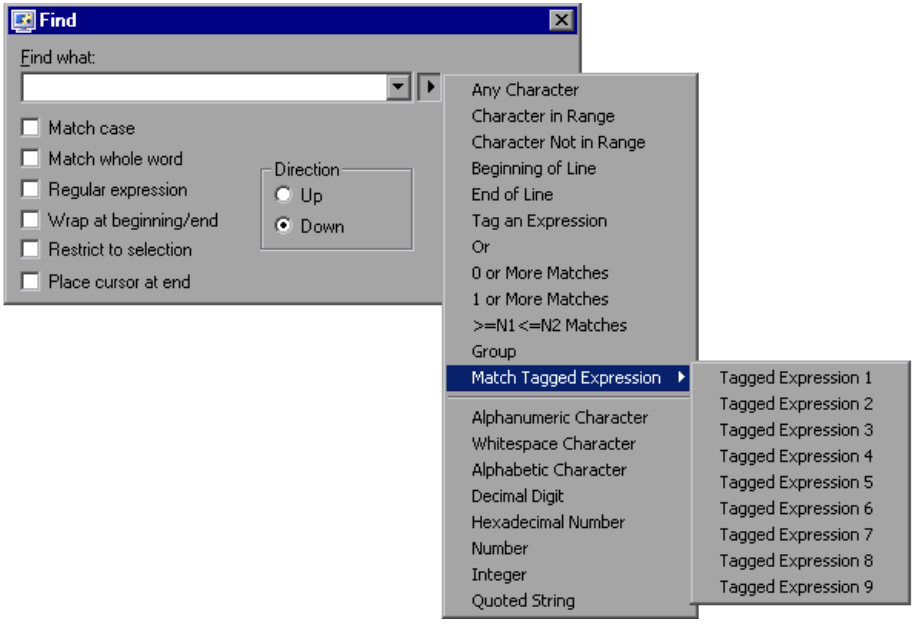
6 Click **Find Next** to highlight the next occurrence of the specified text string in the current action or dialog box, or in the active function library.

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

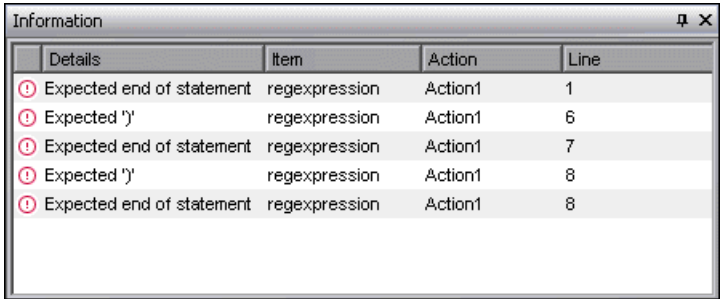
When you select the Keyword View tab from the Expert View, QuickTest attempts to display the updated information in the Keyword View. If a new or updated VBScript statement contains syntax errors, the text **Error** flashes in red at the right of the status bar, and an error message is displayed in the 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 *condition*2 **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.

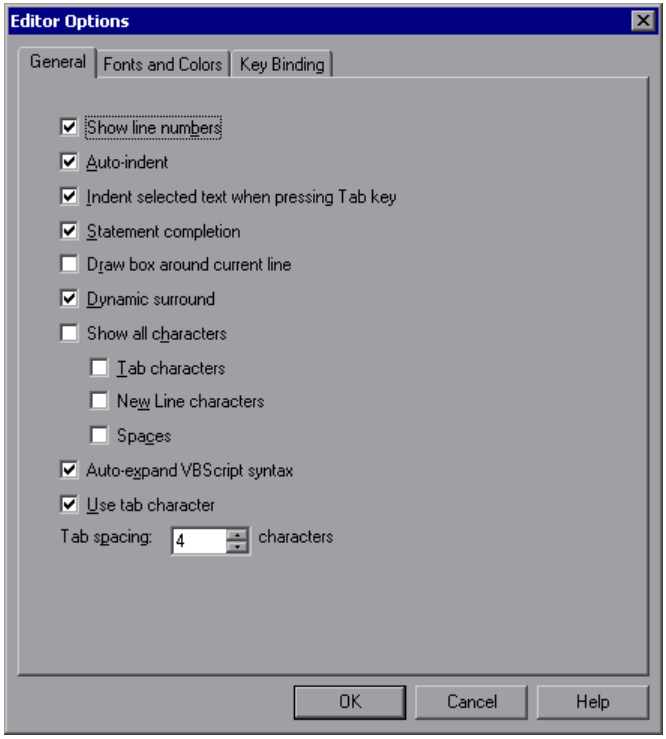
For more information on using the Expert View, see Chapter 29, "Working in the Expert View and Function Library Windows." For more information on working with function libraries, see Chapter 31, "Working with User-Defined Functions and Function Libraries."

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.

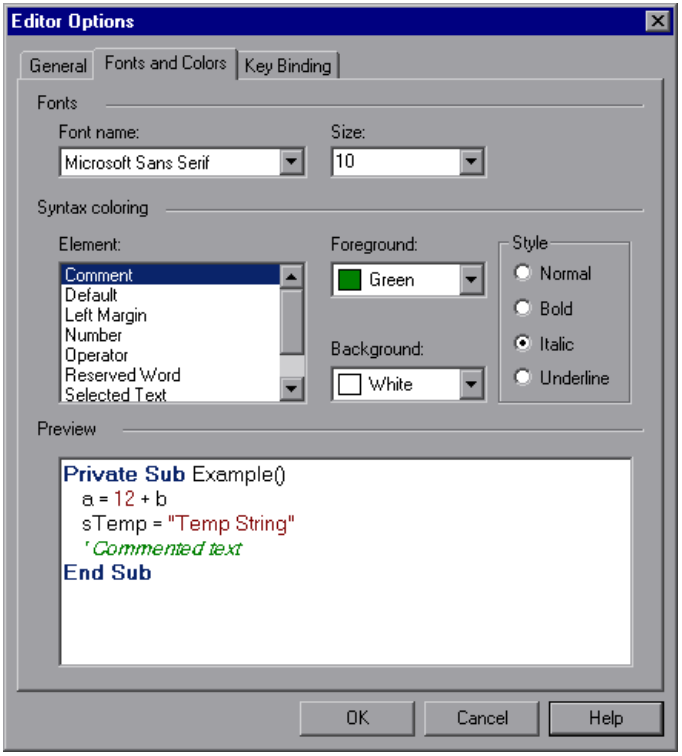
Customizing Element Appearance

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

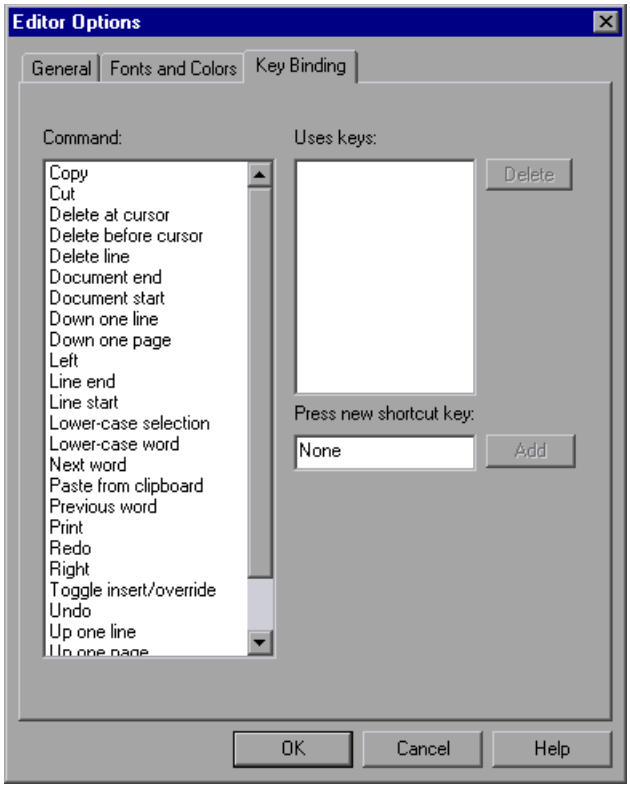
You can personalize the default keyboard shortcuts you use for editing. QuickTest includes keyboard shortcuts that let you move the cursor, delete characters, and cut, copy, or paste information to and from the Clipboard. You can replace these shortcuts with your preferred shortcuts. For example, you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

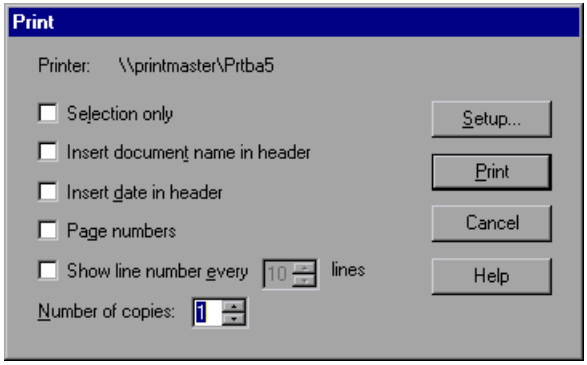
Note: During a debug session, all documents are read-only and cannot be edited. To edit a document during a debug session, you must first stop the debug session.

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.

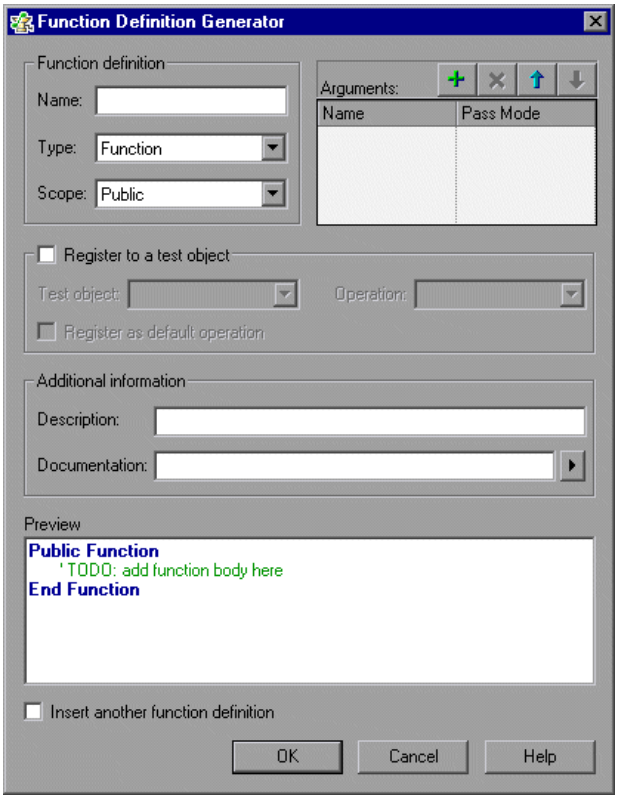
Opening the Function Definition Generator

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.

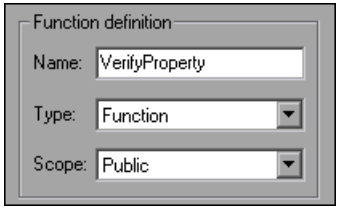
2 Select **Insert** > **Function Definition Generator** or click the **Function Definition Generator** button. The Function Definition Generator opens.



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.

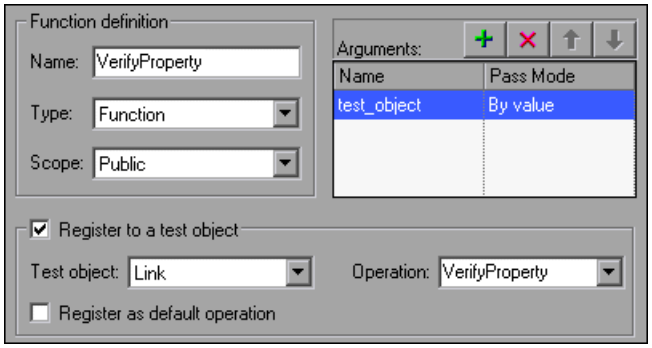
To define this function as the default function, you define the value of the fourth argument of the RegisterUserFunc statement as **True**. For example:

```
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.

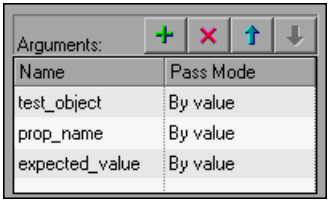
For example, if you were to define the VerifyProperty operation as the default operation for the Link test object, the value **True** would be defined as the fourth argument of the RegisterUserFunc statement, and the syntax would appear as follows:

```
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).

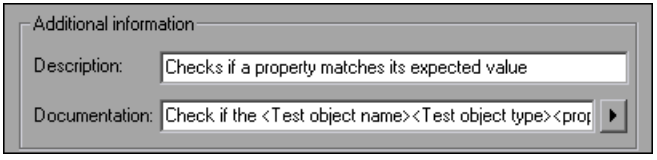
Documenting the Function

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.

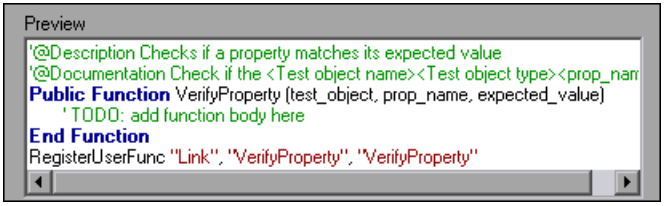
2 In the **Documentation** box, enter the text to be displayed in the **Step documentation** box in the Step Generator in the Keyword View and in the **Documentation** column of the Keyword View. You can use arguments in the **Documentation** text by clicking  and selecting the relevant argument.

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."

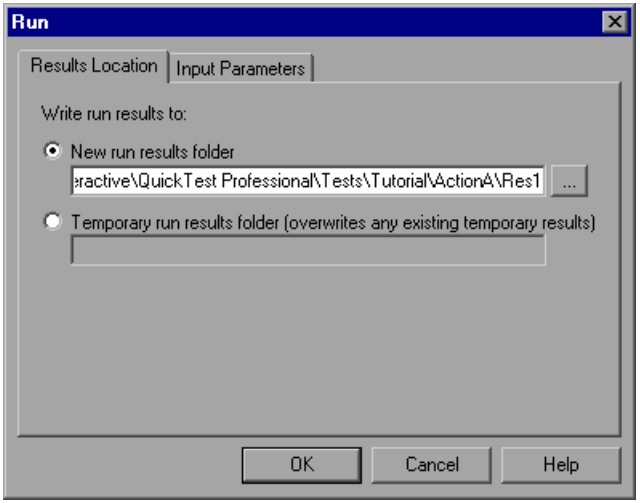
The Test Results summary displays a note indicating that the test was run using the **Run from Step** or **Run Current Action** option.

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.

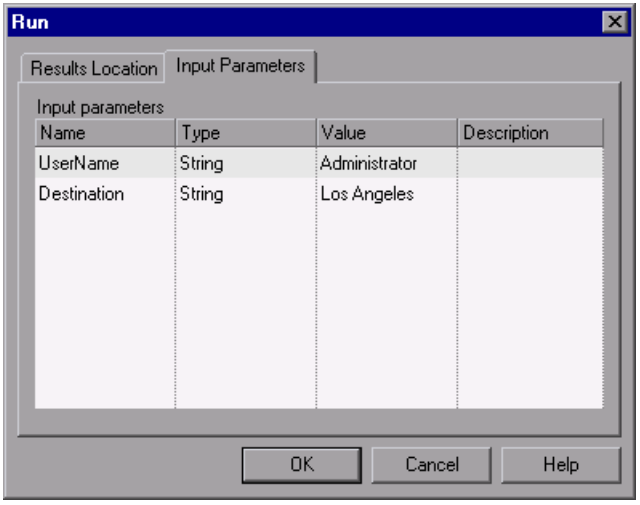
- **Temporary run results folder.** Saves the run results in a temporary folder. This option overwrites any results previously saved in this folder.

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.

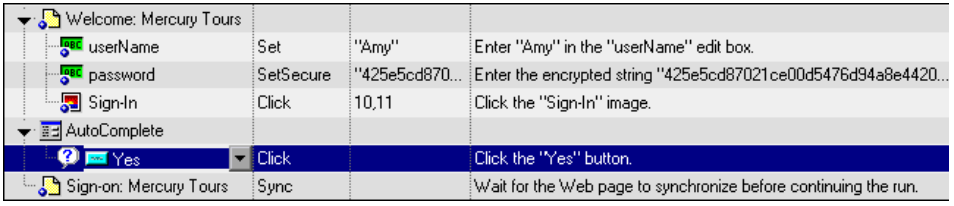
However, if, during a run session, QuickTest cannot find the object from the optional step in the object repository (for example, if the object name was modified in the test but not in the object repository, or if the object was removed from the object repository), an error message is displayed listing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

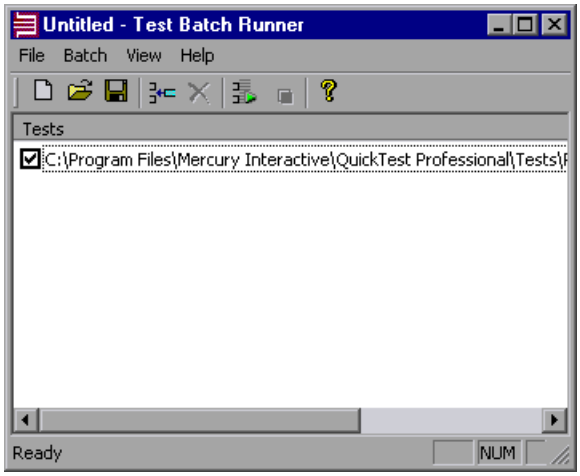
Notes:

- 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.

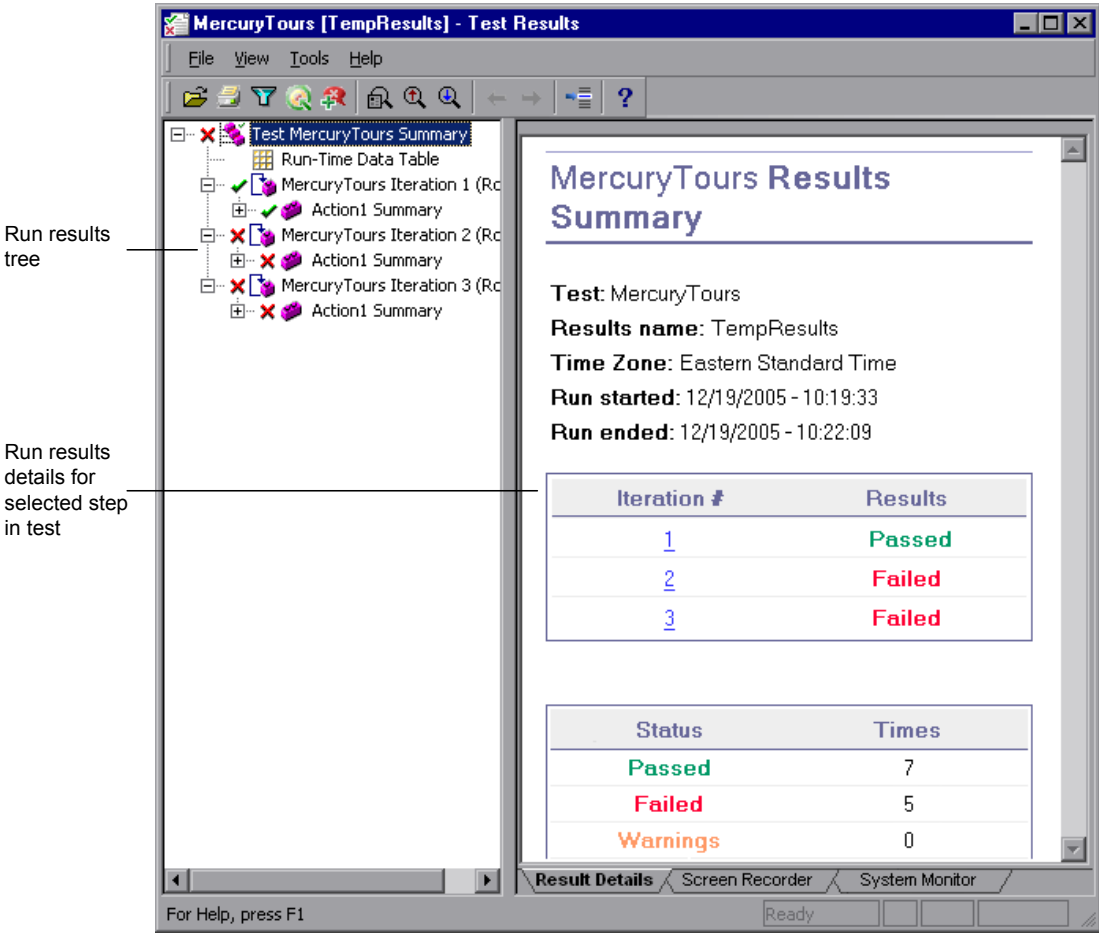
The Test Results Window

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.”

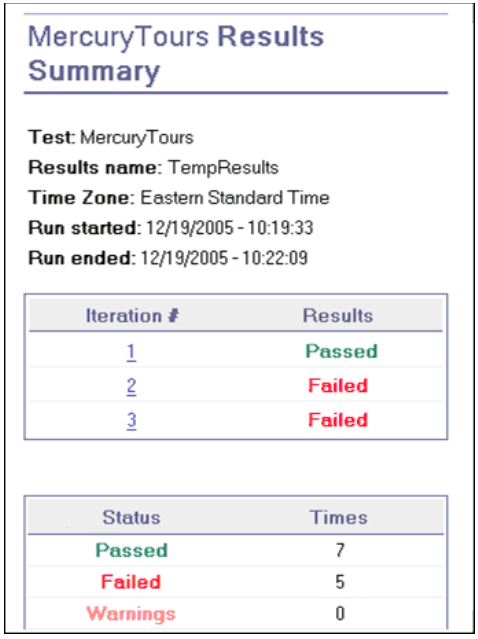
You can collapse or expand a branch in the run results tree to change the level of detail that the tree displays.

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.

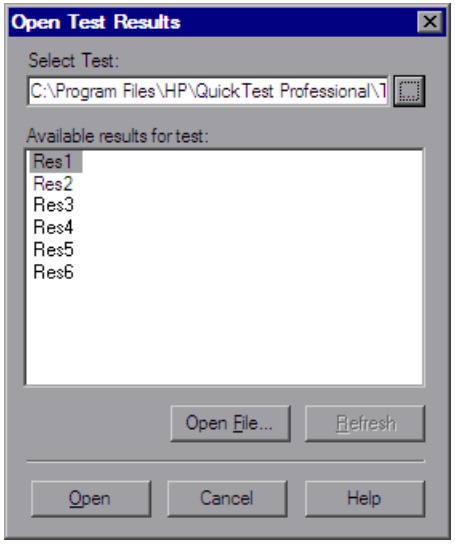
To view the saved results of the current test or other tests:

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.

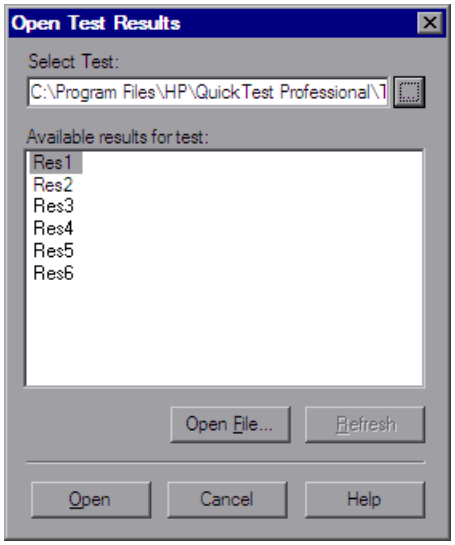
If your QuickTest test is stored in Quality Center, the results are stored in the test folder in Quality Center. You cannot change the location of the run session results.

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**.

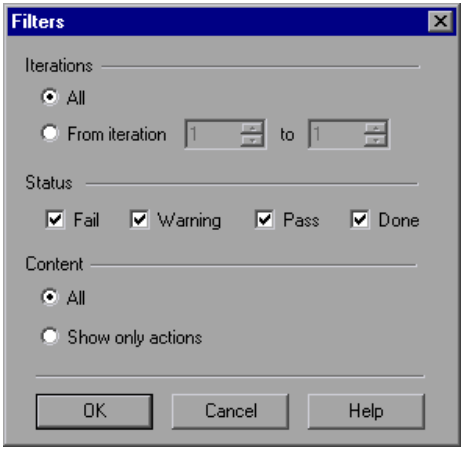
3 The QuickTest window is activated and the step is highlighted.

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:

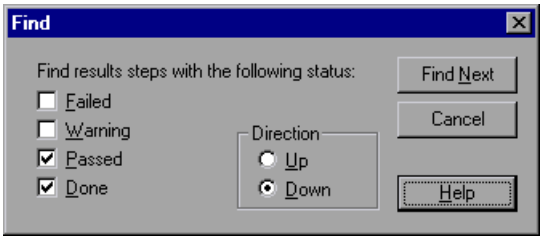
Option	Description
Iterations	(This option is available only for tests.) ➤ **All.** Displays test results from all iterations. ➤ **From iteration X to Y.** Displays the test results from the specified range of test iterations.

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.

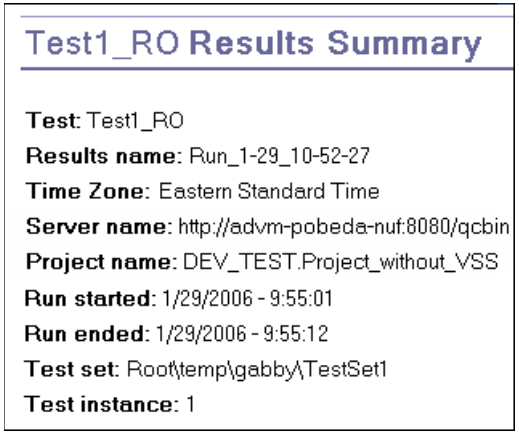
Viewing Results of Tests Run from Quality Center

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

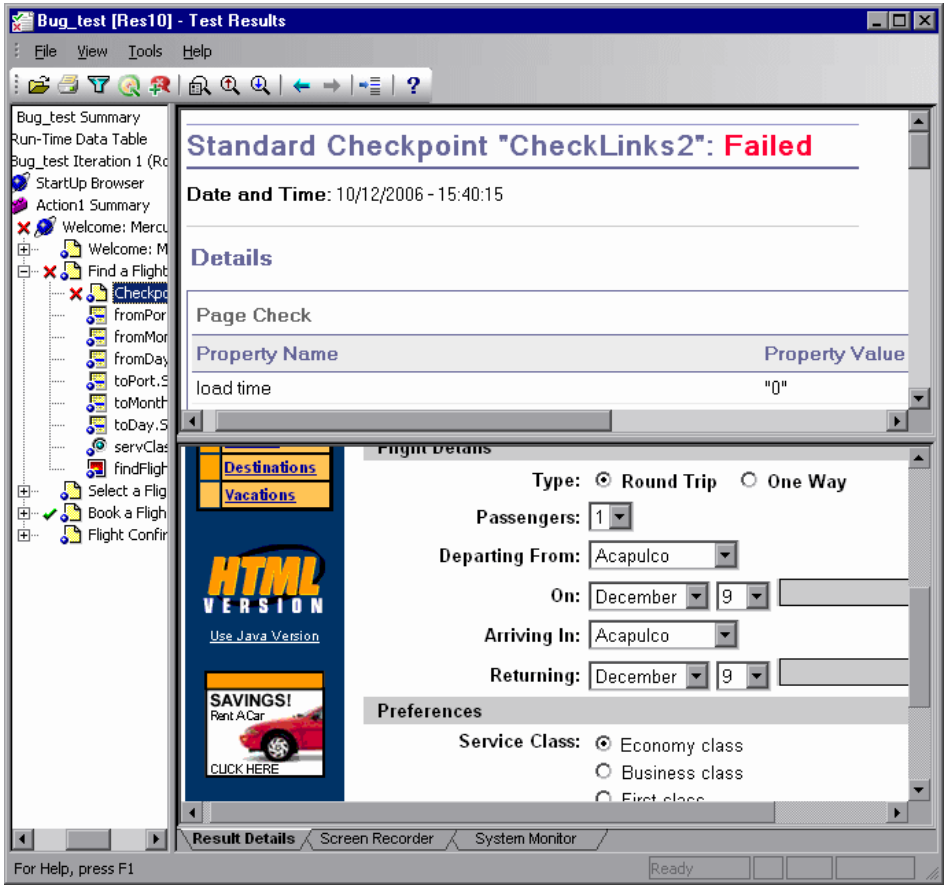
QuickTest Professional can capture still images and movies of your application during a run session. These captured files can be viewed in the Test Results window. The **Result Details** and **Screen Recorder** tabs in the right pane enable you to view either still images and text details, or a movie 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.



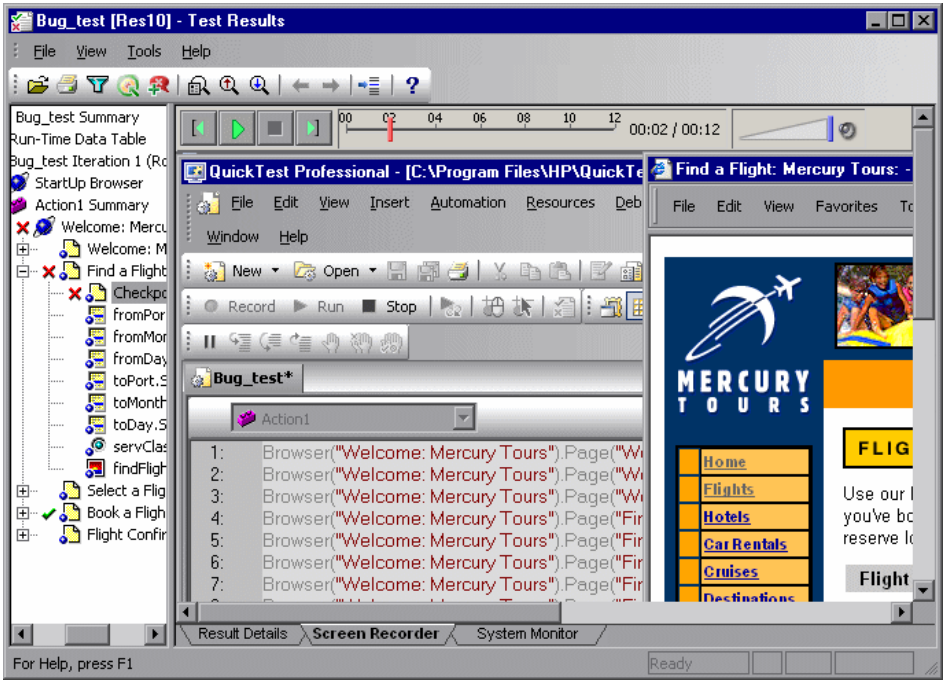
If the highlighted step does not contain an error, the right pane contains the result details with no screen capture.

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.

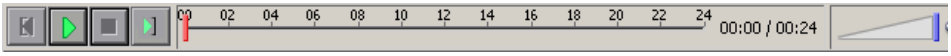
Viewing Movies of Your Run Session

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.

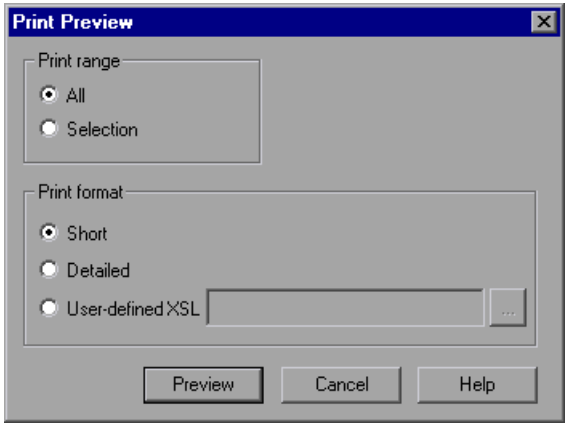
Previewing Test Results

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.

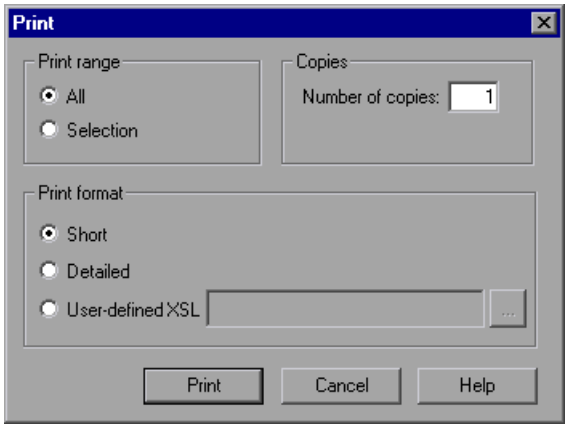
Tip: If some of the information is cut off in the preview, for example, if checkpoint names are too long to fit in the display, click the **Page Setup** button in the Print Preview window and change the page orientation from **Portrait** to **Landscape**.

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

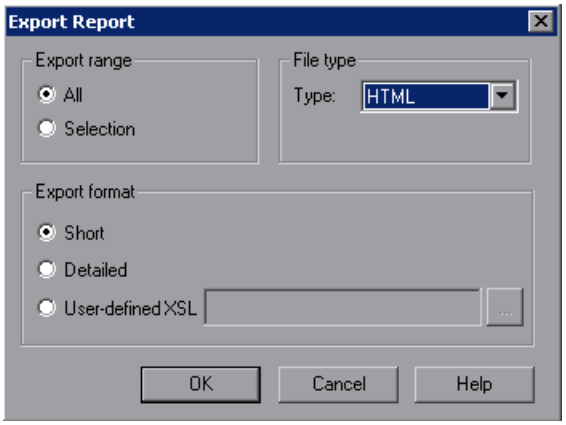
You can export the test result details to an HTML, PDF, or DOC file. This enables you to view the test results even if the QuickTest environment is unavailable. For example, you can send the file containing the test results in an e-mail to a third-party who does not have QuickTest installed. You can 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.

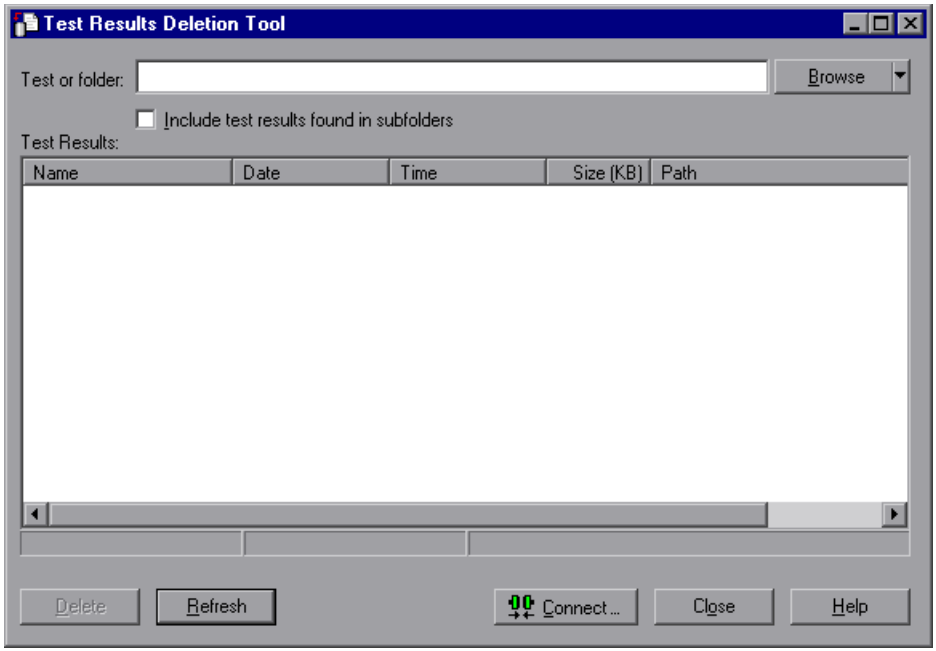
Deleting Results Using the Test Results Deletion Tool

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.

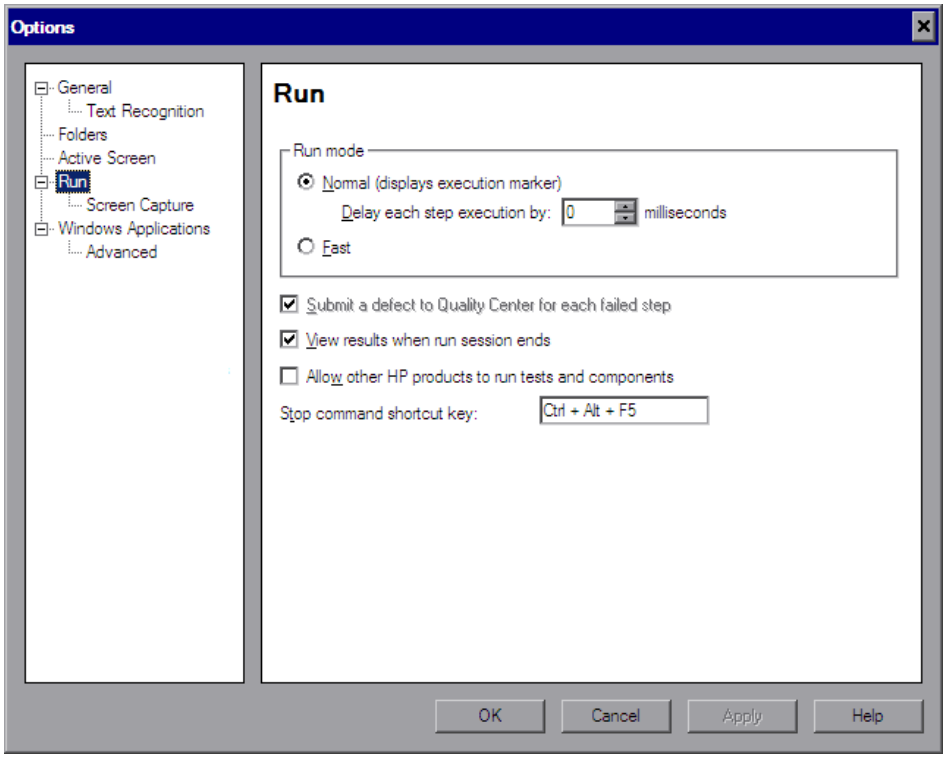
Automatically Submitting Defects to a Quality Center Project

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.

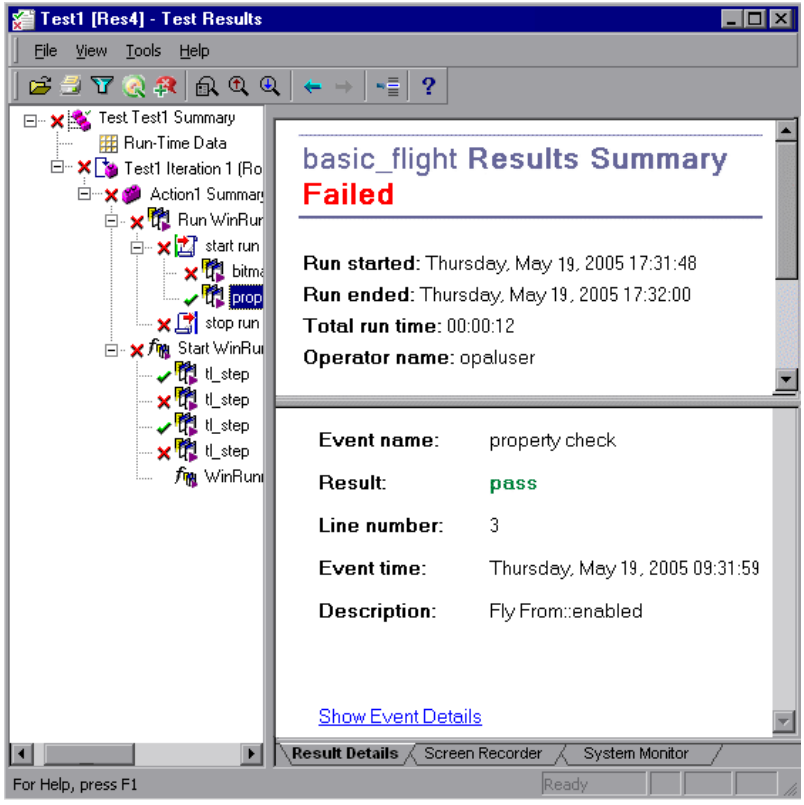
A sample of the information that is submitted to Quality Center for each defect is shown below:

```
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.

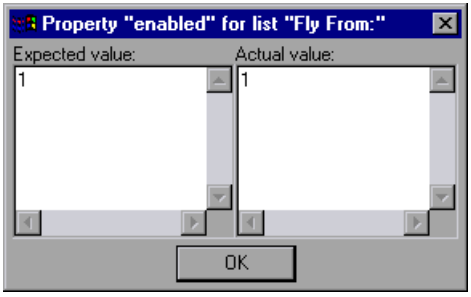
The left pane in the QuickTest test results include a node for each WinRunner event that would normally be included in the WinRunner results. When you select a node corresponding to a WinRunner test event or function call, the right pane displays a summary of the called WinRunner test or function and 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.

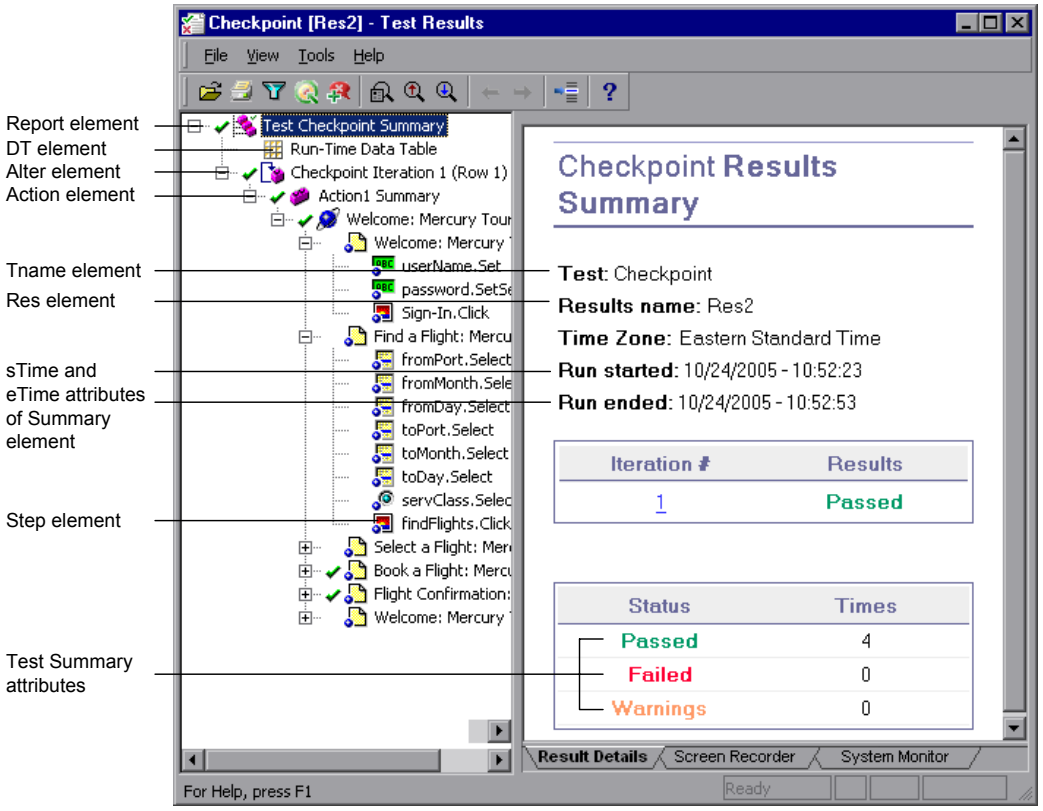
For more information on running WinRunner tests and functions from QuickTest, see Chapter 57, “Working with WinRunner.”

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.

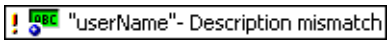
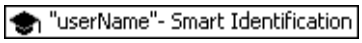
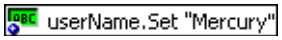
This chapter includes:

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.

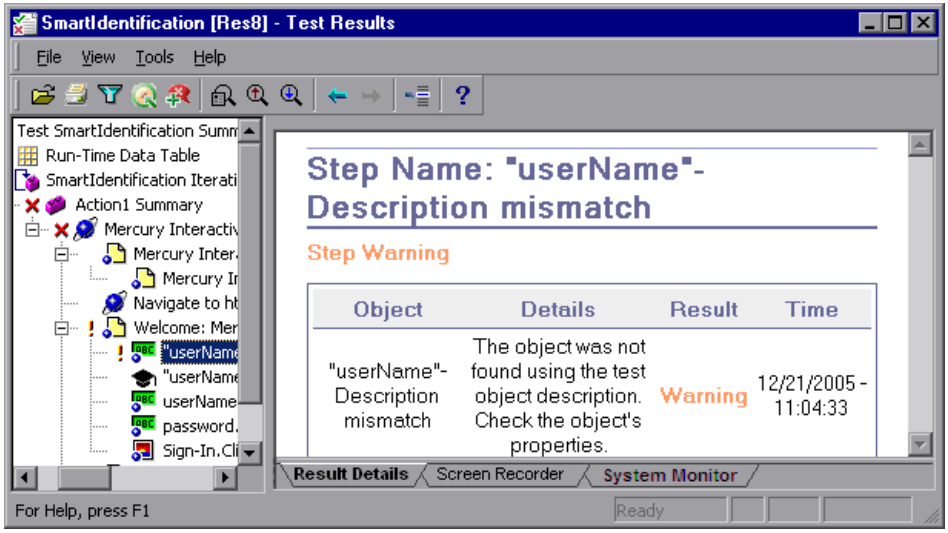
Smart Identification—No Object Matches the Learned Description

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.”

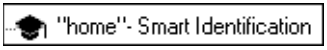
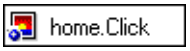
The image below shows the results for a test in which Smart Identification was used to identify the userName WebEdit object after one of the learned description property values changed.



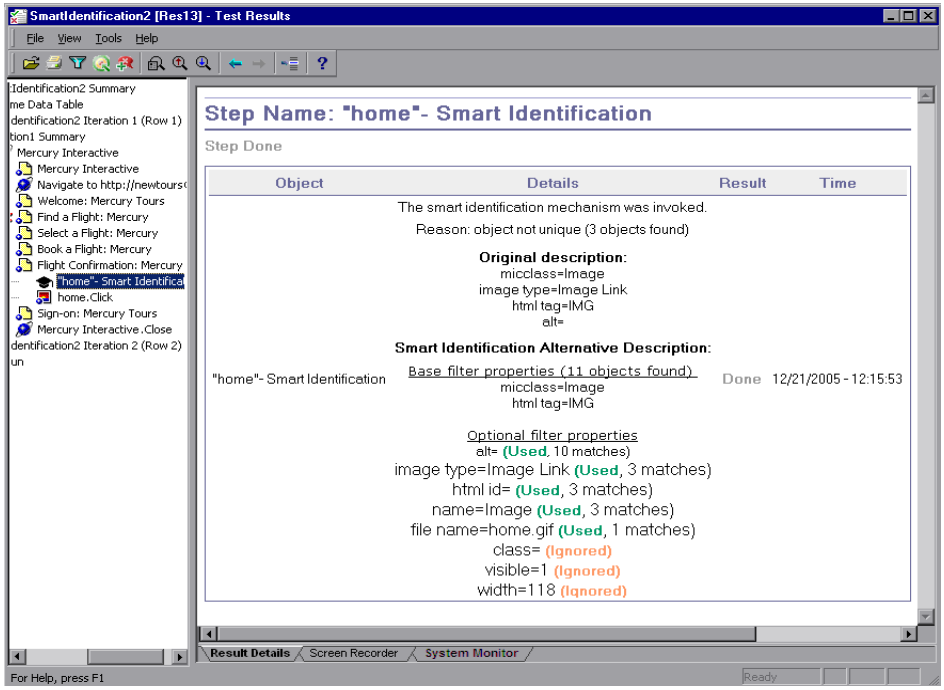
Smart Identification—Multiple Objects Match the Learned Description

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

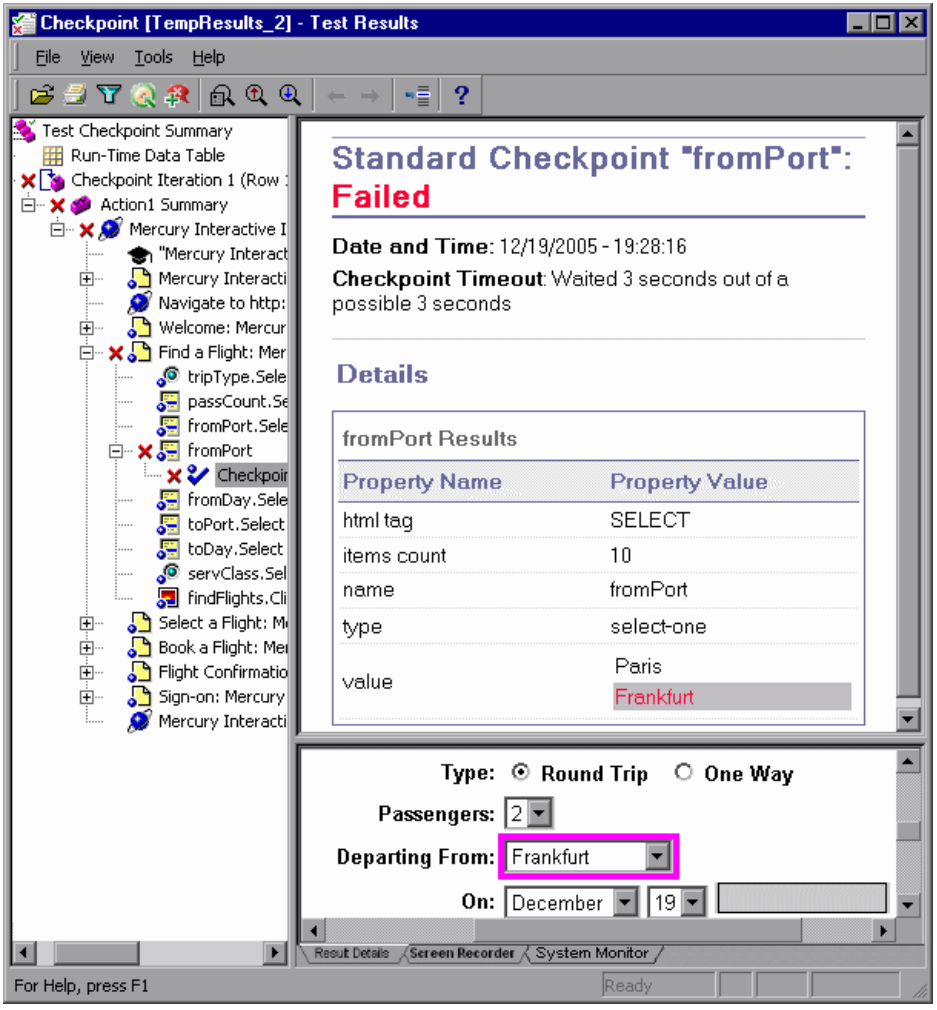
3 Select **File > Exit** to close the Test Results window.

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**.

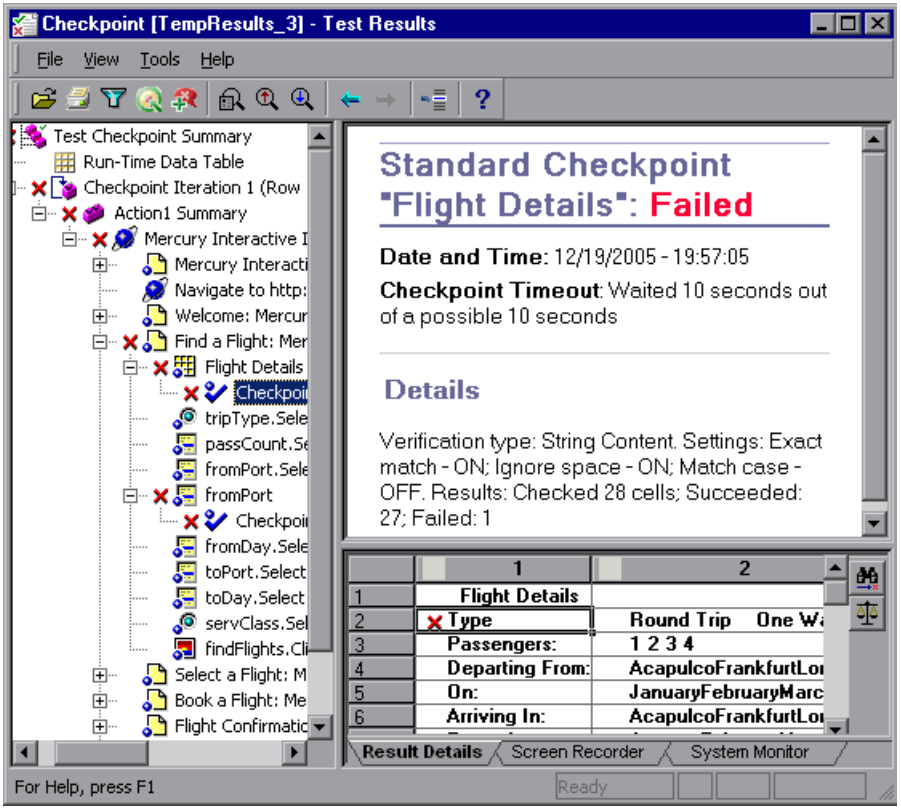
Analyzing Table and Database Checkpoint Results

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.

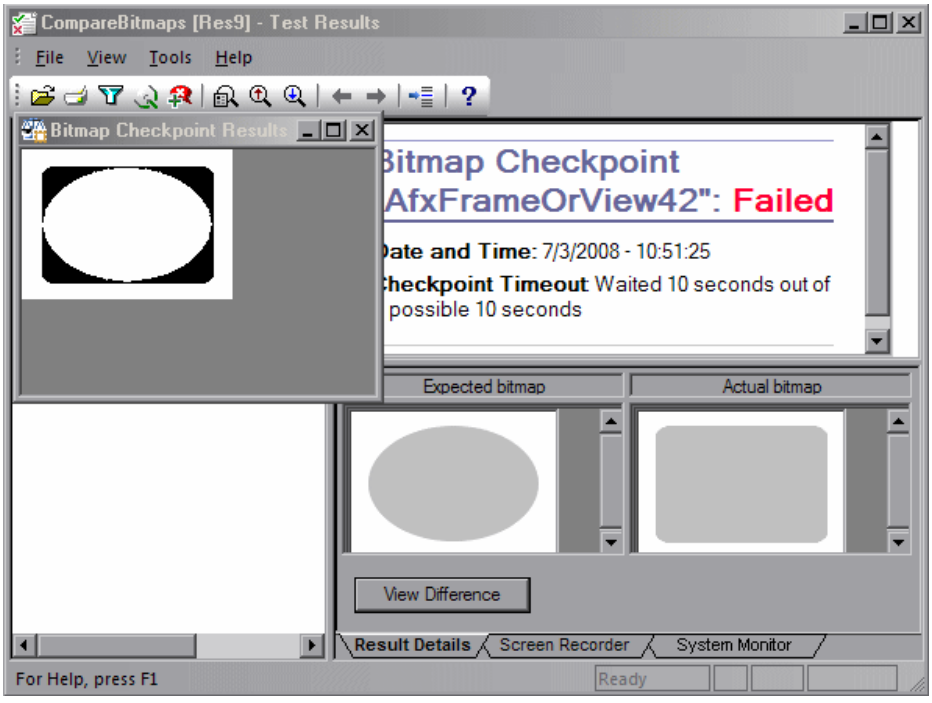
Analyzing Bitmap Checkpoint Results

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

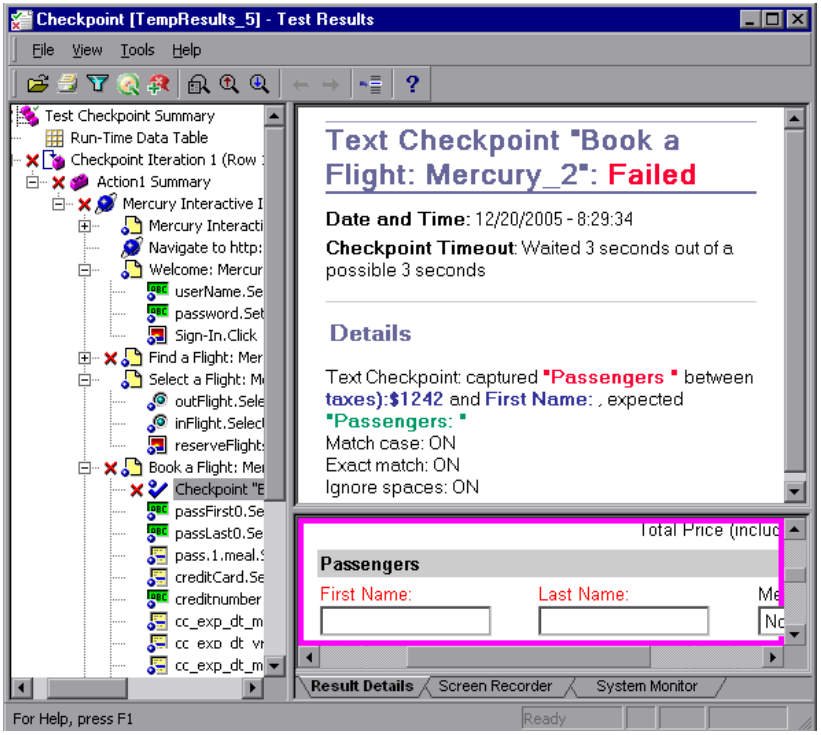
- If the checkpoint is defined to compare only a specific area of the bitmap, the test results display the actual and expected bitmaps with the selected area highlighted.
- When the dimensions of the actual and expected bitmaps are different, QuickTest fails the checkpoint without comparing the bitmaps. 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.

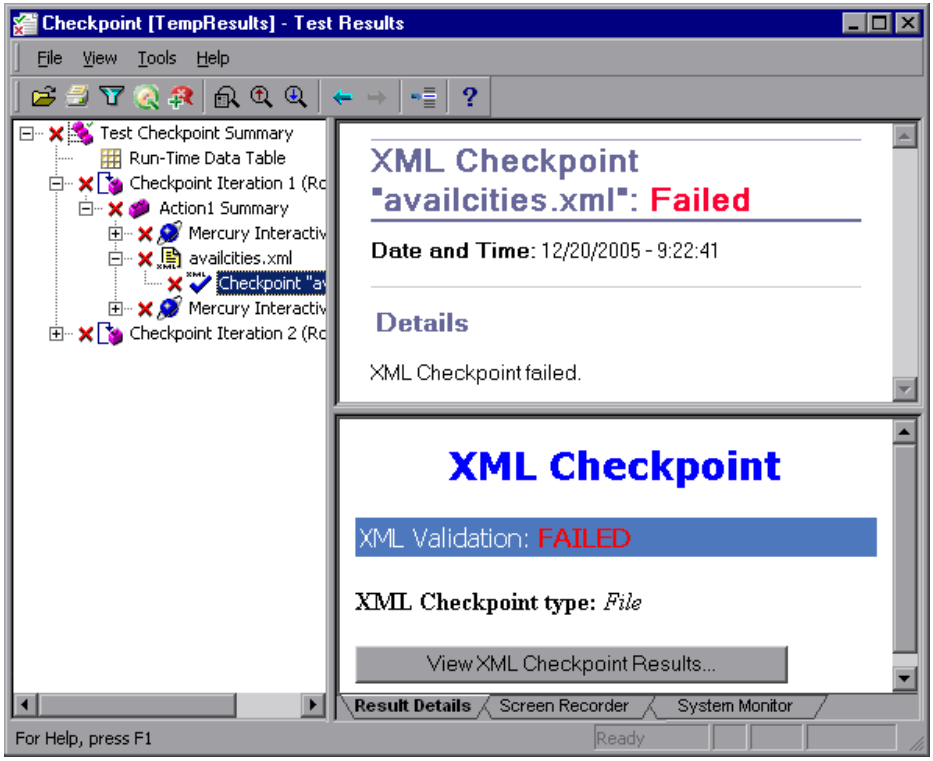


The top pane in the Result Details tab displays the checkpoint step results, including its status (**Passed** or **Failed**), the date and time the checkpoint was run and the portion of the checkpoint timeout interval that was used (if any). It also shows the expected text and actual text that was checked, 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

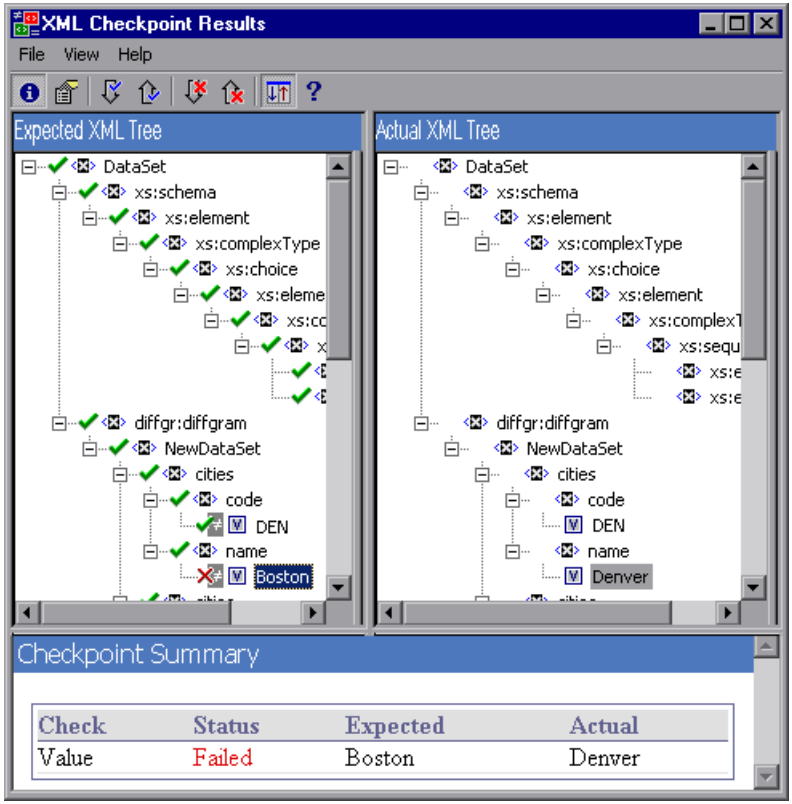
When you click the **View XML Checkpoint Results** button from the Test Results window, the XML Checkpoint Results window displays the XML file hierarchy.

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.



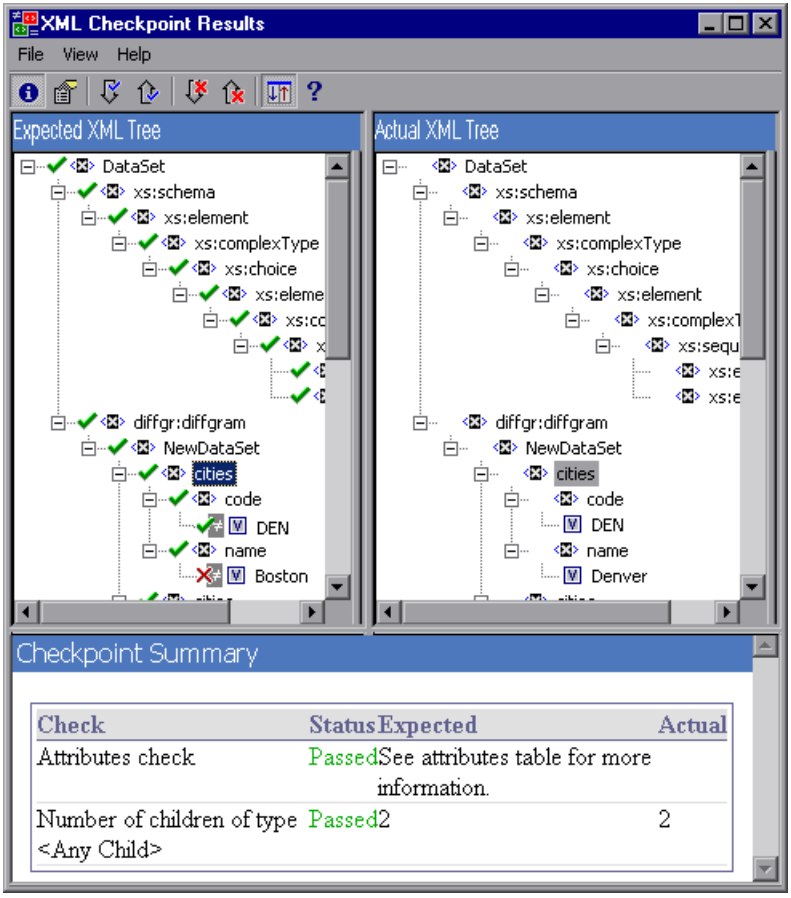
Navigating the XML Checkpoint Results Window

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.

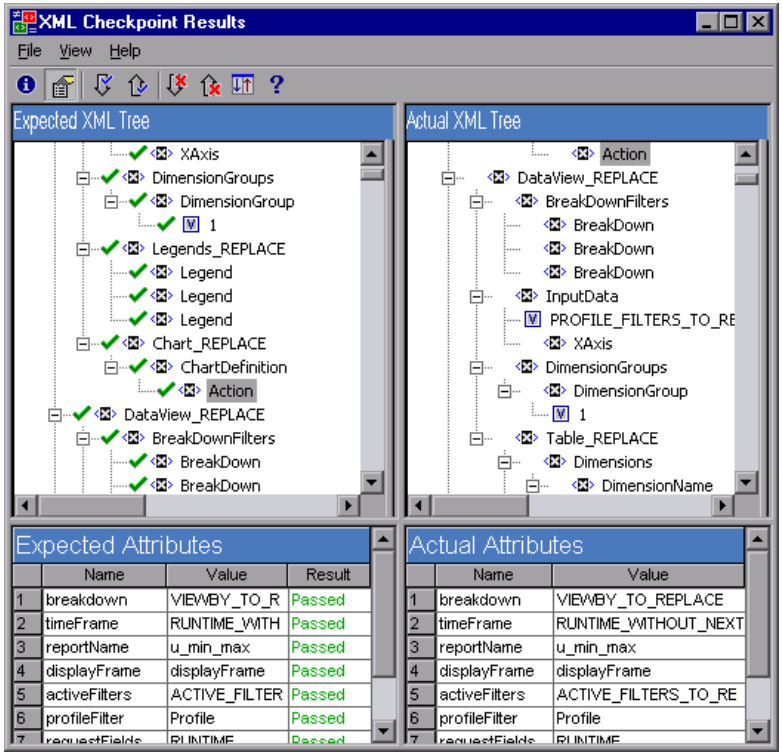
The following example displays the Checkpoint Summary for the cities element in an XML file.



➤ **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

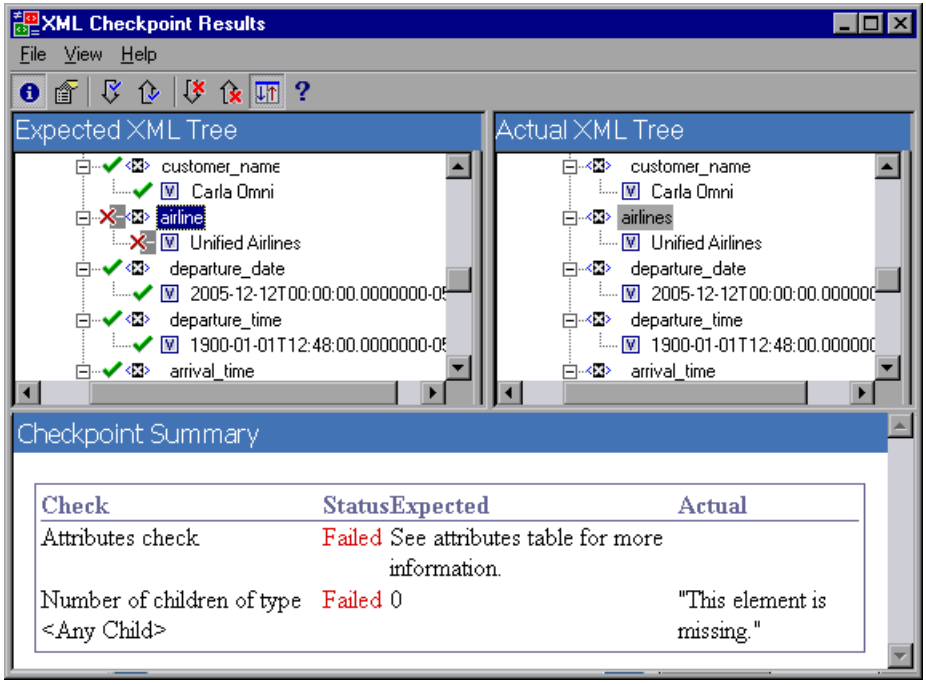
Below are four sample XML checkpoint scenarios. Each example describes the changes that occurred in the actual XML document, explains how you locate the cause of the problem in the XML checkpoint results, and displays the corresponding XML Checkpoint Results window.

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.

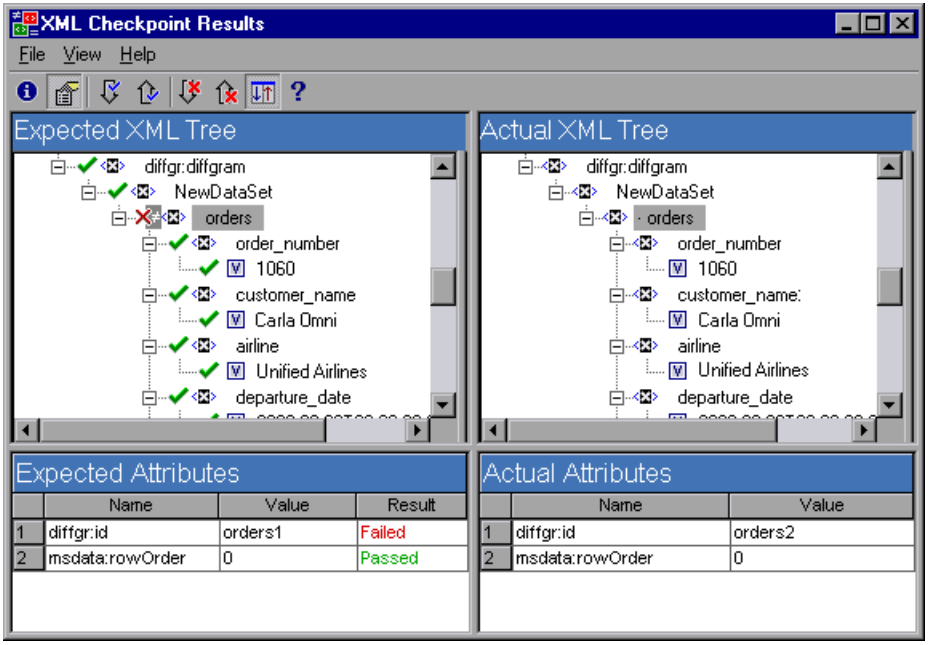


Scenario 2

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.

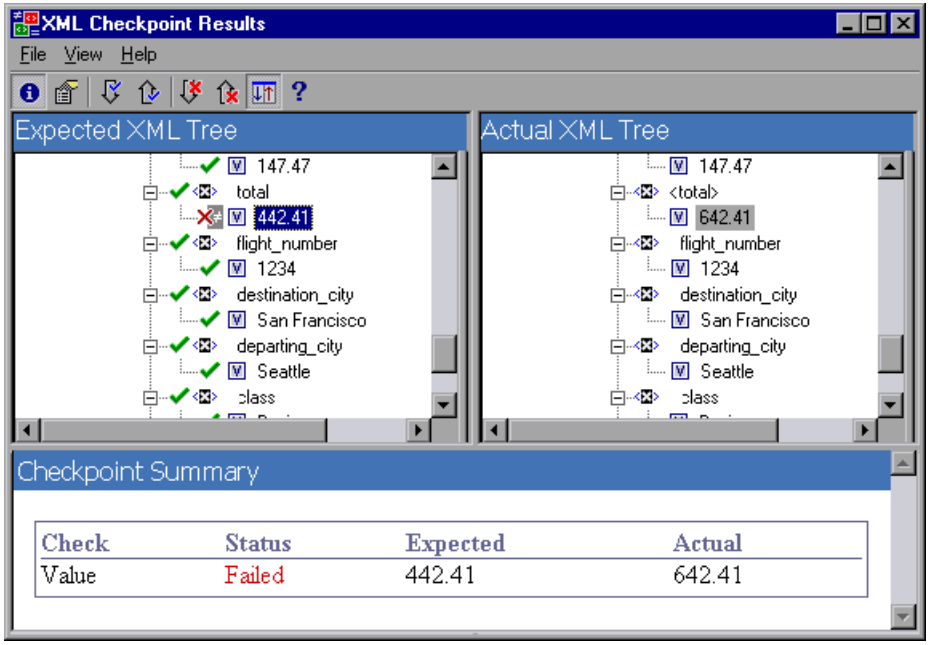


Scenario 3

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.



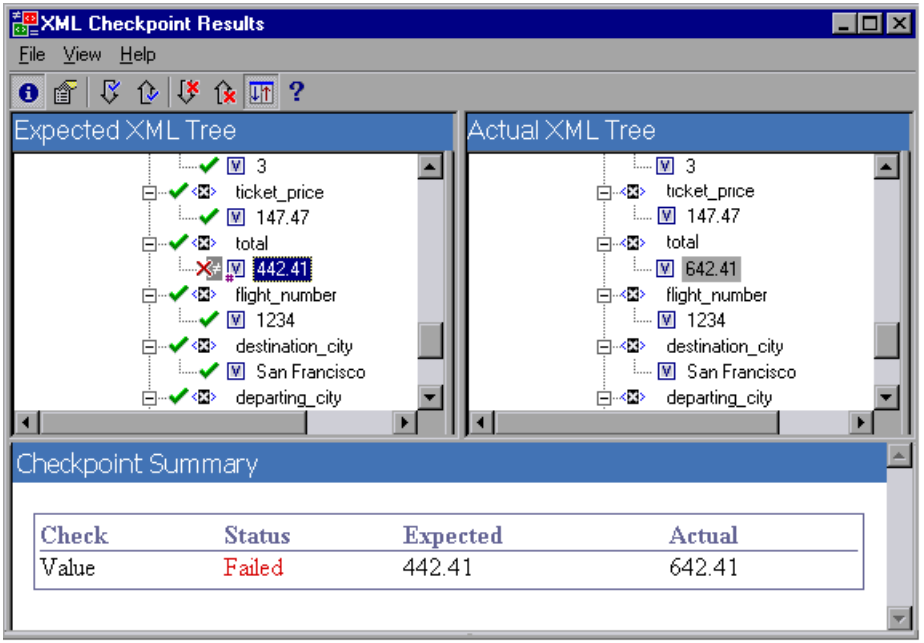
Scenario 4

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.

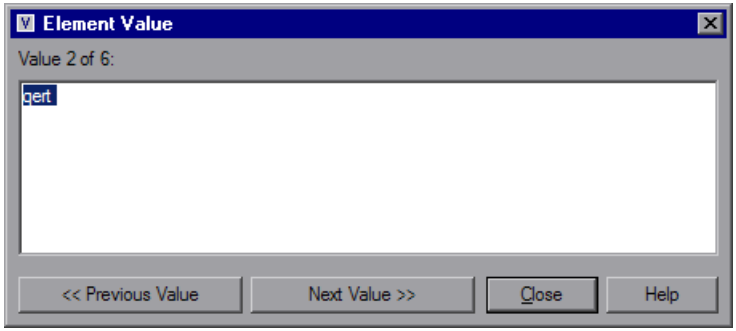
Using the Checkpoint Summary pane, you can compare the expected and actual values of the total element.



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

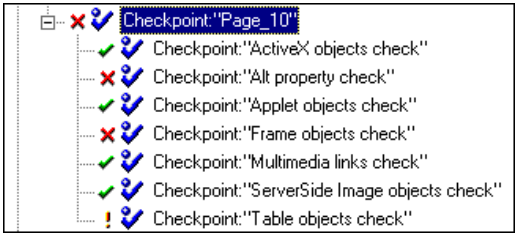
Option	Description
Value x **of** y	Indicates the ordinal position of the selected value within the **Expected XML Tree** or **Actual XML Tree**.
edit window	Displays the full value of the element or attribute in a multi-line window.

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:



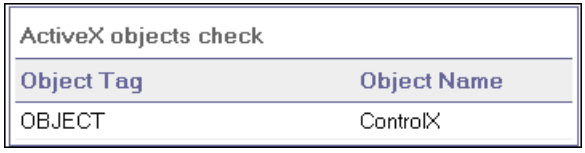
The test result details provide information that can help you pinpoint parts of your Web site that may not conform to the W3C Web Content Accessibility Guidelines. The information provided for each check is based on the W3C requirements.

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

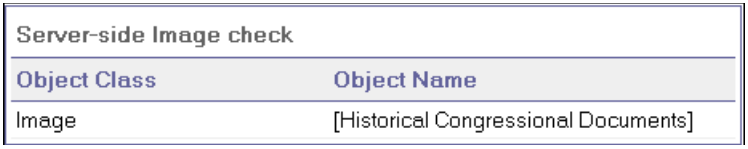
The bottom pane in the Result Details tab of the Test Results window displays the captured page or frame, so that you can see the frames listed in the Frame Titles check list.

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

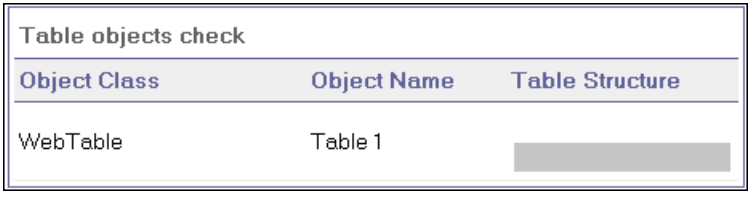
Guideline 1.2 of the W3C Web Content Accessibility Guidelines requires you to provide redundant text links for each active region of a server-side image map. Guideline 9.1 recommends that you provide client-side image maps instead of server-side image maps except where the regions cannot be defined with 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.

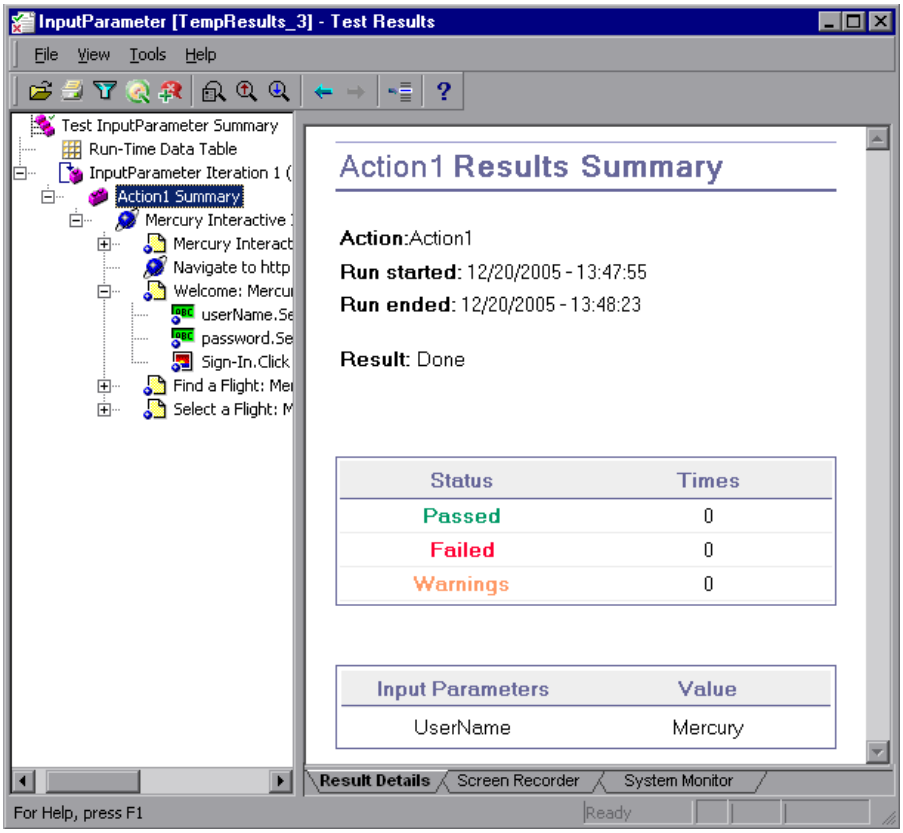
Viewing Parameterized Values in the Test Results Window

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.”

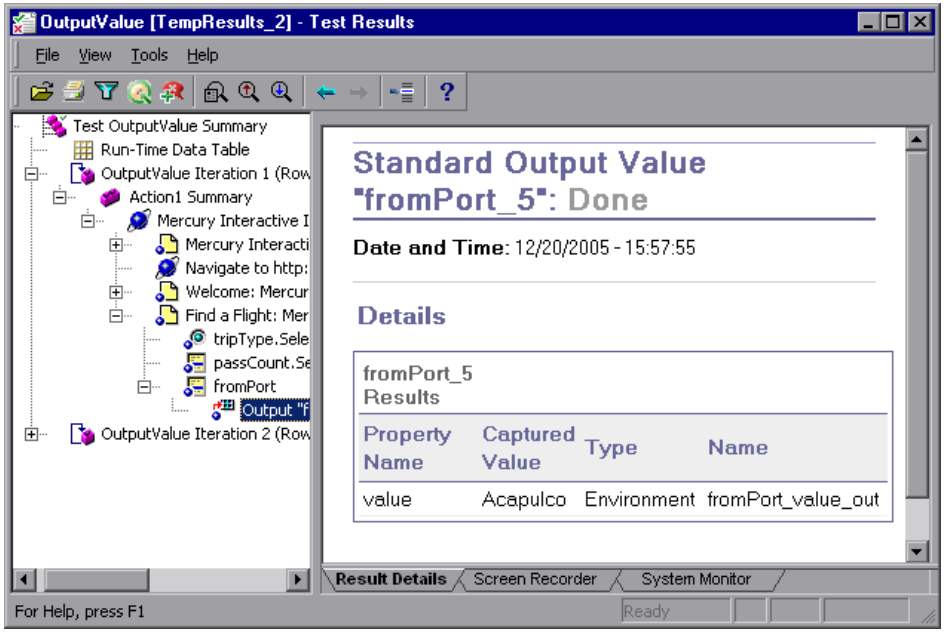
Viewing Output Value Results in the Test Results Window

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.

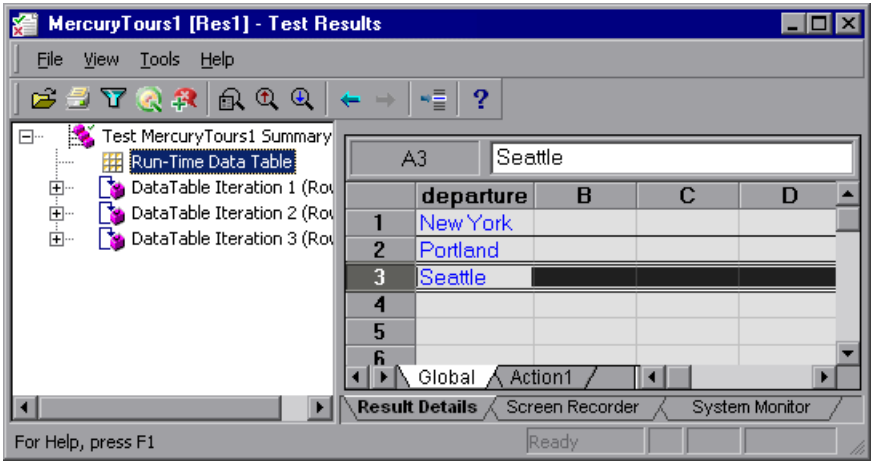
Viewing the Run-Time Data Table

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.



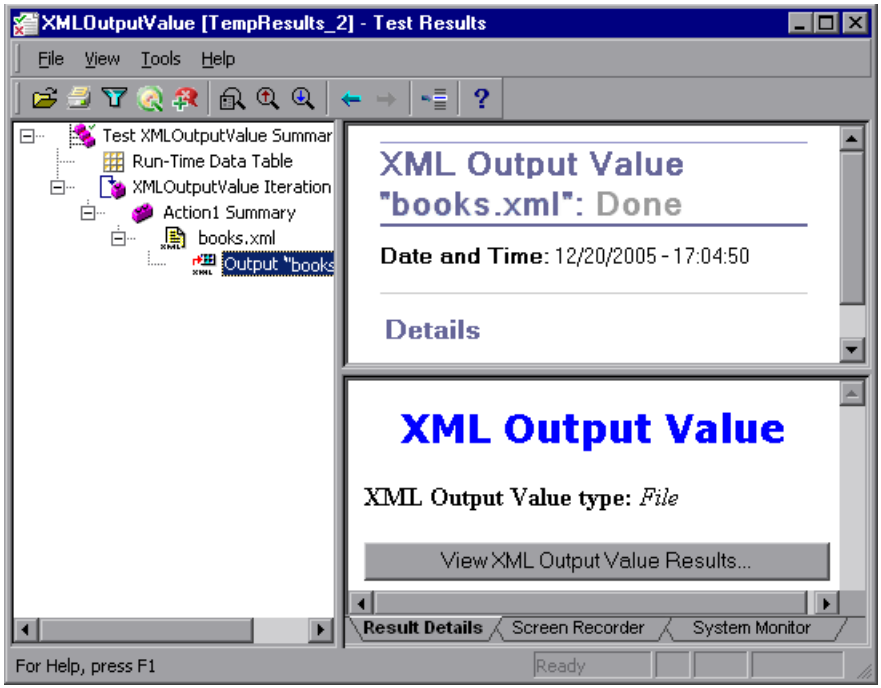
In the above example, the Run-Time Data Table contains the parameterized flight departure values.

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.

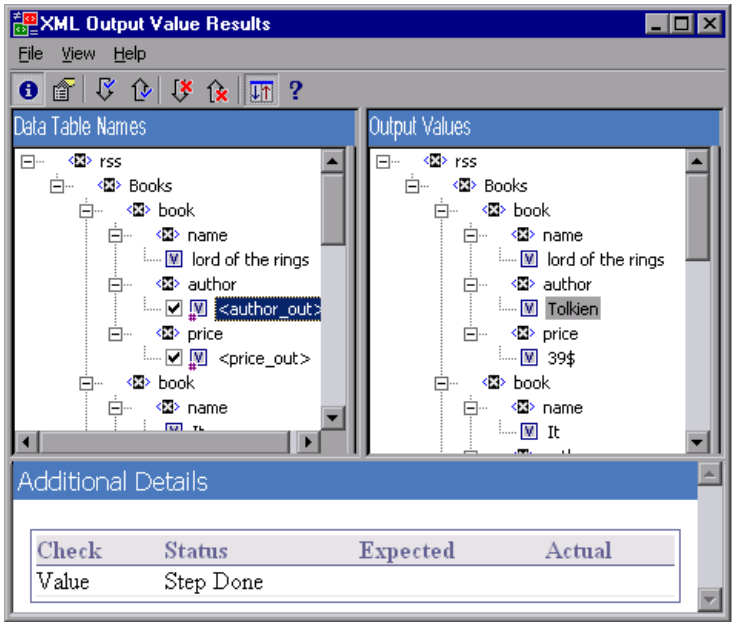
Understanding the XML Output Value Results Window

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.

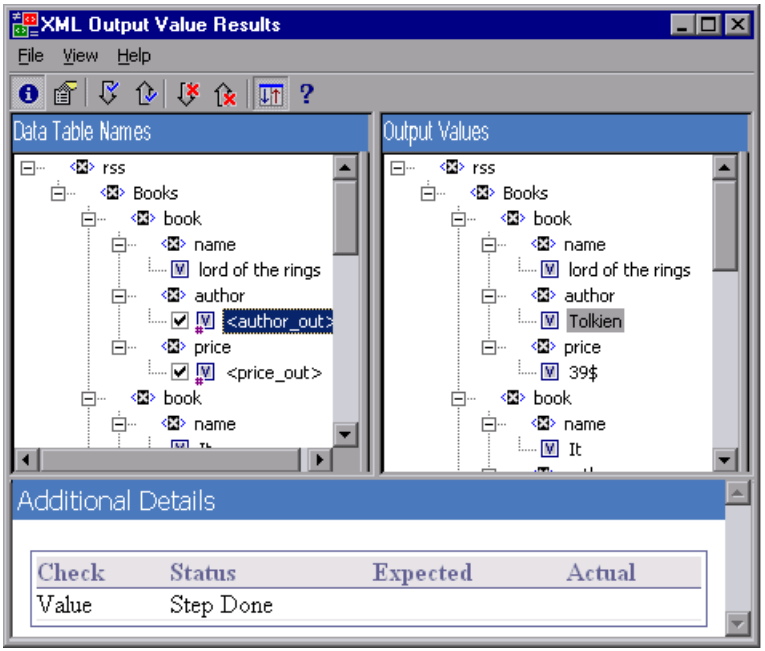


Navigating the XML Output Value Results Window

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.

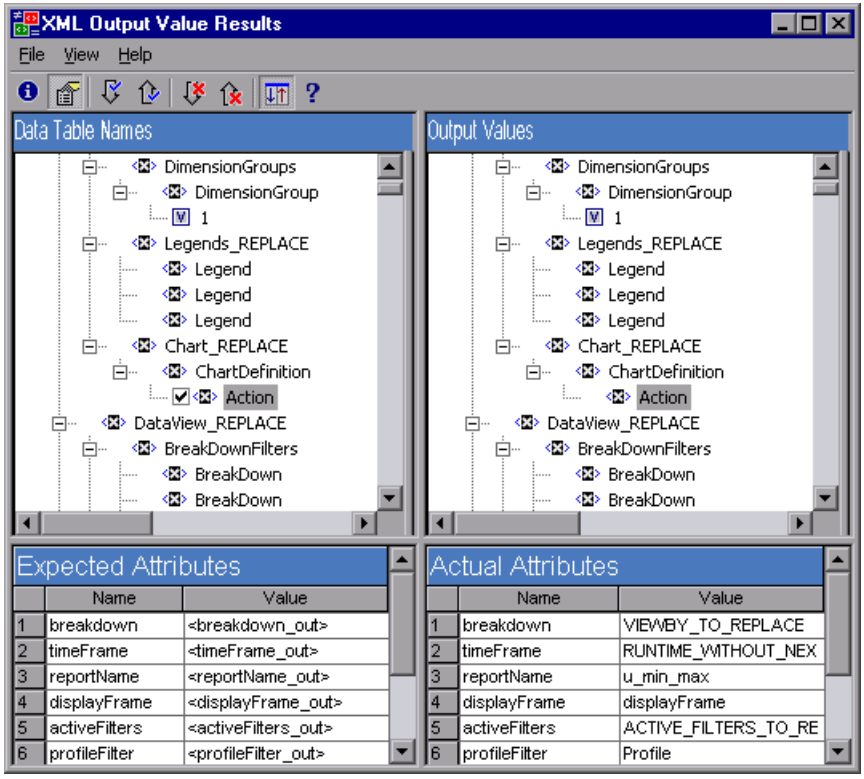
You can use the commands or toolbar buttons described below to navigate your XML output value results:

➤ **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.

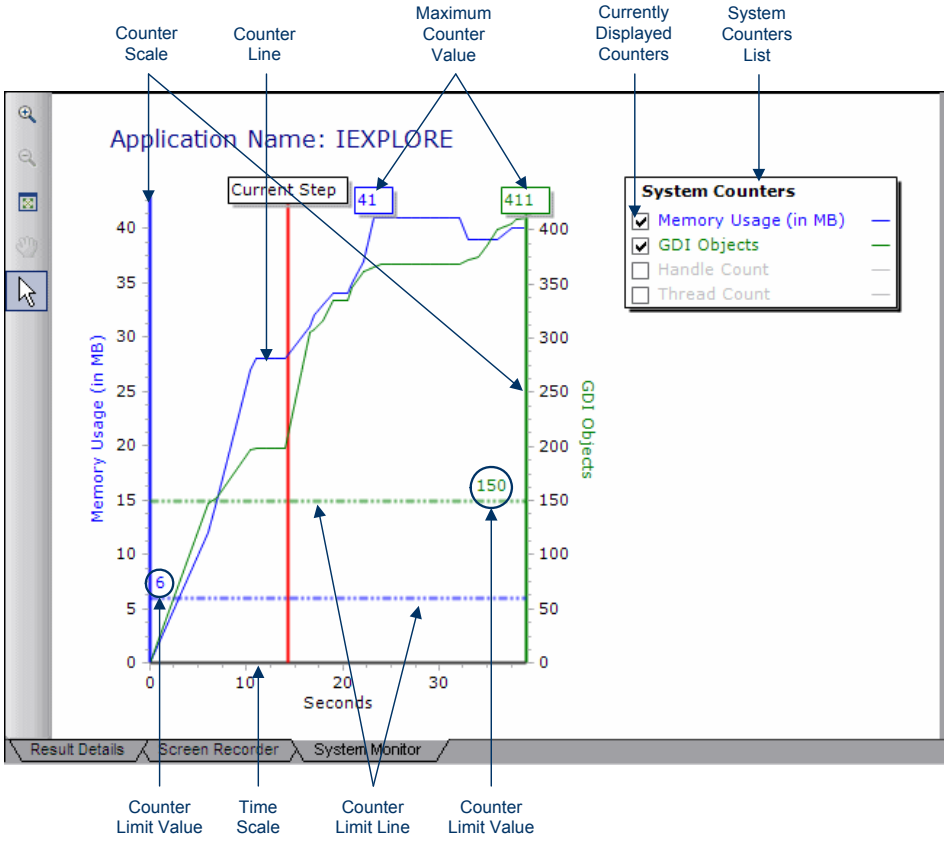
➤ **Help Topics.** Select **Help** > **Help Topics** or click the **Help Topics** button to view help on the XML Output Value Results window.

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.

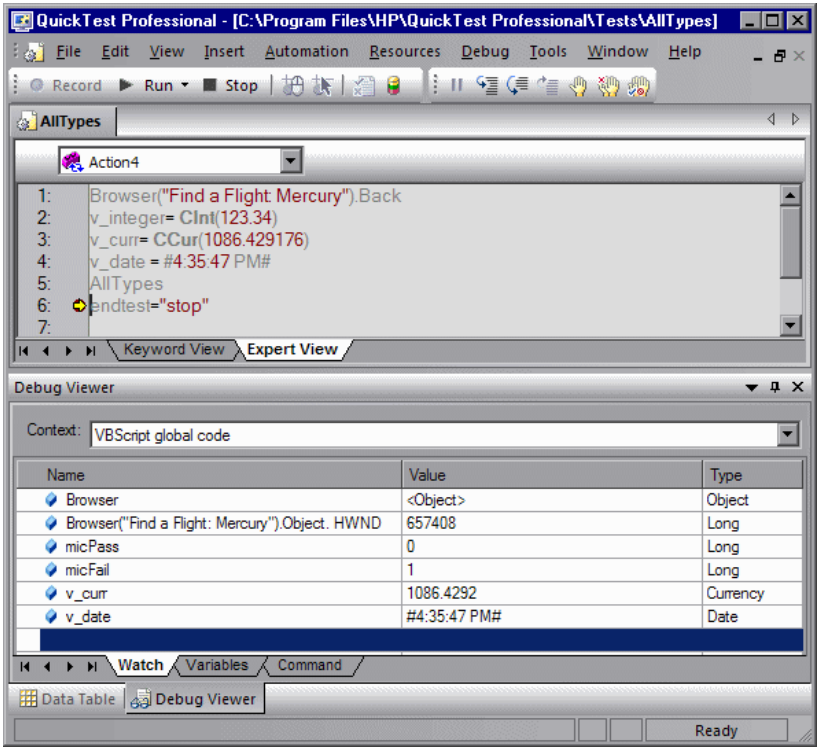
The Debug Viewer Pane: Watch Tab

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:

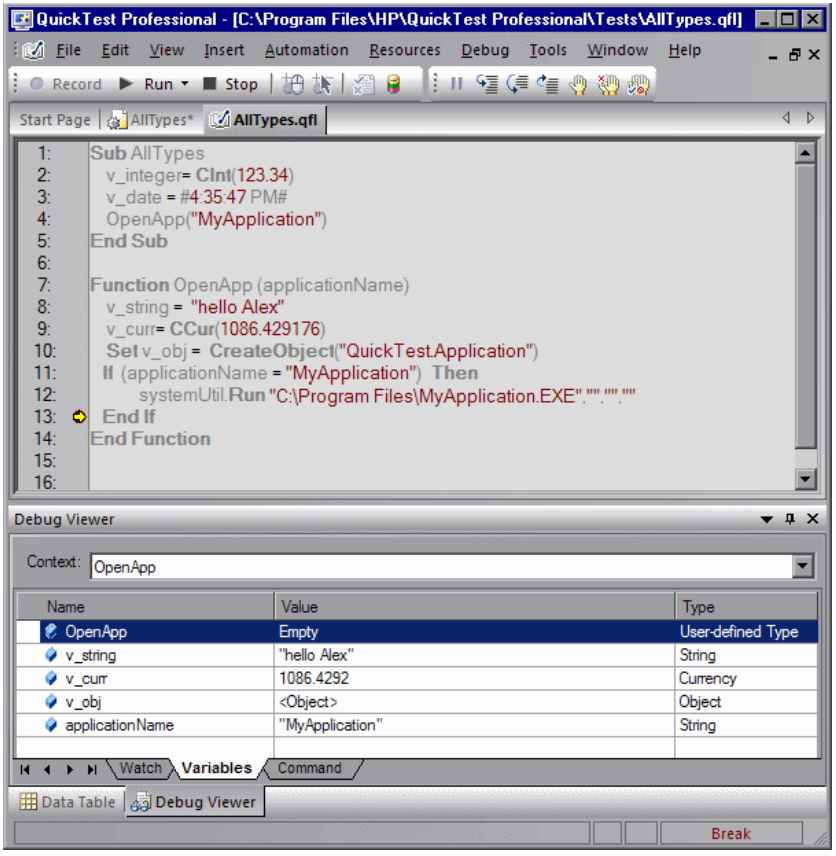
In the Watch tab, select the row that you want to remove and press the Delete key on your keyboard.

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.



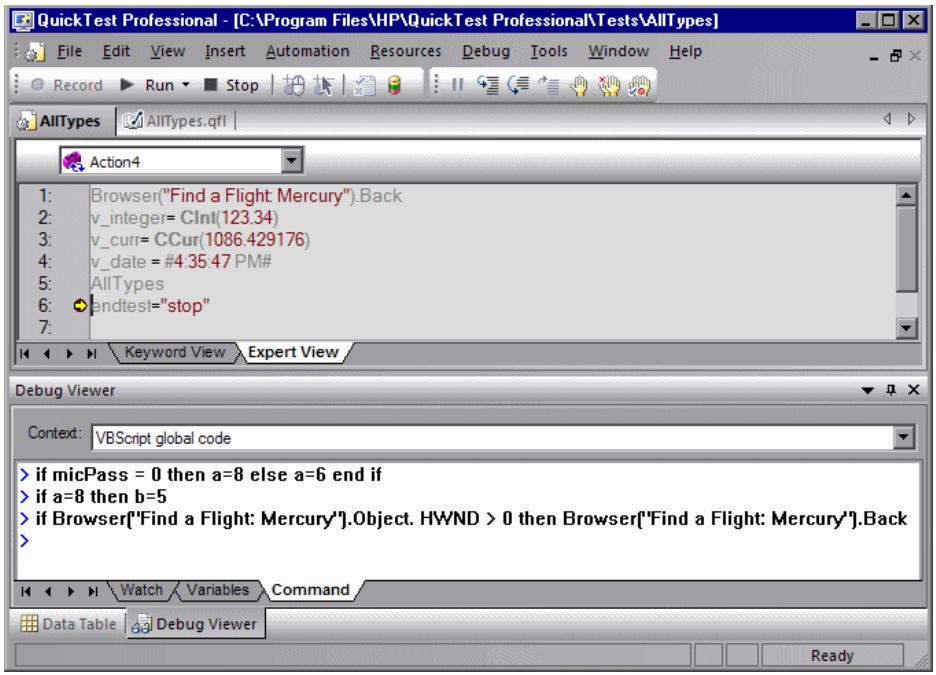
Debug Viewer Variables Tab Details

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.

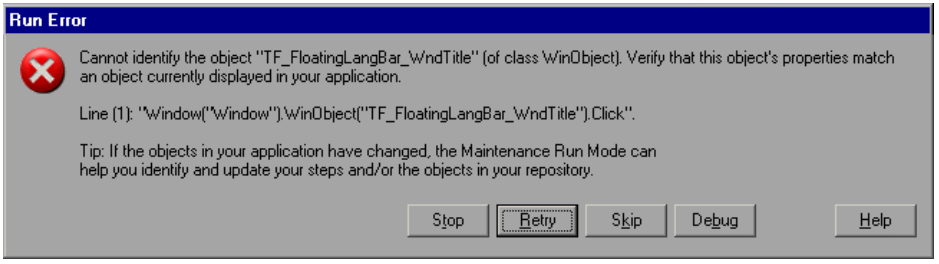
- **Right-click context menu.** Provides commands that you can use to edit the content of the Command tab.
  - The **Cut**, **Copy**, and **Paste** commands enable you to use the clipboard to copy text from the command history and to edit the active command line.
  - The **Clear All** command enables 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.

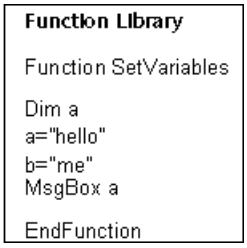
Practicing Debugging an Action or a Function

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.

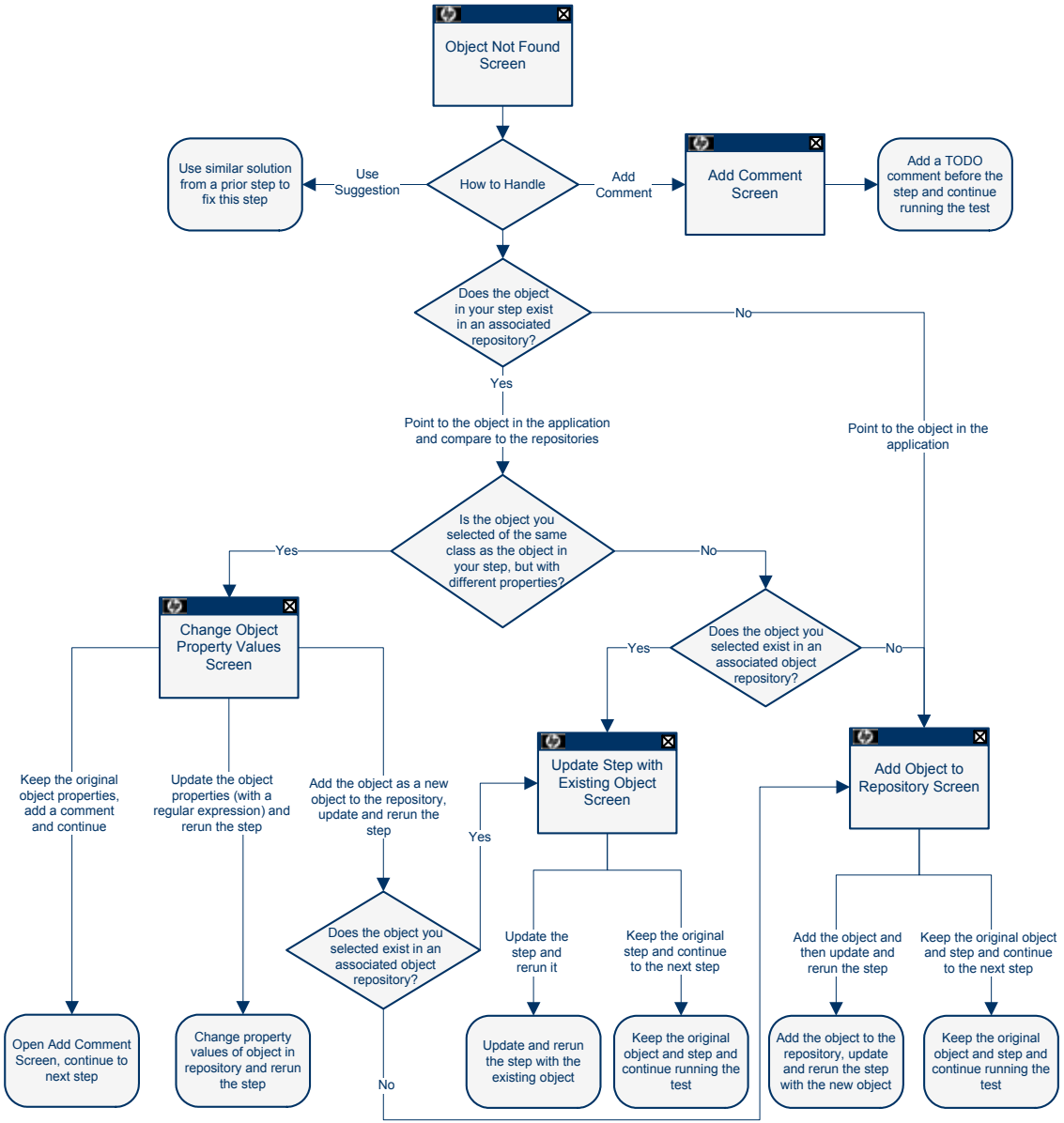
Running Tests with the Maintenance Run Wizard

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.

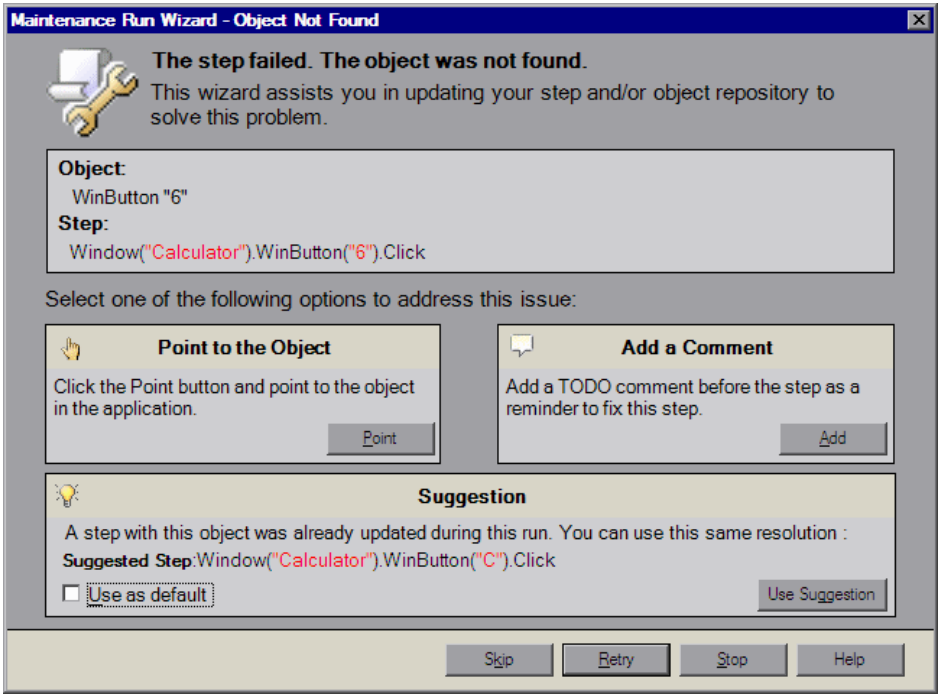
2. Specify the results location and the input parameter values (if applicable) for the Maintenance Run Mode session. For more information, see "The Run Dialog Box: Results Location Tab" on page 960, and "The Run Dialog Box: Input Parameters Tab" on page 962.

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:

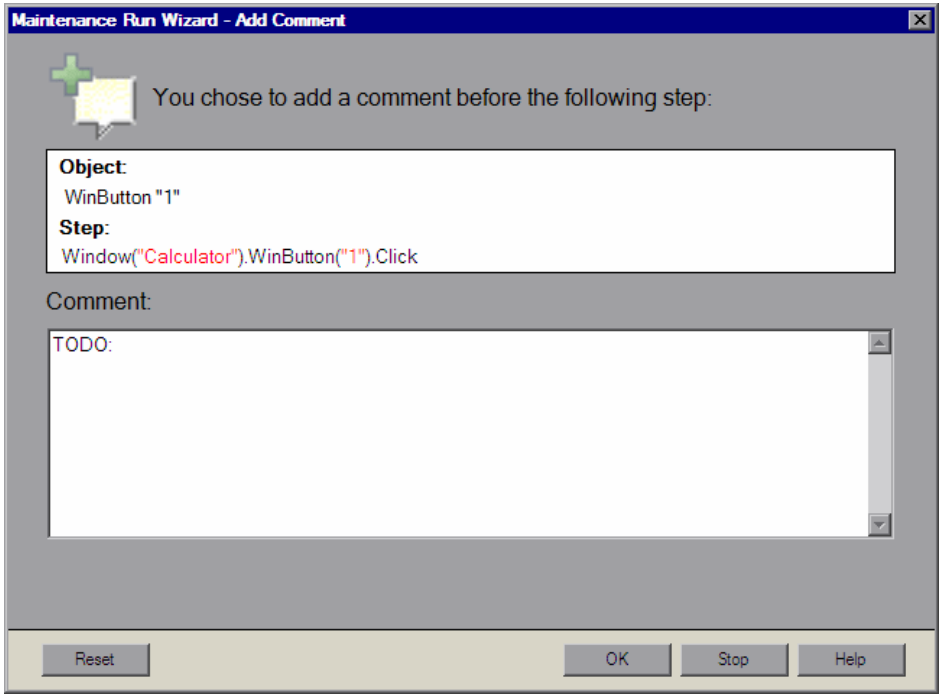
- **Skip.** Skips the current step in the test and continues to run the Maintenance Run wizard on the remainder of the test. This can be used when the problem is in the application being tested and not the QuickTest test.

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.

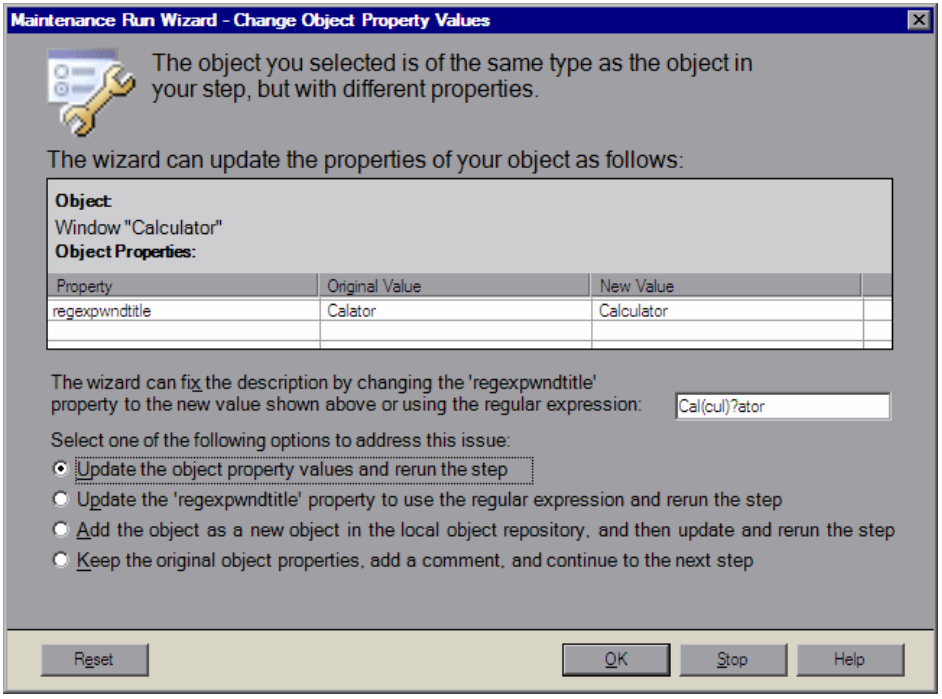


The Add Comment screen creates a comment in your test beginning with the word TODO along with text you add, as a reminder to fix the step at a later time.

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

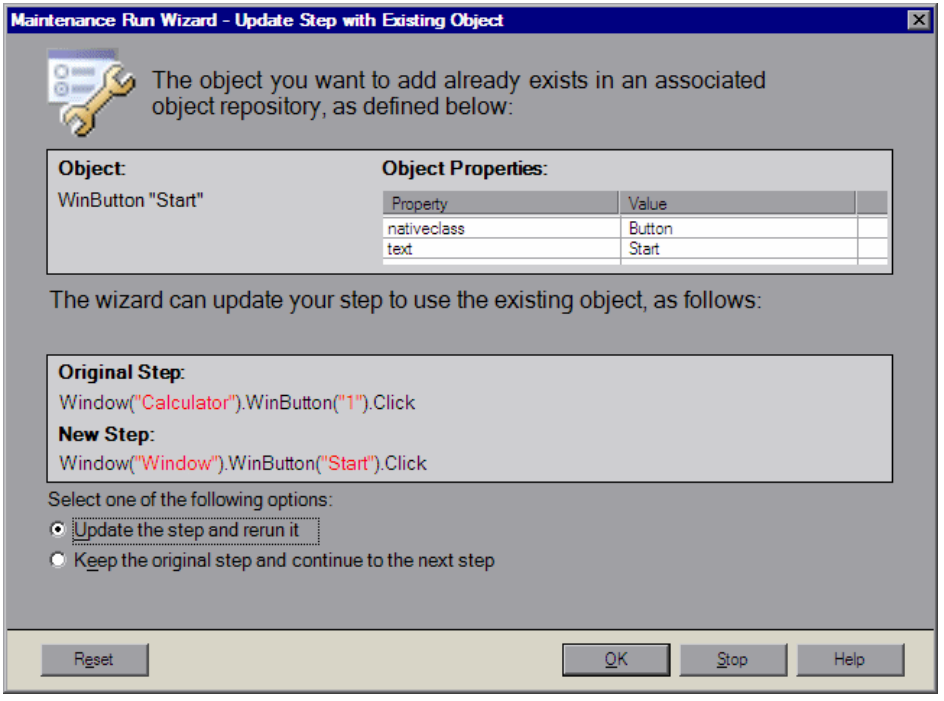
The Update Step with Existing Object screen opens if the object to which you pointed in the Object Not Found screen exists in an associated object repository and:

➤ 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

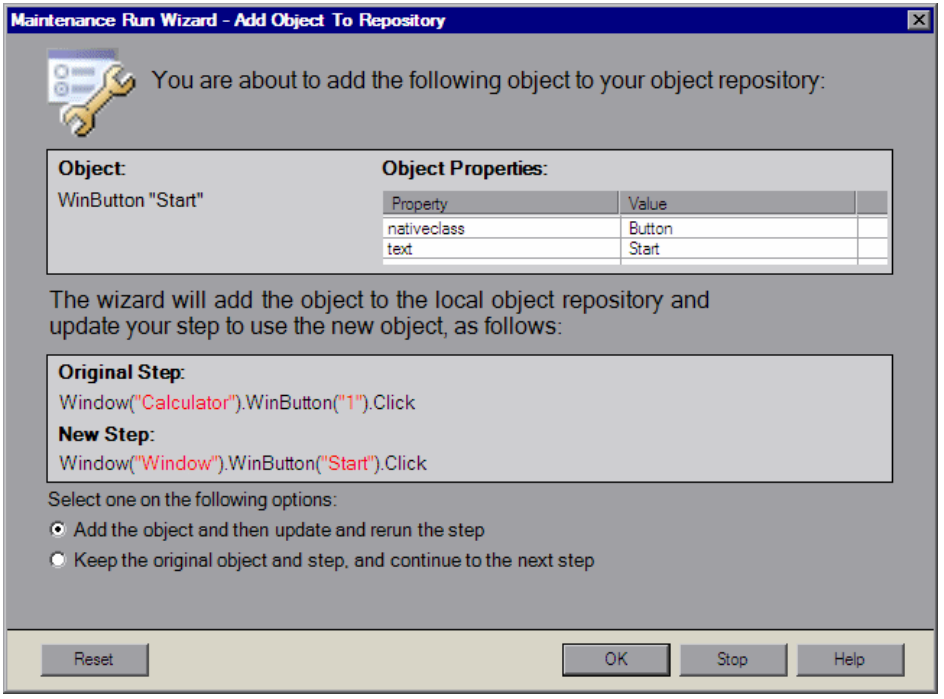
The Add Object to Repository screen opens in the following cases:

- 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:

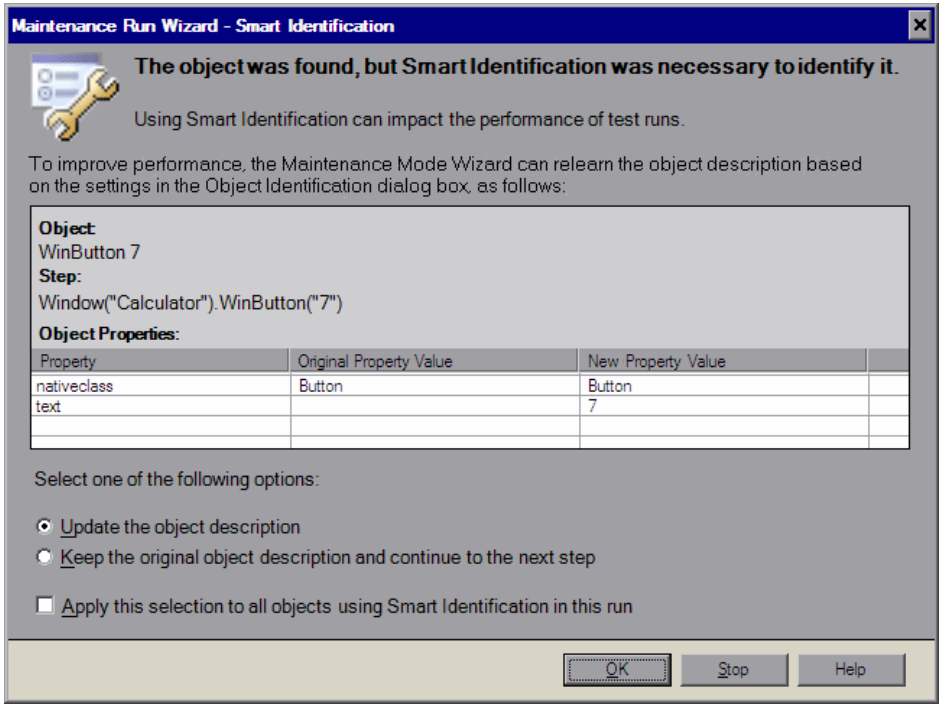
- The Maintenance Run wizard makes changes to the local object repository only. If you want the new object to appear in a shared object repository use the Object Repository Manager. For more information, see "Performing Merge Operations" on page 240.
- The Maintenance Run wizard does not remove the original 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:

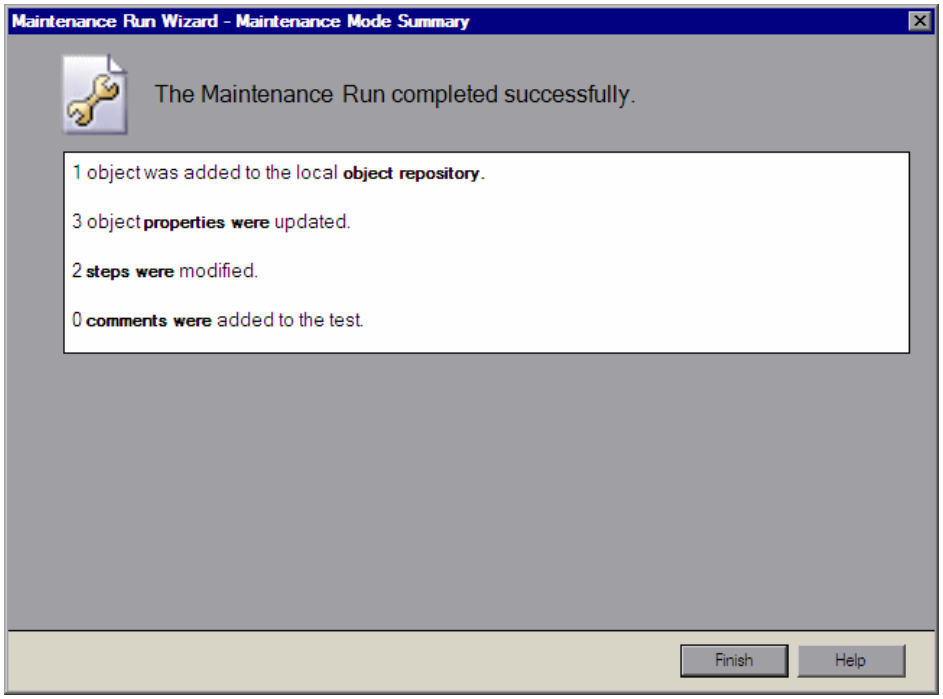
Section	Description
Object	The object in your application that required the Smart Identification mechanism to be identified.
Step	The step in your test in which the object is referenced.
Object Properties	**Property.** The list of properties in the old and new object description. **Original Property 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

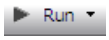
When you run a test in Update Run Mode, QuickTest runs the test to update the test object descriptions, the Active Screen images and values, and/or the expected checkpoint values. After you save the test, the updated data is used for subsequent runs.

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:

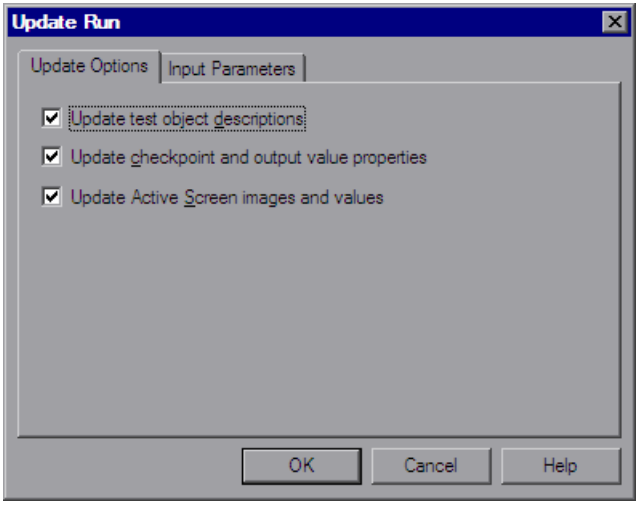
- When a test runs in Update Run Mode, it does not update parameterized values, such as Data Table data and environment variables. For information on parameterized values and environment variables, see Chapter 24, "Parameterizing Values." Update Run Mode does not modify the property values of existing 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.

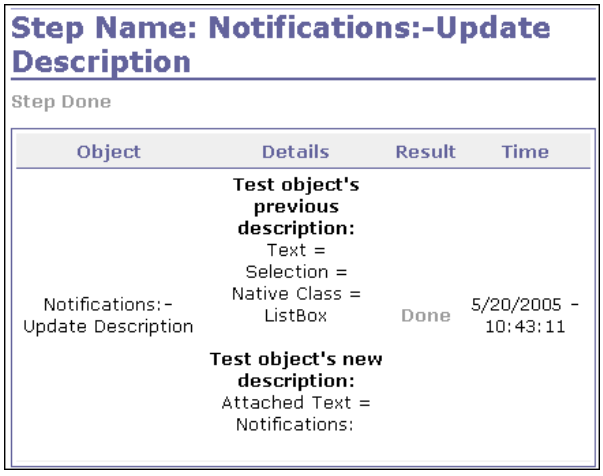
For information on viewing the results, see Chapter 33, "Viewing Run Session Results."

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:



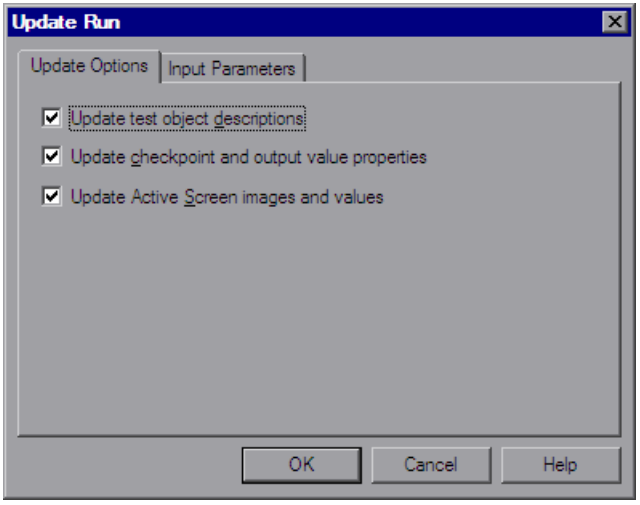
Step Name: Notifications:-Update Description

Step Done

Object	Details	Result	Time
Notifications:-Update Description	**Test object's previous description:** Text = Selection = Native Class = ListBox **Test object's new description:** Attached Text = Notifications:	Done	5/20/2005 - 10:43:11

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.

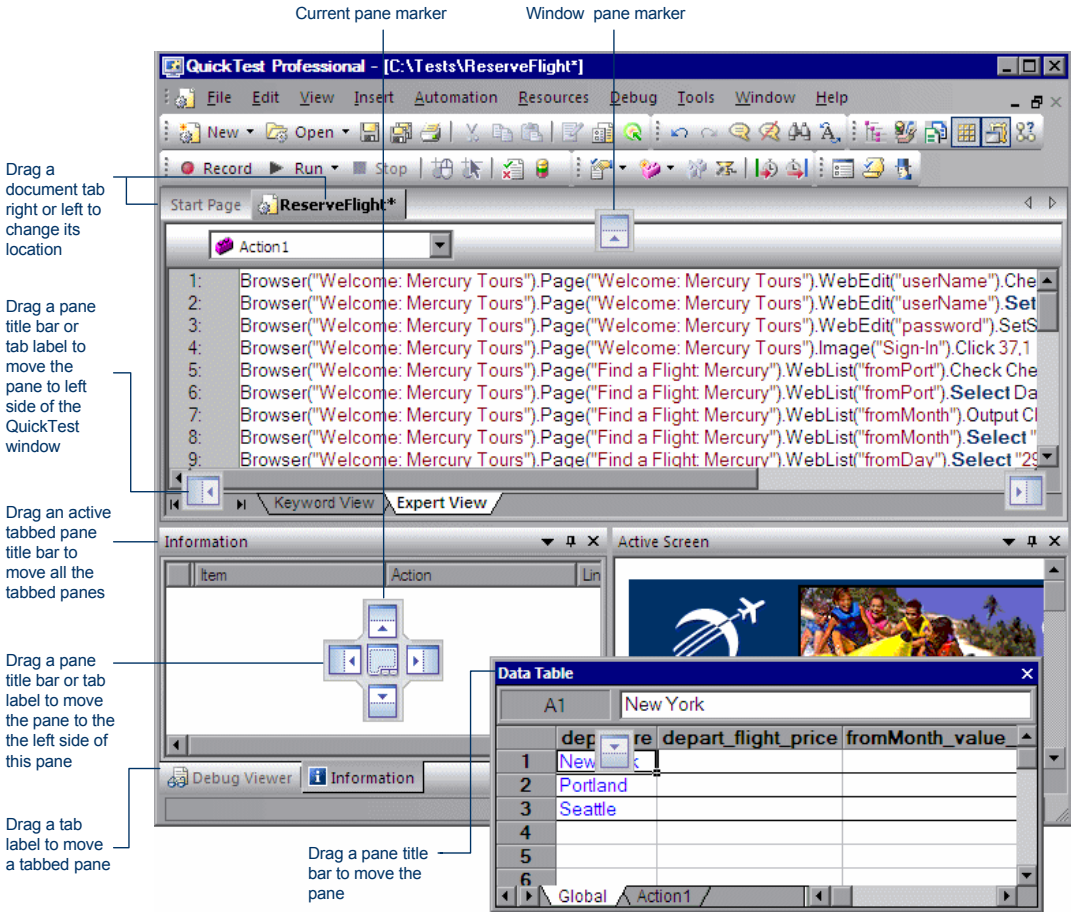
While dragging a pane, markers are displayed on the QuickTest window. If you hold the cursor over one of these markers, the area represented by the marker is shaded, enabling you to preview the window layout if the pane is moved to the selected position.

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.)

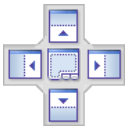
For example, you can move the Data Table tabbed pane located at the bottom left to be a new pane at the top right of the window. As you drag the pane, markers are displayed in the active pane and on each edge of the QuickTest window.



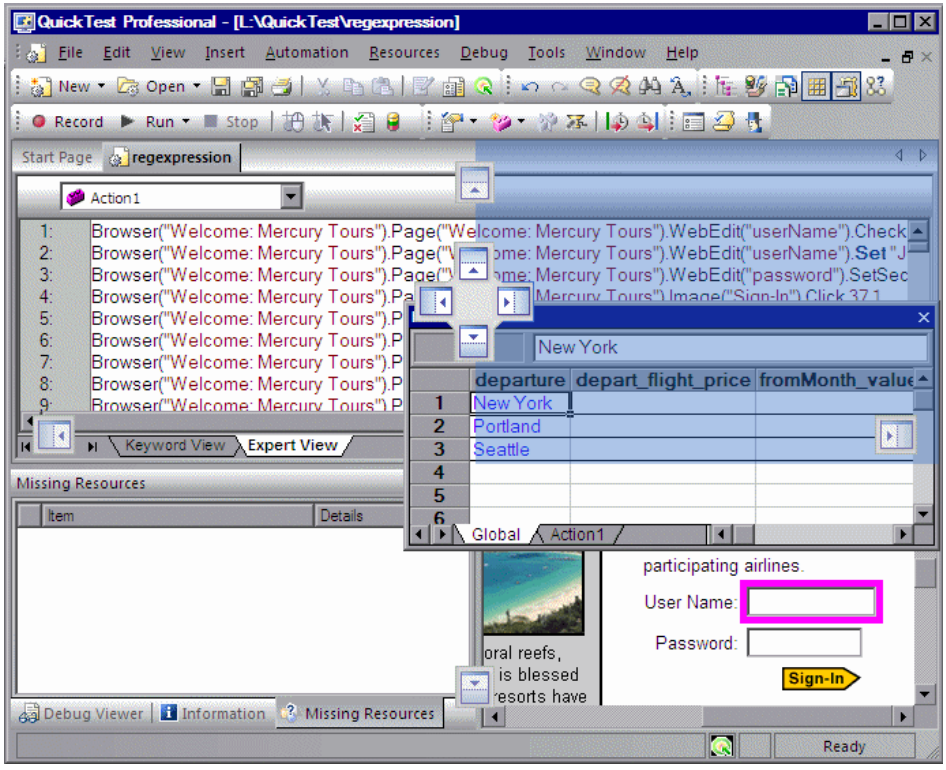
Tips:

- 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.

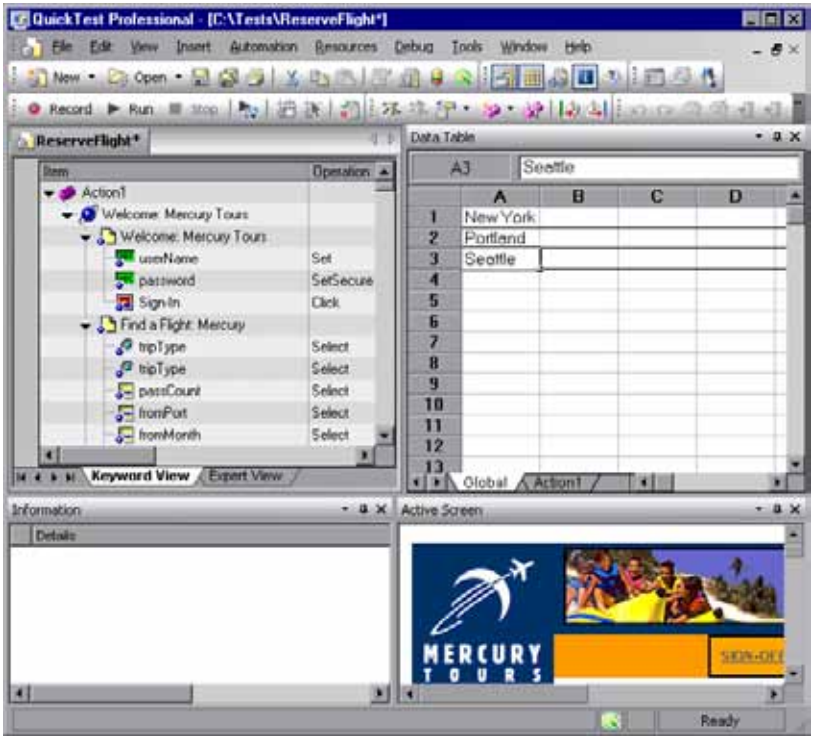
The following markers are displayed:

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:

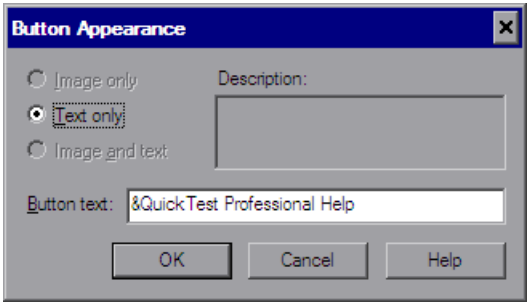
Option	Description
Restore Default	Restores the default setting for the button. This selection is disabled for menus.
Copy Button Image	Copies the button image to the clipboard. This selection is disabled for items that have no default image.

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.

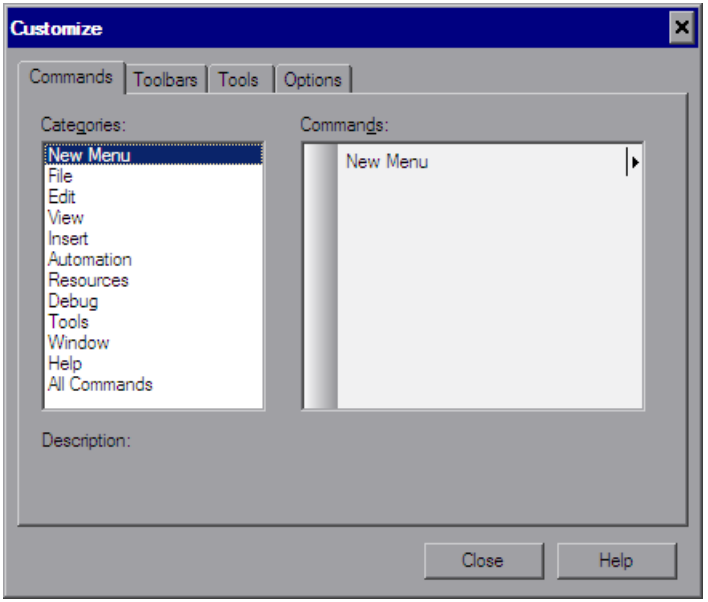
Option	Description
Description	The description of the button.
Button text	The text label for the button or menu. The text label for the button can be modified when either the **Text only** or **Image and text** radio buttons are selected. You can create a mnemonic (an underlined character for keyboard 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.

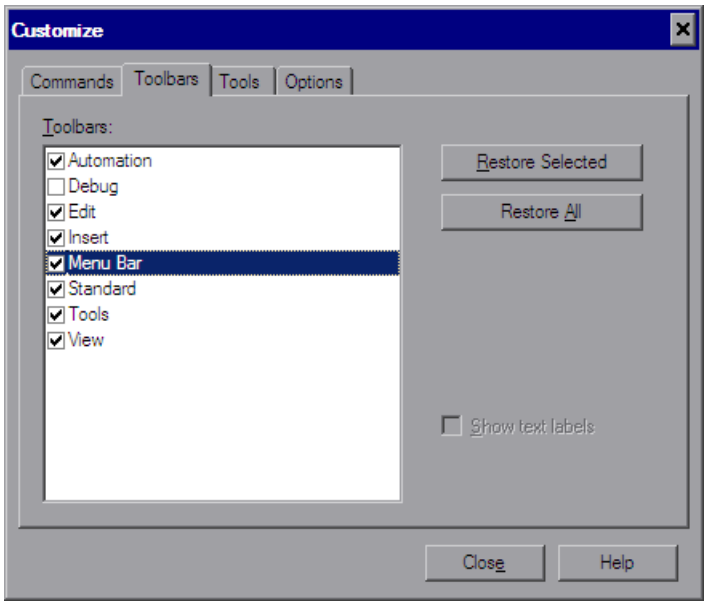
To add a command to a toolbar or menu:

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:



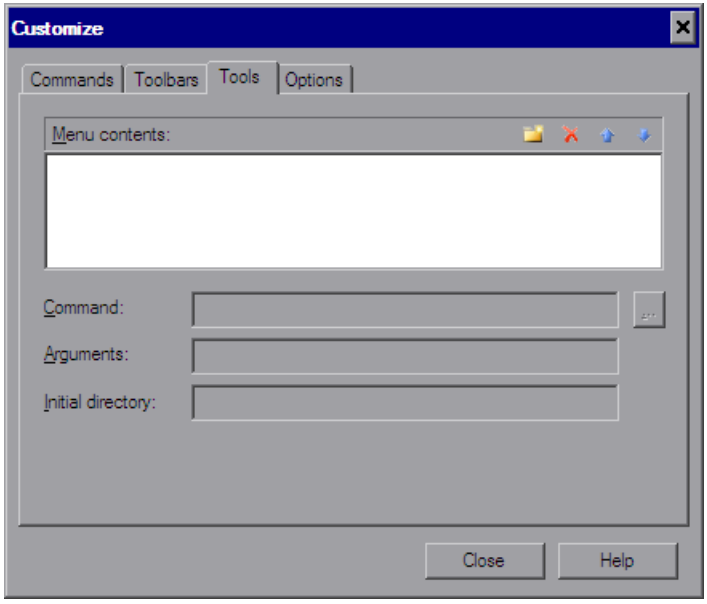
Customize Dialog box - Toolbars Tab Options

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:



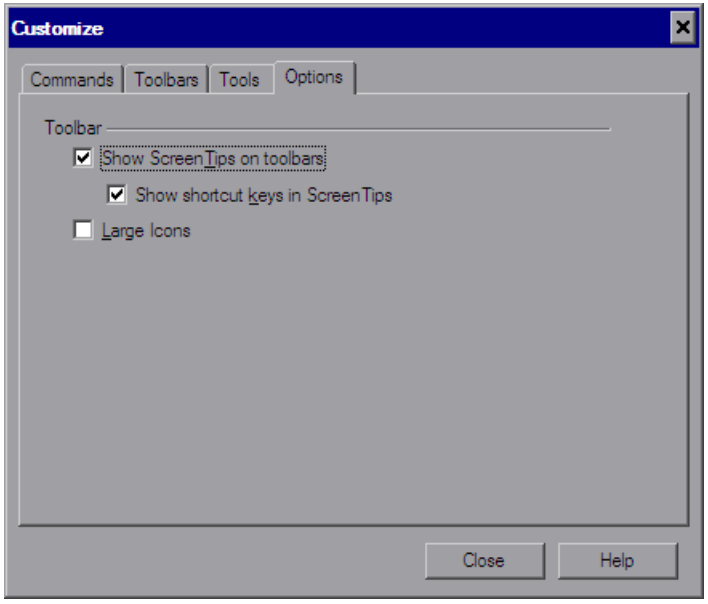
Customize Dialog box - Tools Tab Options

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

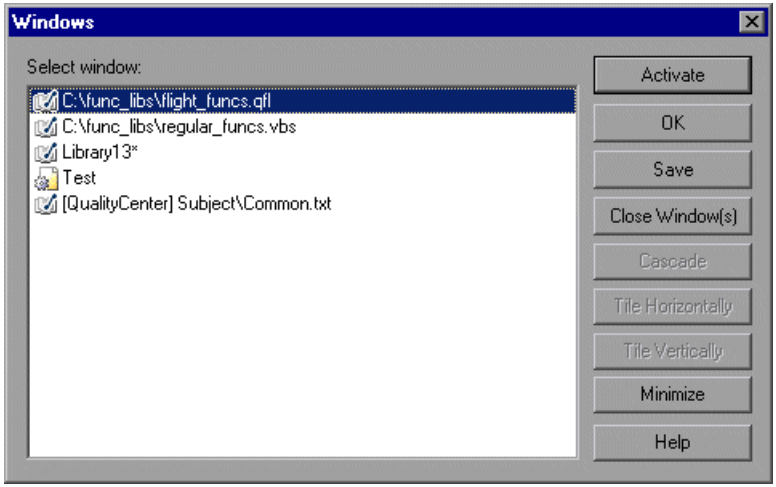
QuickTest enables you to open and work on one test at a time. In addition, you can open and work on multiple function libraries simultaneously. You can open any function library, regardless of whether it is associated with the currently open test.

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

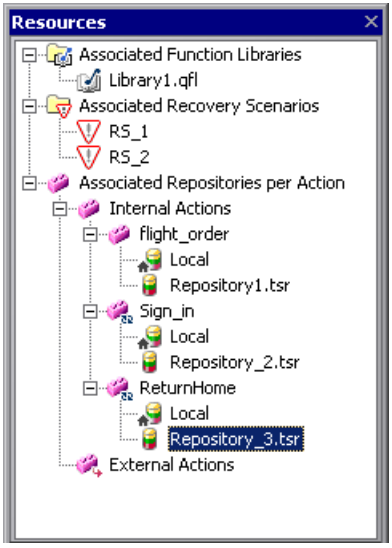
QuickTest enables you to manage the resources associated with your test in one pane. Using the Resources pane, you can associate, remove, open, change the priority, and otherwise manage the function libraries, recovery scenarios, and object repositories in your test.

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.

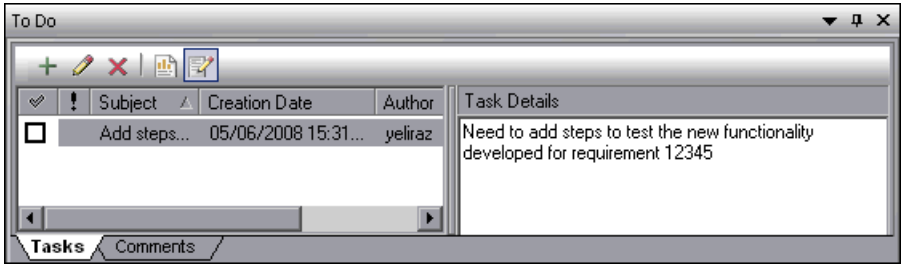
The To Do Pane

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.

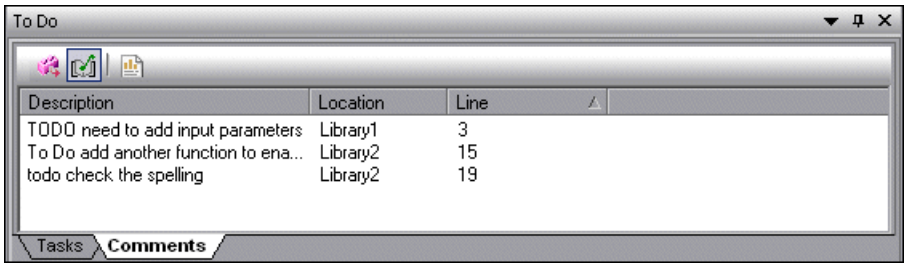
Tasks Tab Context Menu Options

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.

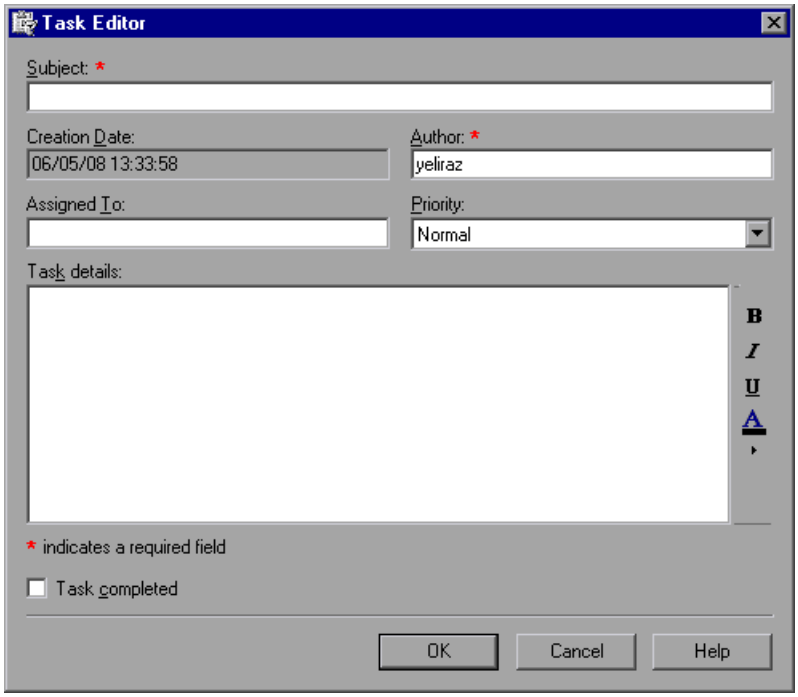
Comments Tab Context Menu Options

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.

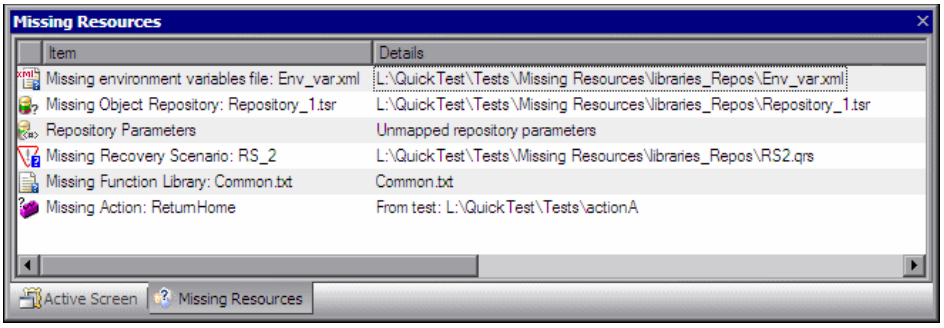
This chapter includes:

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.

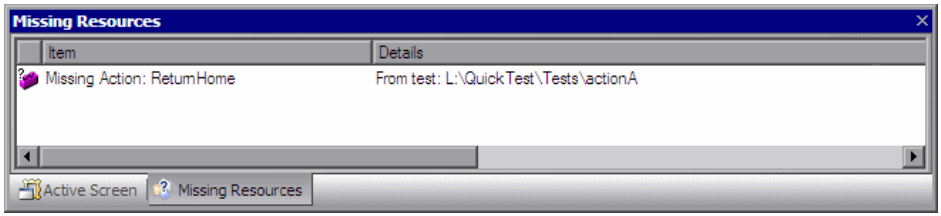
- **Object Repositories.** Displays a row for each shared object repository that cannot be found, specifying the path QuickTest uses to search for the shared object repository.
- **Recovery Scenarios.** Displays a row for each recovery scenario that cannot be found, specifying the path QuickTest uses 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:

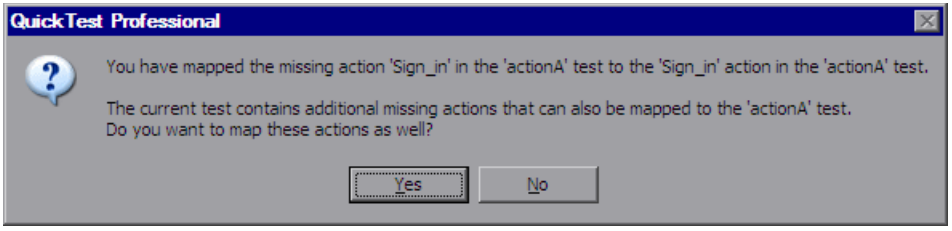
- Locating Missing Actions
- Removing Missing Actions

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:

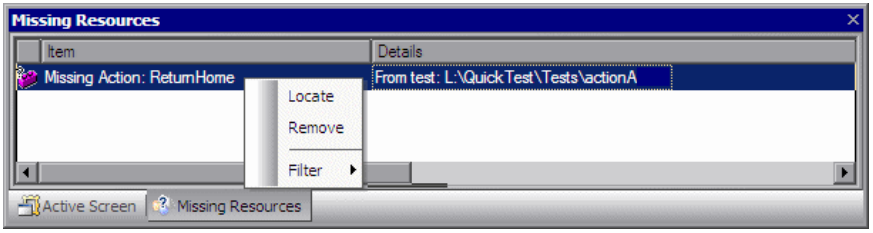


This can occur, for example, if the source test, which contains the actions that are being called was renamed or was moved to another folder.

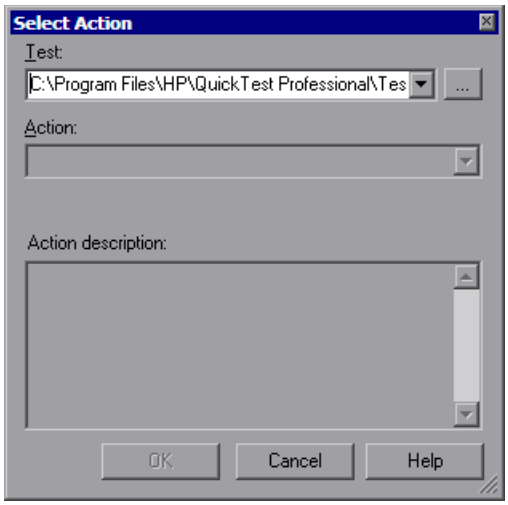
You can instruct QuickTest to locate these actions simultaneously, or you can handle each call to a missing action individually.

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.

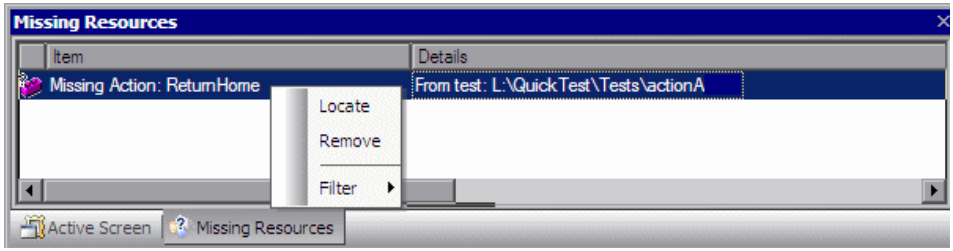
Note: If your test contains additional missing actions that can be located in the same test, QuickTest opens a message box asking you if you want to map these actions as well. Click **Yes** to map all relevant actions, or click **No** to map only the action you specified.

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.

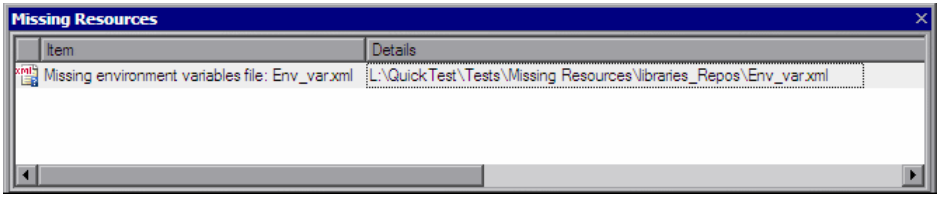


A confirmation message is displayed. Click **OK** to remove the missing action. QuickTest removes the action from your test and removes the action from the Missing Resources pane.

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

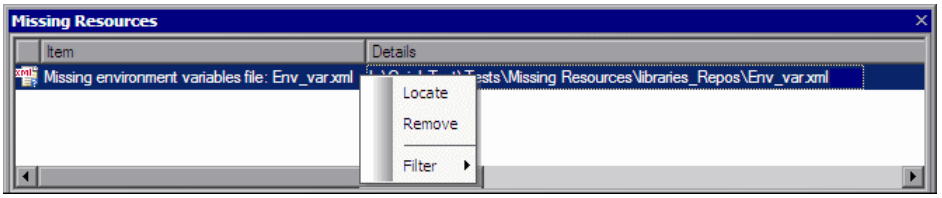
When you open a test that uses an external environment variables file, QuickTest verifies that the file is accessible. If an external environment variables file cannot be found, QuickTest displays its name and path in the Missing Resources pane when you open your test.



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.

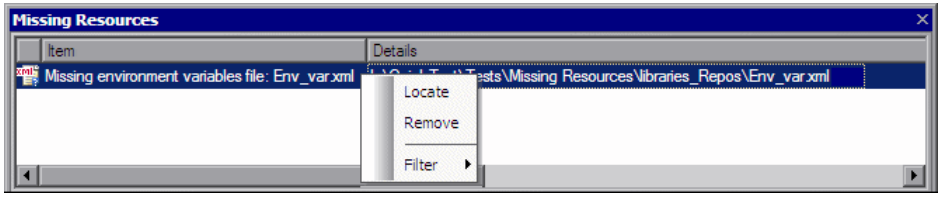


The Locate Environment Variable File dialog box opens.

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.

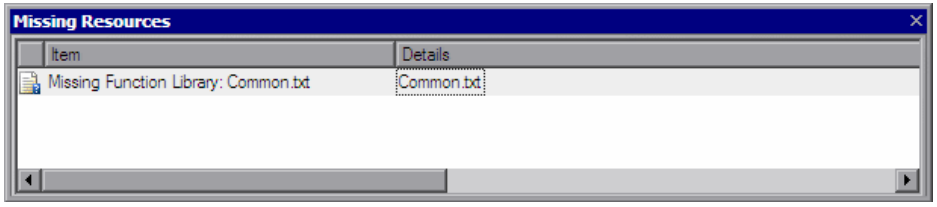
To remove a missing environment variable file:

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

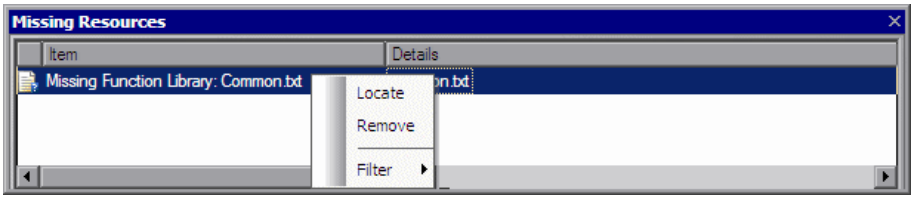
When you open a test that has associated function libraries, QuickTest verifies that the libraries you specified are accessible. If a function library cannot be found, QuickTest displays its name and path in the Missing Resources pane when you open your test.



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.

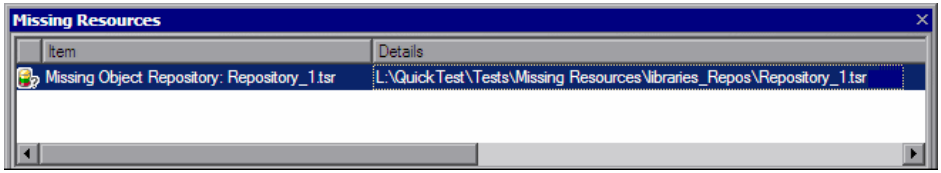
To remove a missing function library:

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.

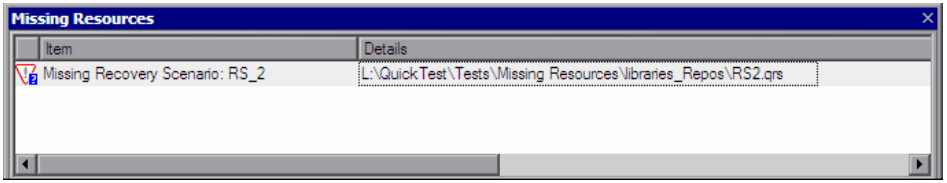
You can right-click the line displaying the missing object repository and select **Resolve** or double-click the line displaying the missing object repository and the Associate Repositories dialog box opens. The Associate Repositories dialog box enables you to associate one or more shared object repositories 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:

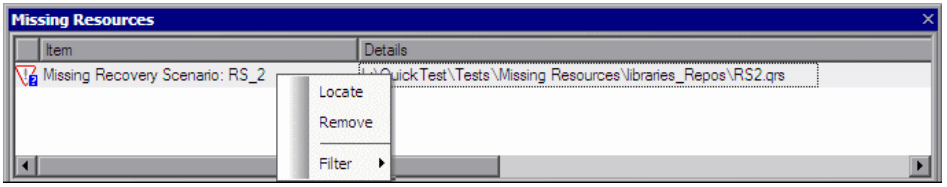
- Locating Missing Recovery Scenarios
- Removing Missing Recovery Scenarios

Locating Missing Recovery Scenarios

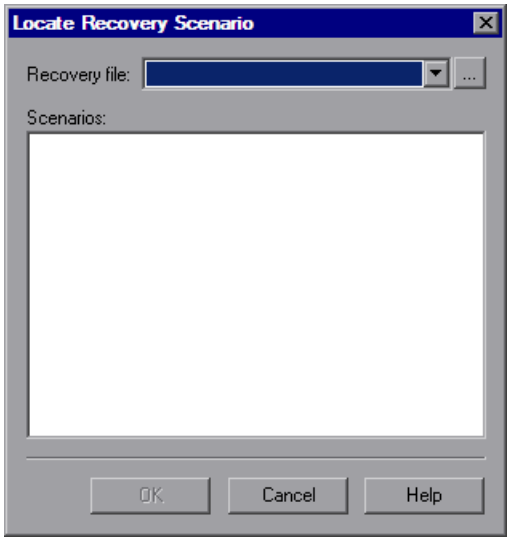
The Missing Resources pane enables you to locate missing recovery scenarios in your test. If your test contains more than one missing recovery scenario, when you locate the missing scenario in a recovery file, QuickTest may identify additional missing scenarios in that file. You can instruct QuickTest 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.

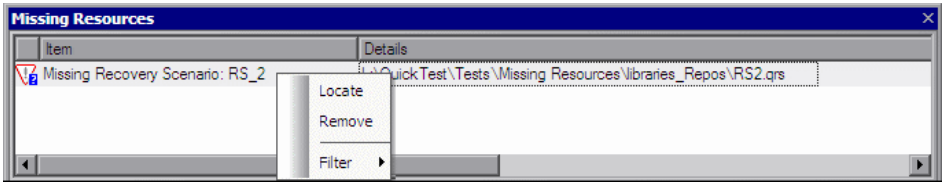
3 Select the missing recovery scenario from the list of recovery scenarios. Click **OK**. The selected recovery scenario is associated with your test and the missing recovery scenario is removed from the Missing Resources pane.

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.

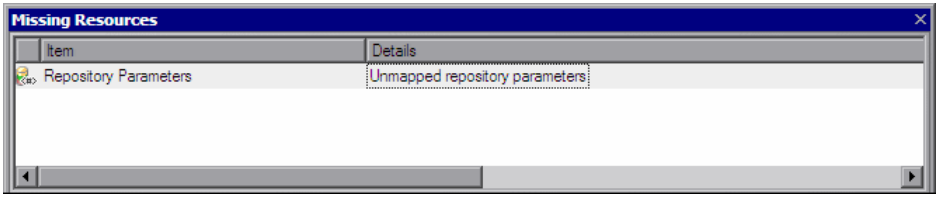
To remove a missing recovery scenario, in the Missing Resources pane, right-click the recovery scenario you want to remove and select **Remove** from the context-sensitive menu. A confirmation dialog is displayed. Click **OK** to remove the recovery scenario. The missing recovery scenario is removed 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

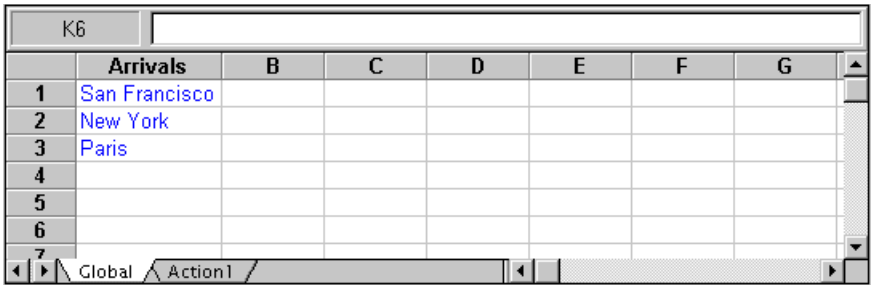
When working with tests, the Data Table has two types of data sheets—**Global** and **Action**. You can access the different sheets by clicking the appropriate tabs below the Data Table.

- 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.

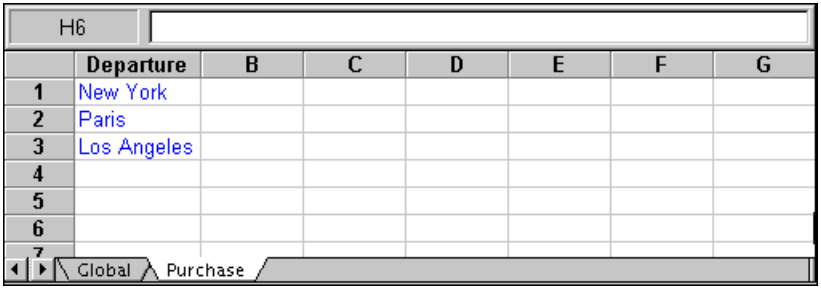
Global Sheet

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.

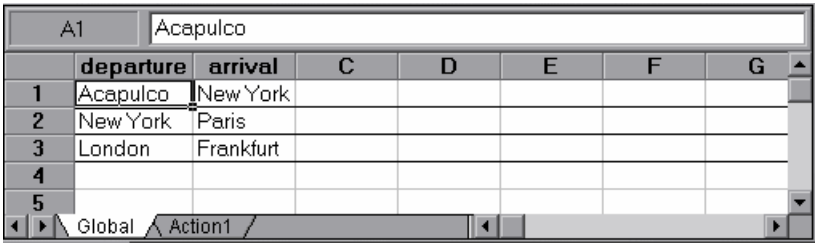
Note: If you select an external file as your Data Table, you must make sure that the column names in the external Data Table match the parameter names in the test and that the sheets in the external Data Table match the action names in the test.

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 (%).

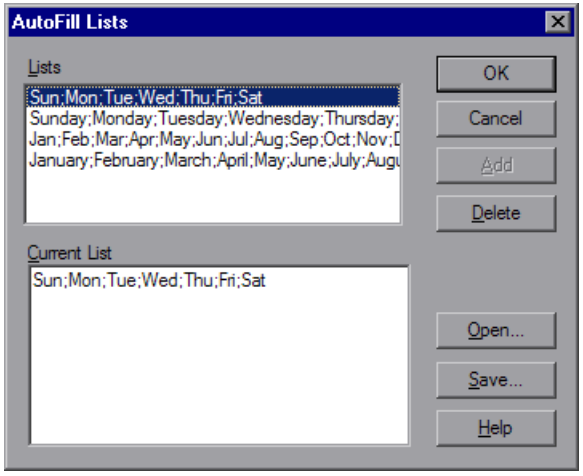
Format Command	Description
Fraction	Sets format to fraction in numerator denominator form, e.g. 1/2.
Scientific	Sets format to scientific notation with two decimal places.
date (dynamic)	Sets format to Date with the M/D/YY format.
Time: h:mm AM/ PM	Sets format to Time with the h:mm AM/PM format.
Custom 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

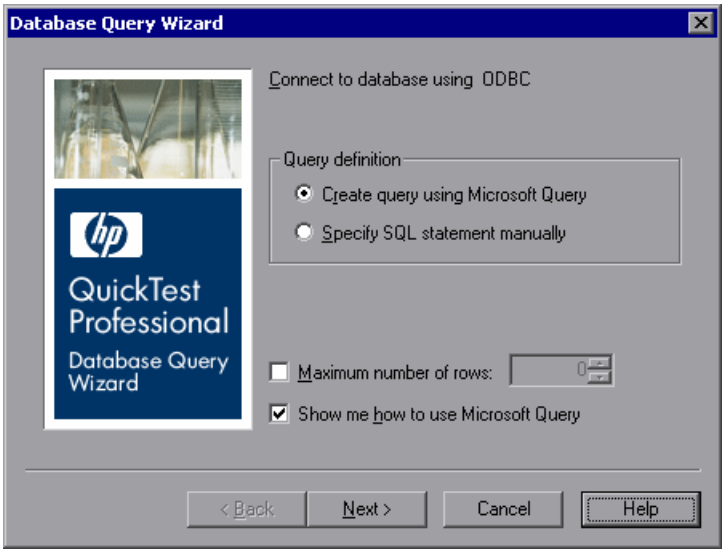
You can import data from a database by selecting a query from Microsoft Query or by manually specifying an SQL statement.

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.

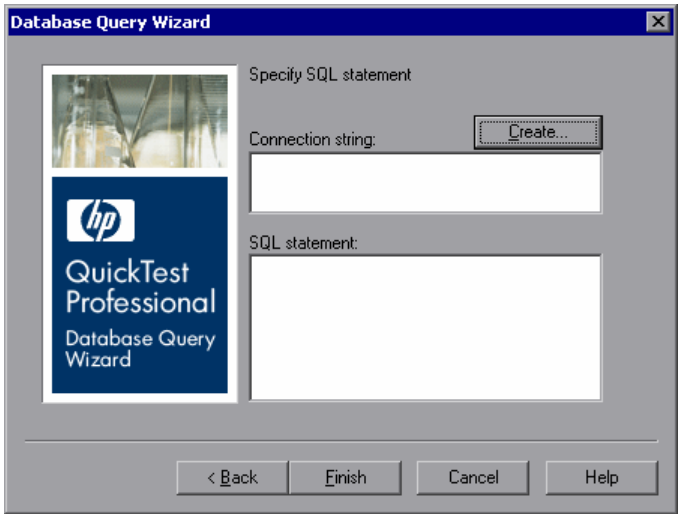


2 Select your database selection preferences and click **Next**. You can choose from the following options:

- **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.

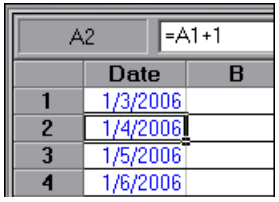
For more information on using worksheet functions, see the Microsoft Excel documentation.

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.

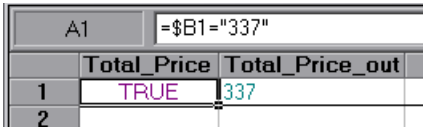


For more information on using parameters, see Chapter 24, "Parameterizing Values."

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.



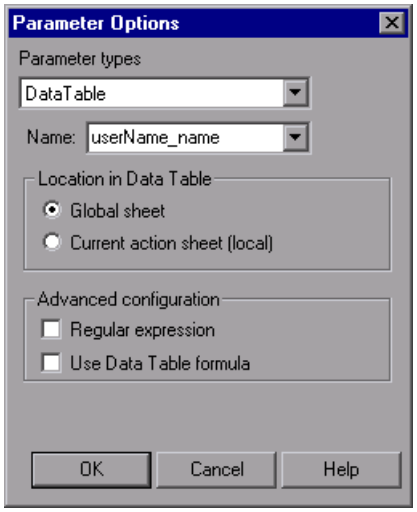
A FALSE result in the checkpoint column during a test run causes the test to fail.

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.

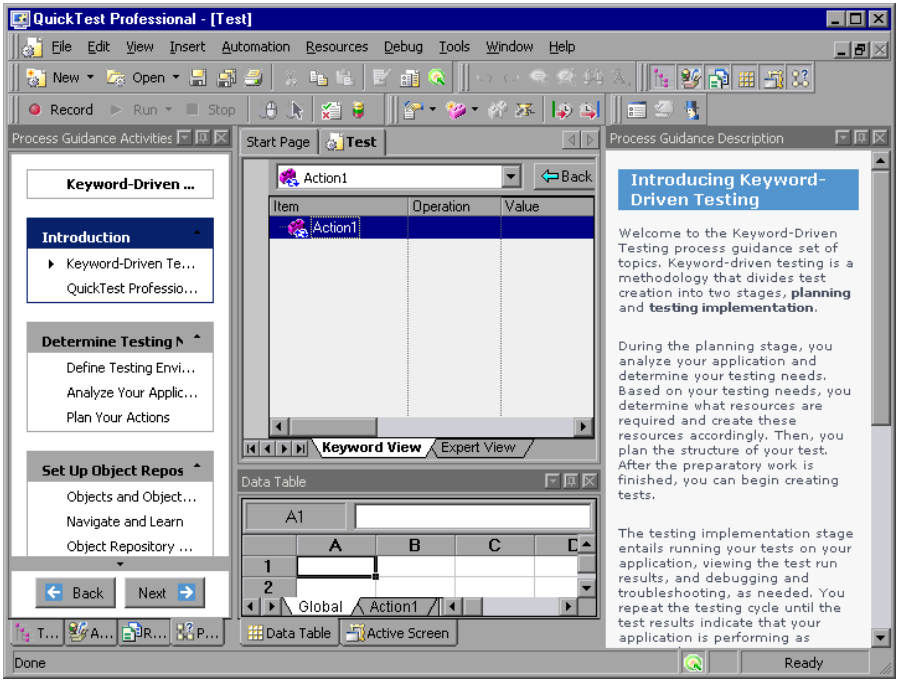
Your organization can also create its own custom processes to guide users through specific requirements and best practices relevant to your organization. For more information, see "Creating Custom Process Guidance Packages" on page 1569.

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.

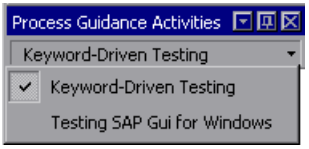
Automation Menu Command

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

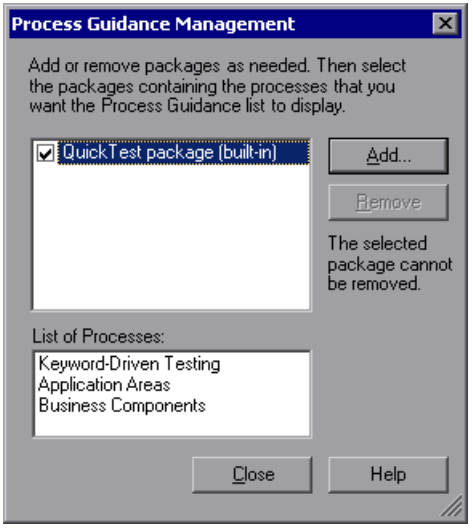
Processes are stored in process guidance packages. QuickTest provides a built-in package containing several processes. This package is listed by default in the Process Guidance Management dialog box.

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.

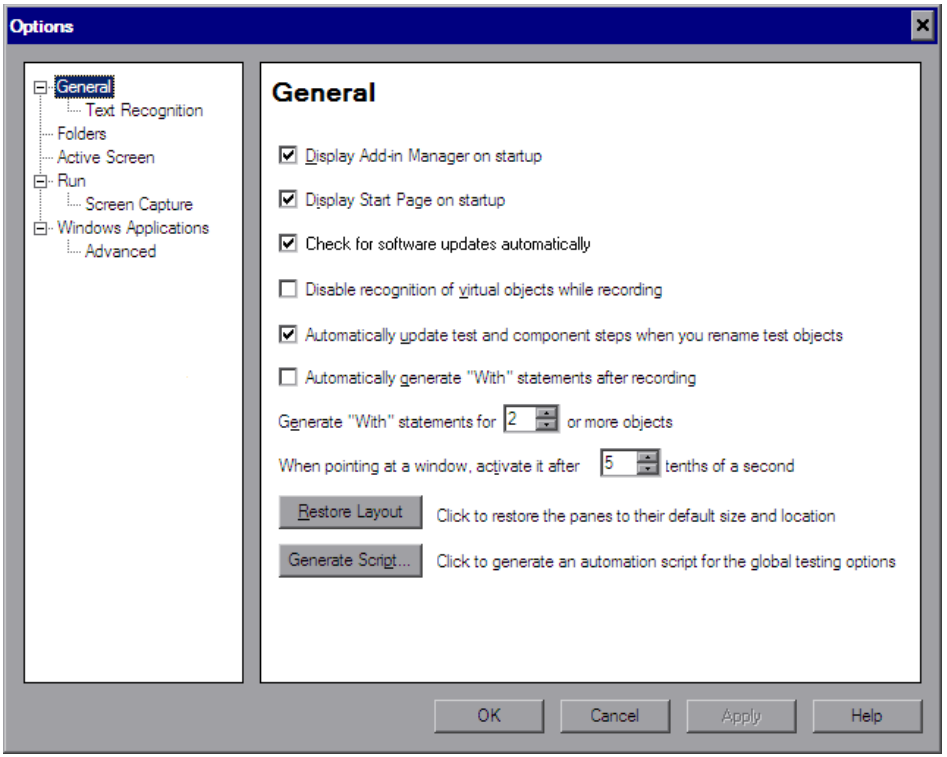
You can also set testing options that affect only the test currently open in QuickTest. For more information, see Chapter 45, "Setting Options for Individual Tests."

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.

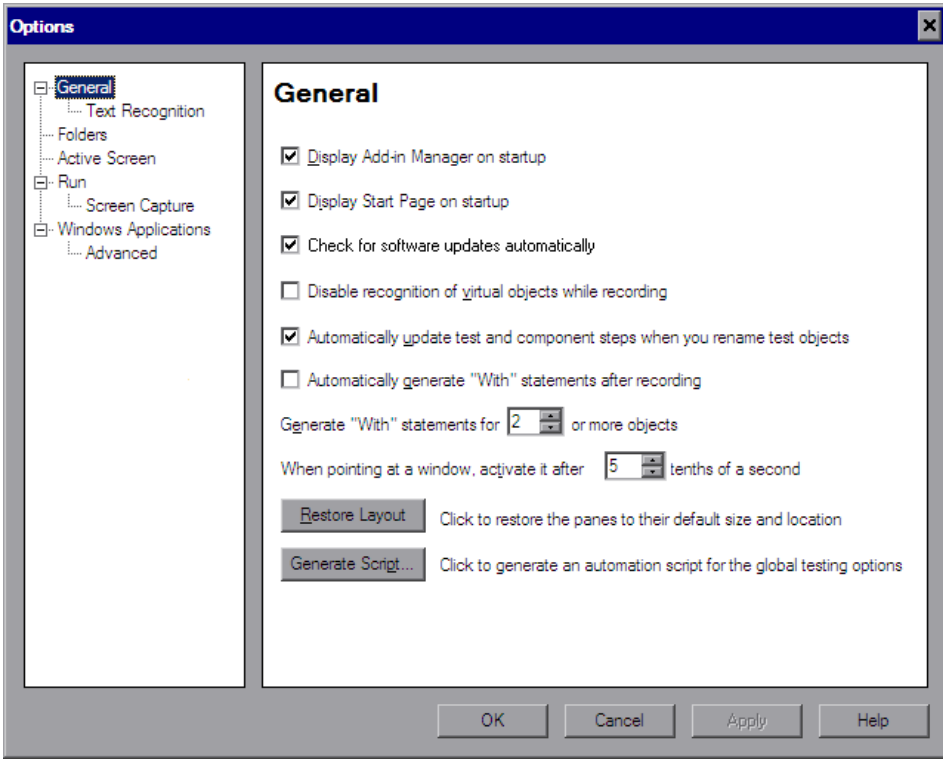
The navigation tree contains the following nodes:

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.

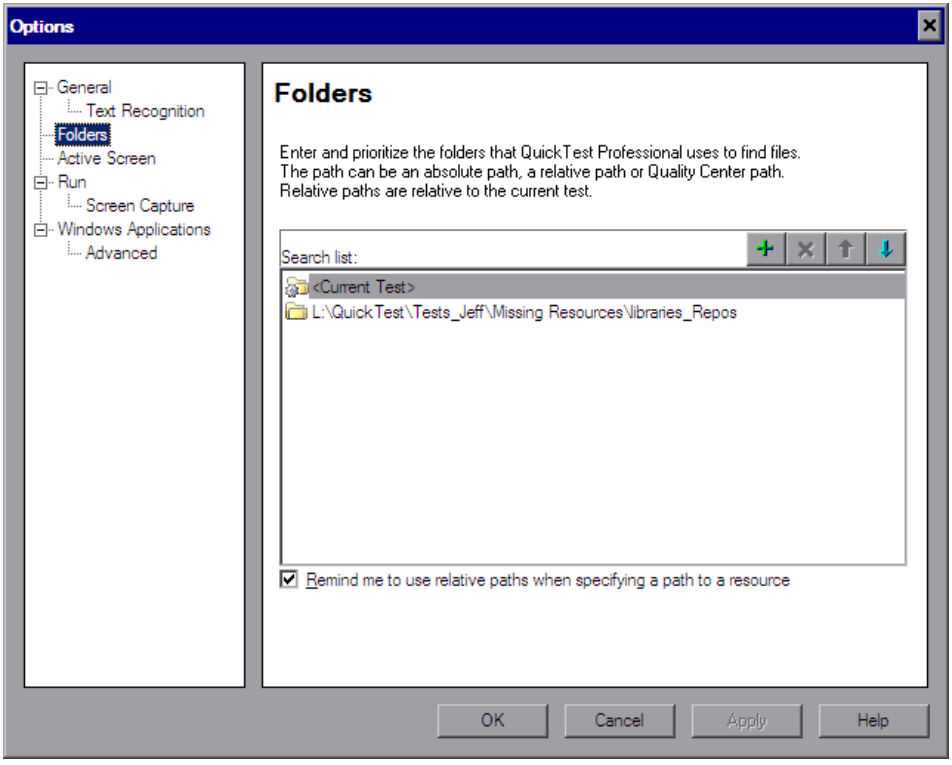
Setting Folder Testing Options

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:

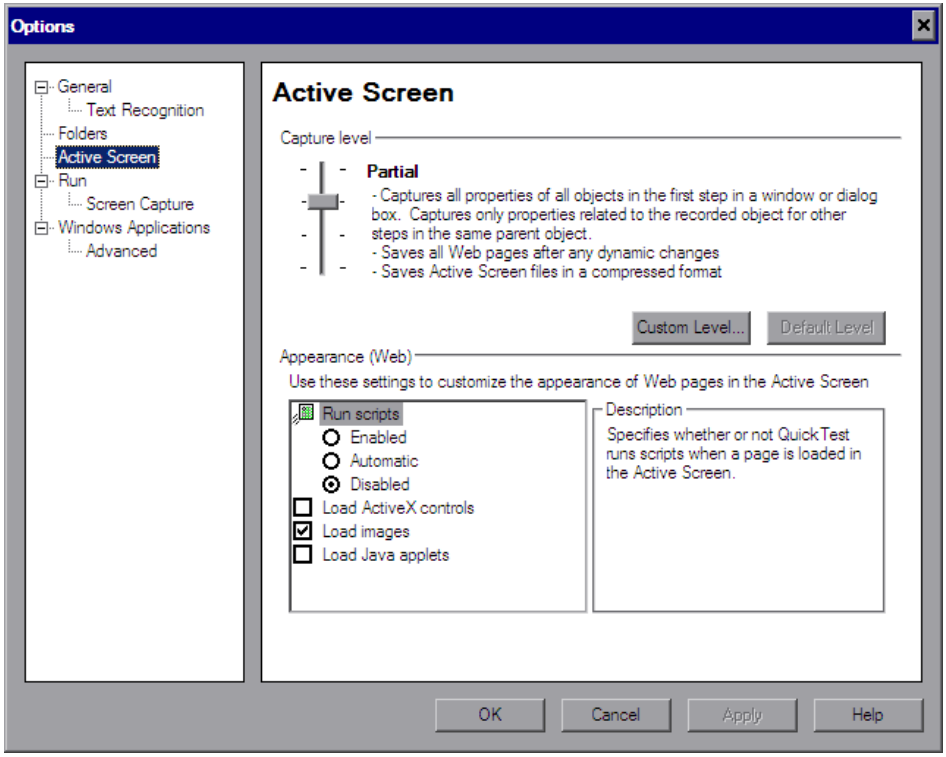
Option	Description
Search list	Indicates the folders in which QuickTest searches for tests, actions, or files. If you define folders here, you do not need to designate the full path of a test, action, or file in other dialog boxes or call statements. The order of the search paths in the list determines 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

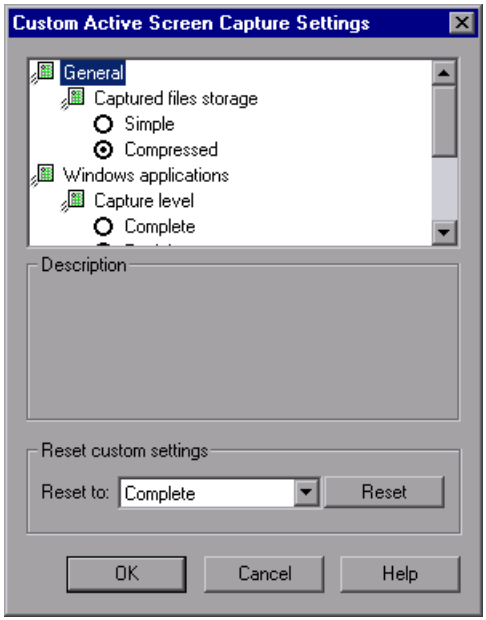
Related User Interface Topics	"The Custom Active Screen Capture Settings Dialog Box" on page 1244
Related Tasks	➤ "Updating a Single Active Screen Capture" on page 380 ➤ "Updating all Active Screen Captures in a Test Using Update Run Mode" on page 1125
Other Related Information	"Tips for Improving Active 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.

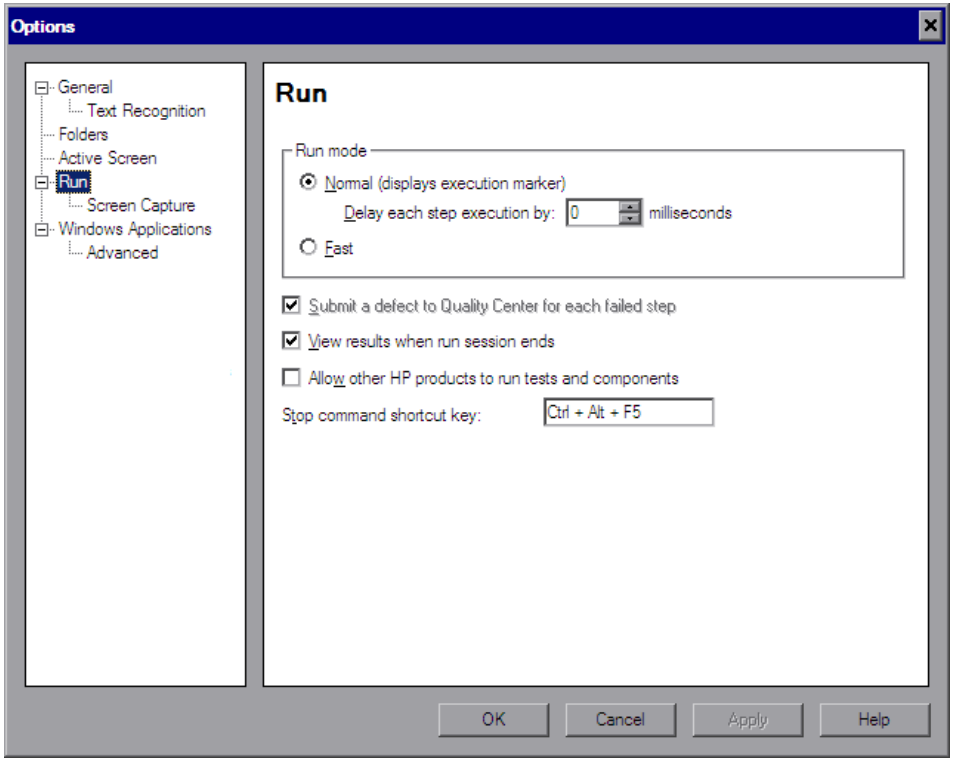
- **Disable Active Screen capture.** Disables the screen capture of all steps in the Active Screen.

  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.

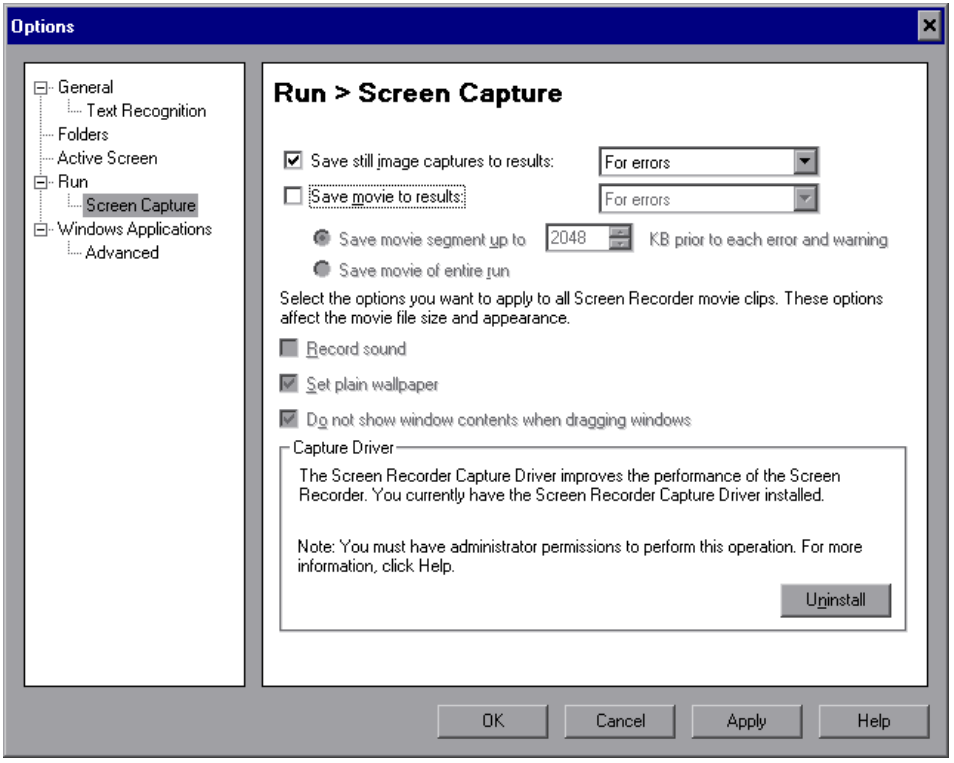
Option	Description
Allow other HP products to run tests and components	Enables other HP products such as Quality Center and Test Batch Runner to run QuickTest tests on this computer. **Note:** This option is not required to enable WinRunner to run QuickTest tests.
Stop command shortcut key	Enables you 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:

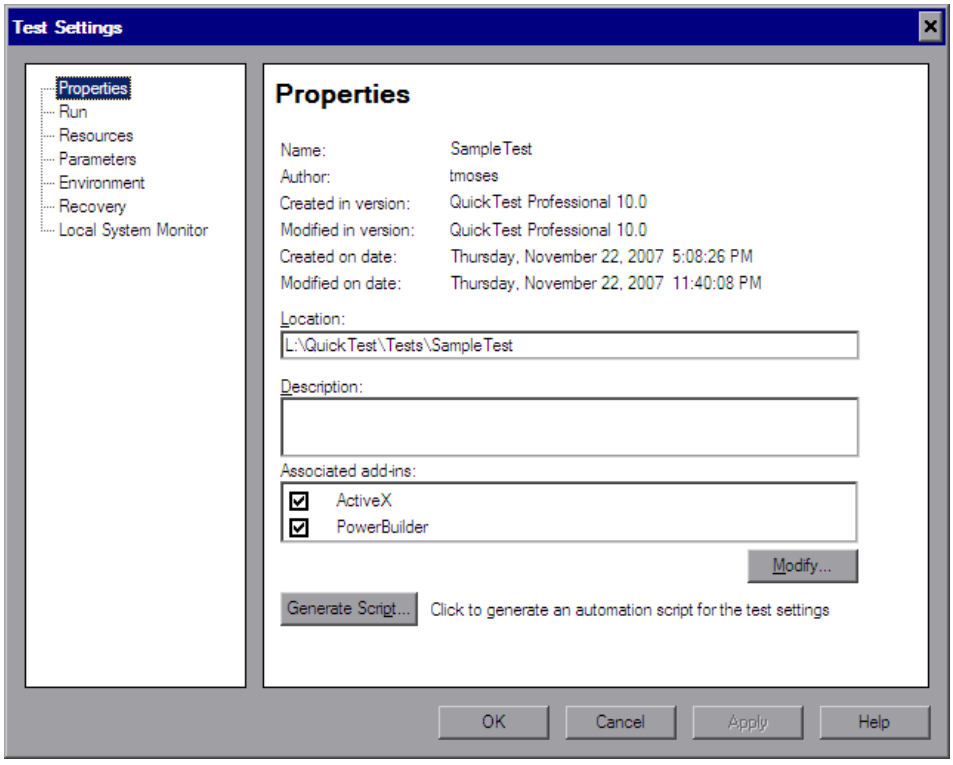
Using the Test Settings Dialog Box

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.

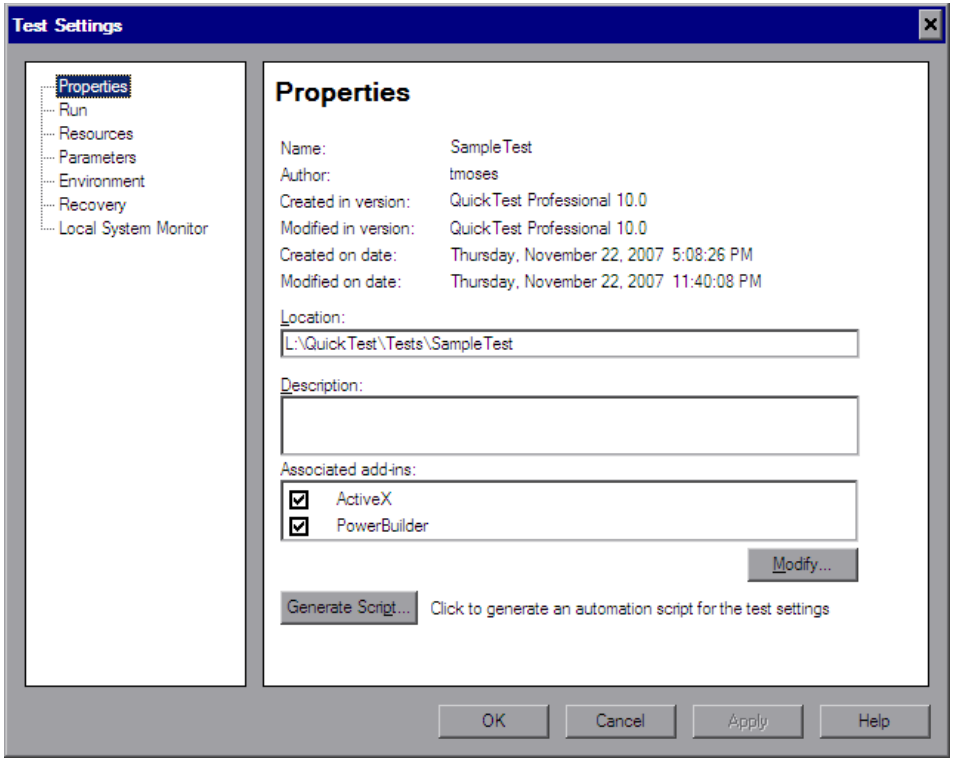
The navigation tree contains the following nodes:

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.

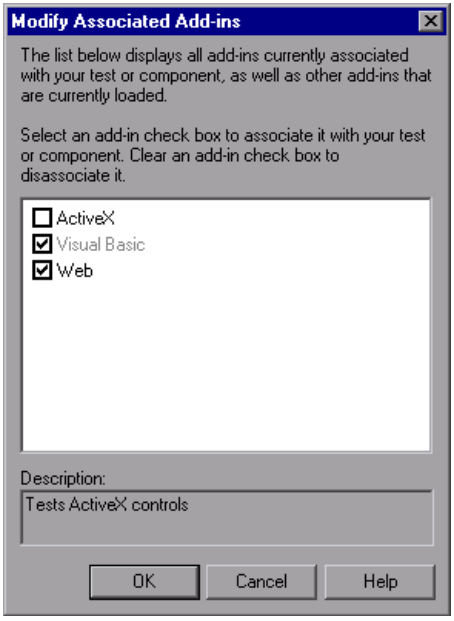
Choosing to associate an add-in with your test instructs QuickTest to check that the associated add-in is loaded each time you open that test.

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."

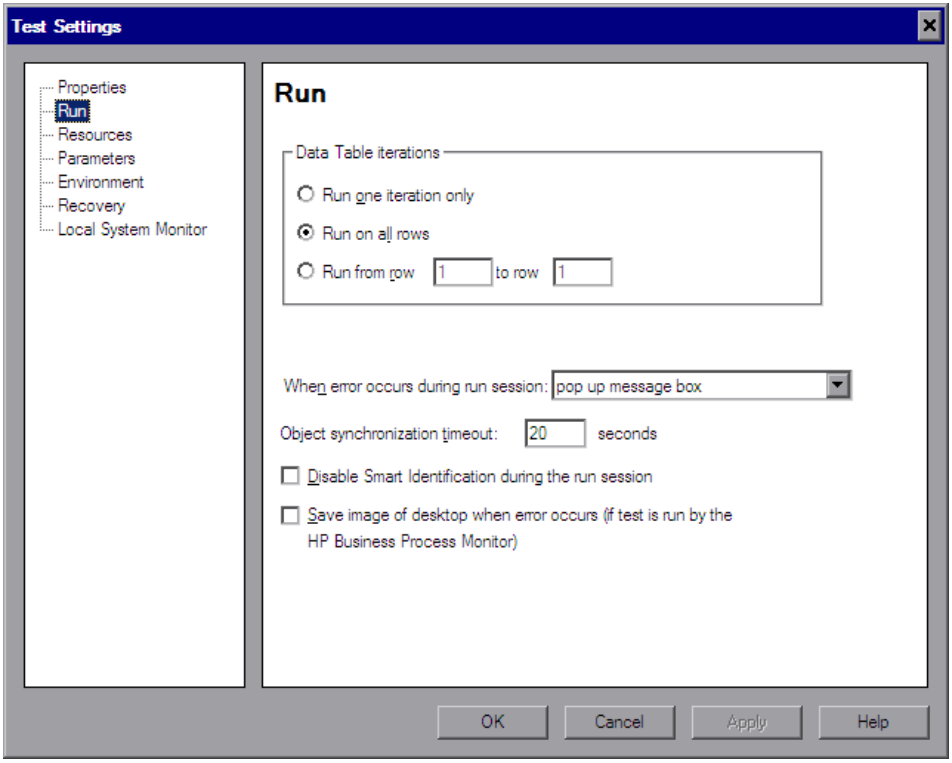
For more information on the Add-in Manager, see the section on working with QuickTest add-ins in the *HP QuickTest Professional Add-ins Guide*.

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.

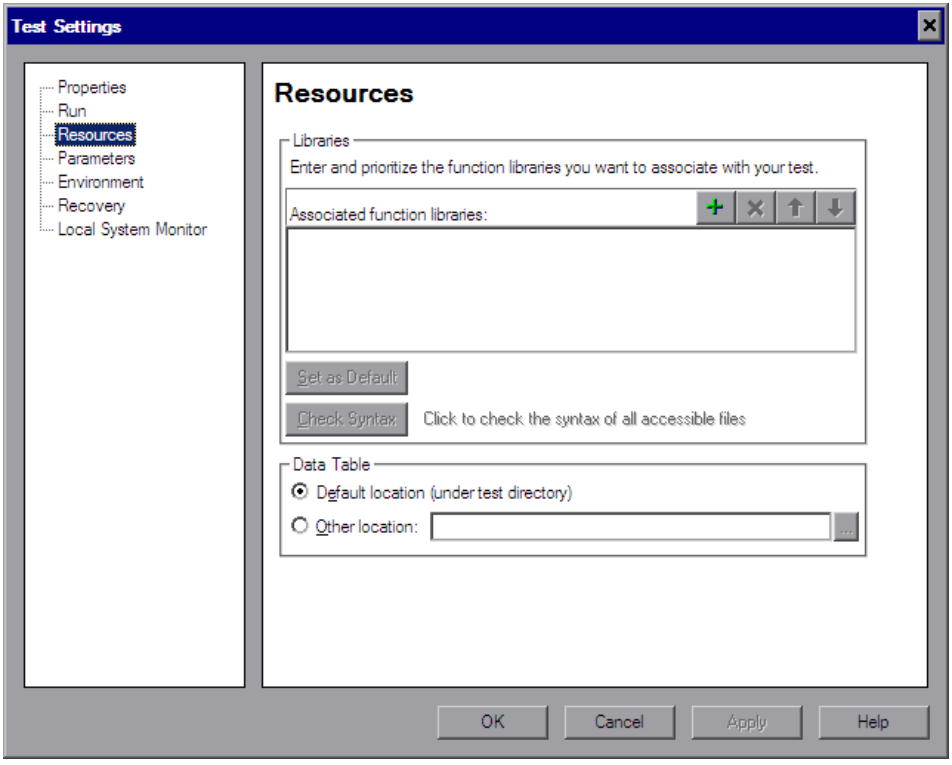
QuickTest first performs any recovery scenarios associated with the test, and performs the option selected above only if the associated recovery scenarios do not resolve the error. For more information, see "Defining Recovery Scenario Settings for Your Test" on page 1291.

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.

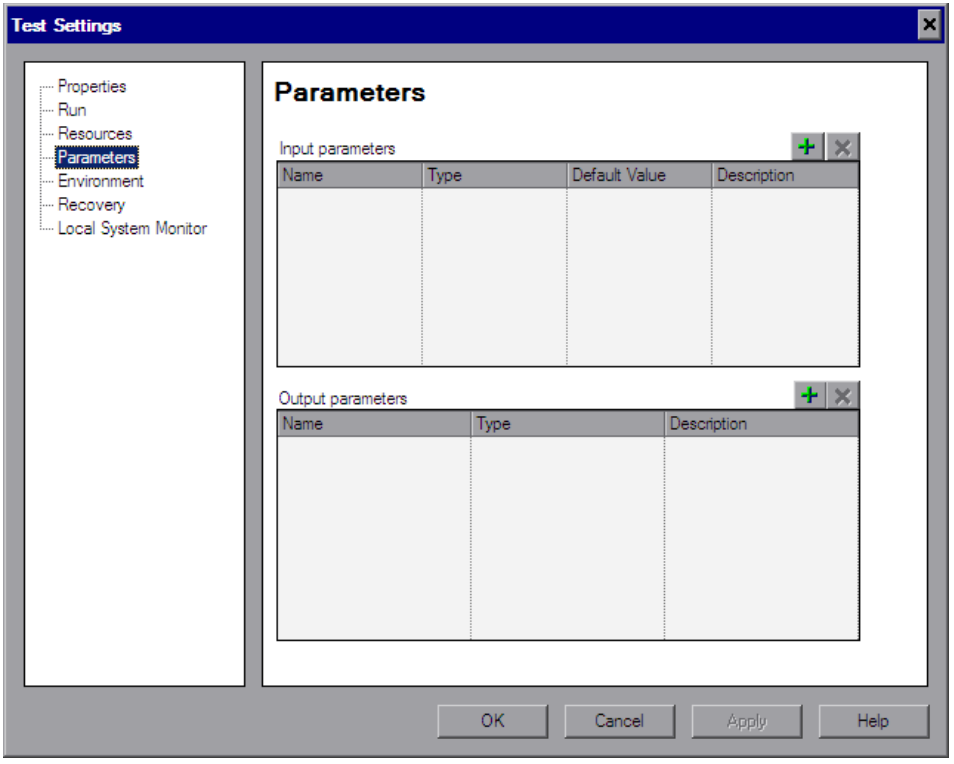
Note: When running a test, QuickTest uses associated function libraries from Quality Center project folders only when you are connected to the corresponding Quality Center project.

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.

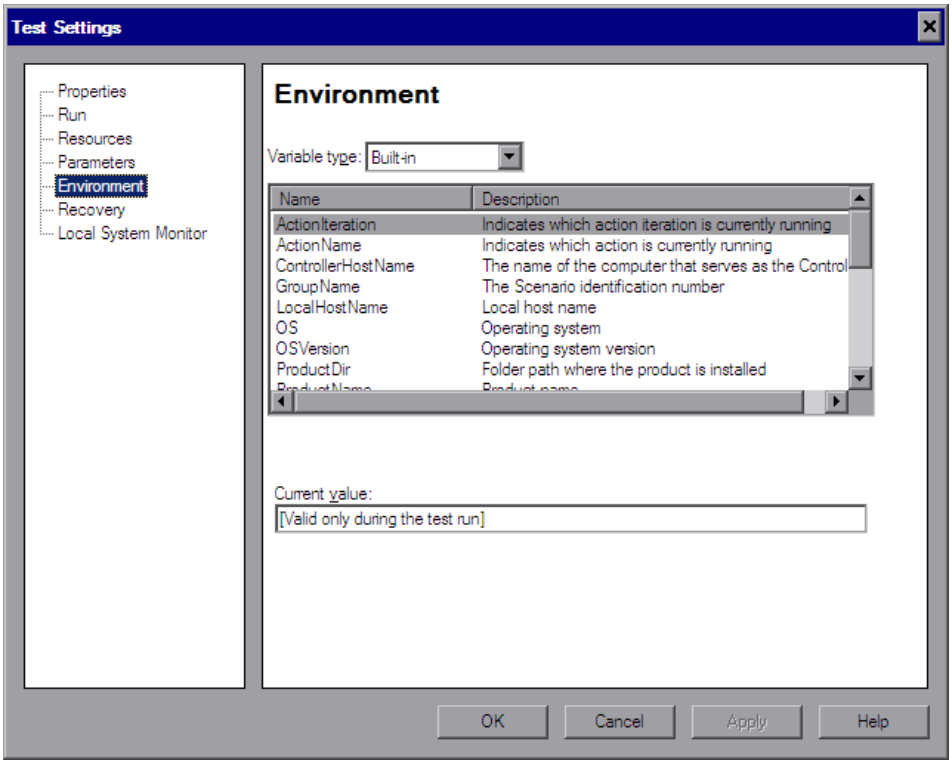
For more information on environment variables and environment parameters, see "Using Environment Variable Parameters" on page 645.

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.

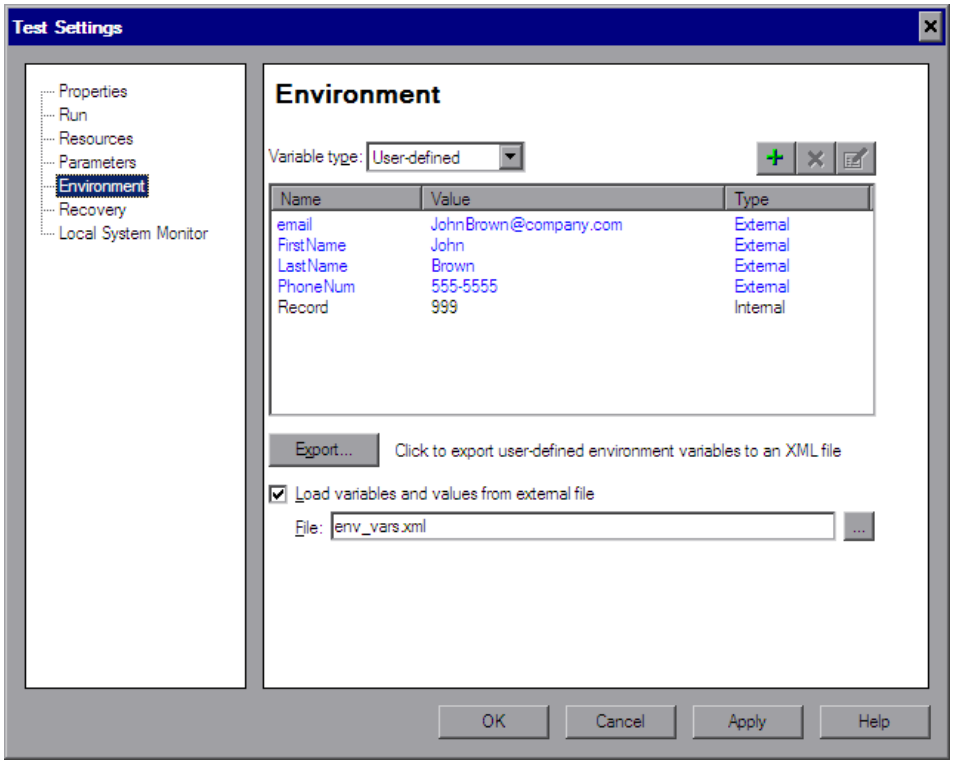


The following information is displayed for built-in environment variables:

- **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.

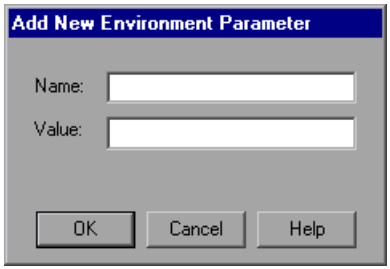
Adding User-Defined Environment Variables

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.

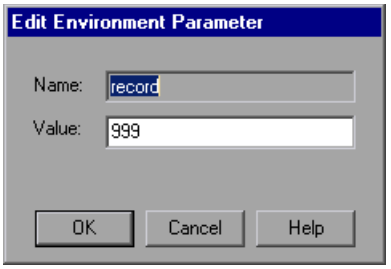
Viewing and Modifying User-Defined Environment Variables

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

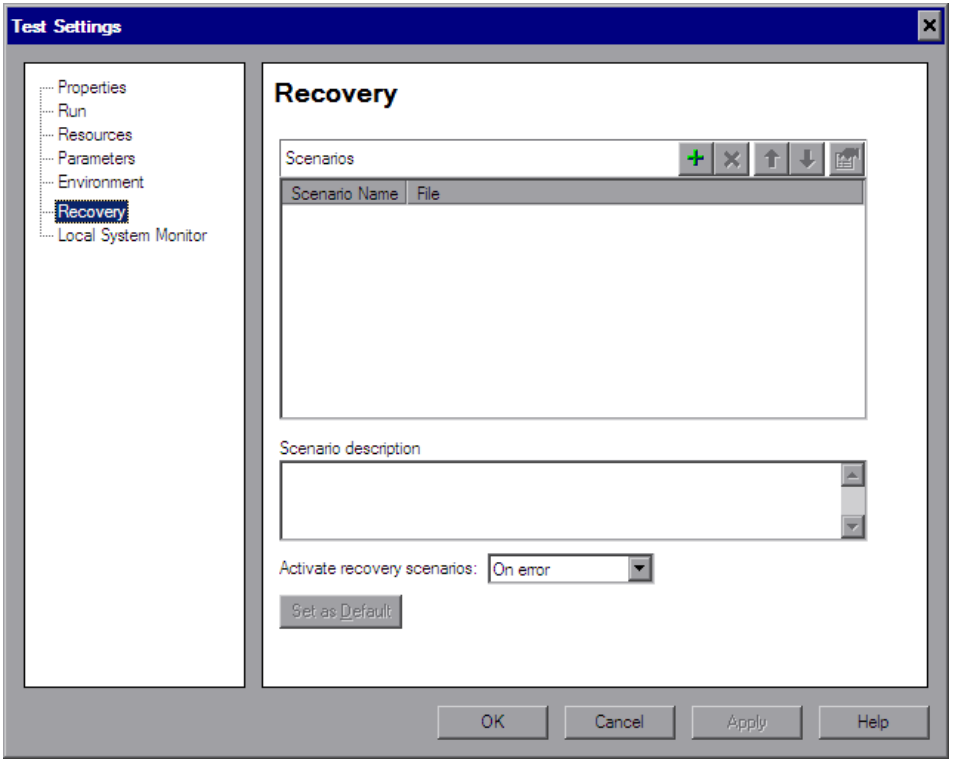
The Recovery pane of the Test Settings dialog box (**File** > **Settings** > **Recovery** node) displays a list of all recovery scenarios associated with the current test. It also enables you to associate additional recovery scenarios with the test, remove scenarios from the test, change the order in 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.

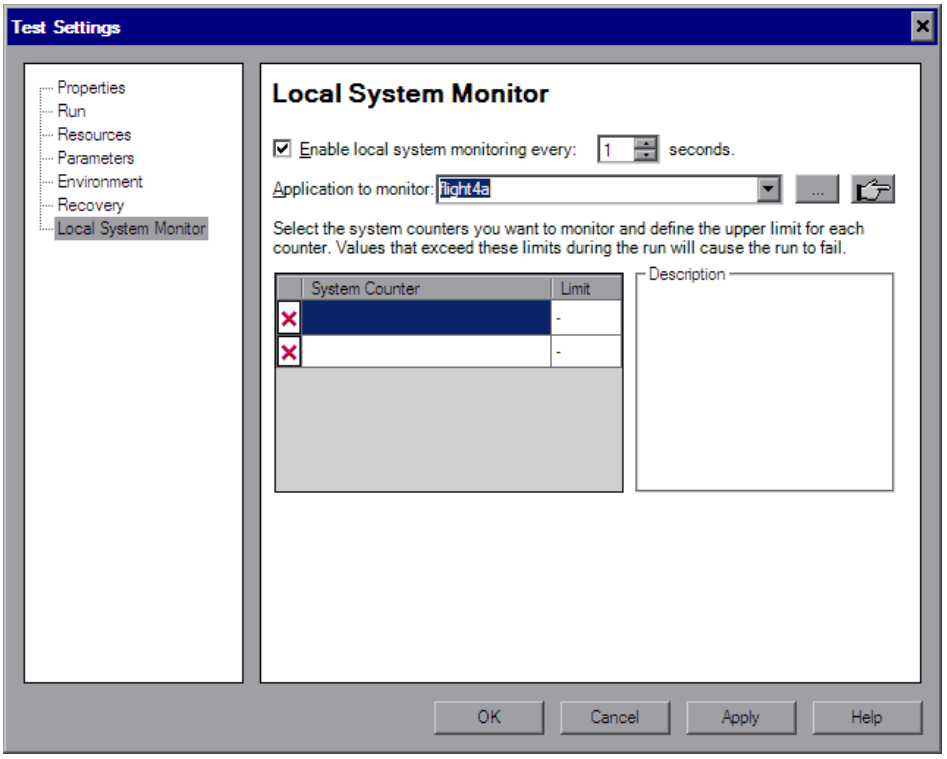
The system counters that can be monitored are the process counters accessible through the Performance Console (Select **Start** > **Run** > and then enter Perfmon). For information on the process counters accessible through the Performance Console, see the Performance Console Help.

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

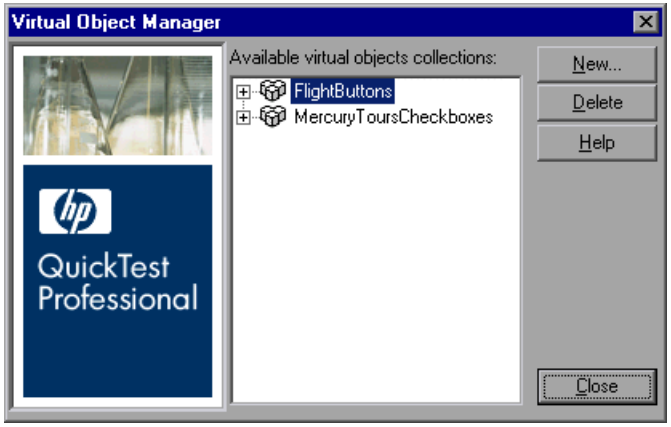
The Virtual Object Manager enables you to view and manage the virtual object collections defined on your computer. From the Virtual Object Manager, you can define and delete virtual objects and collections.

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.

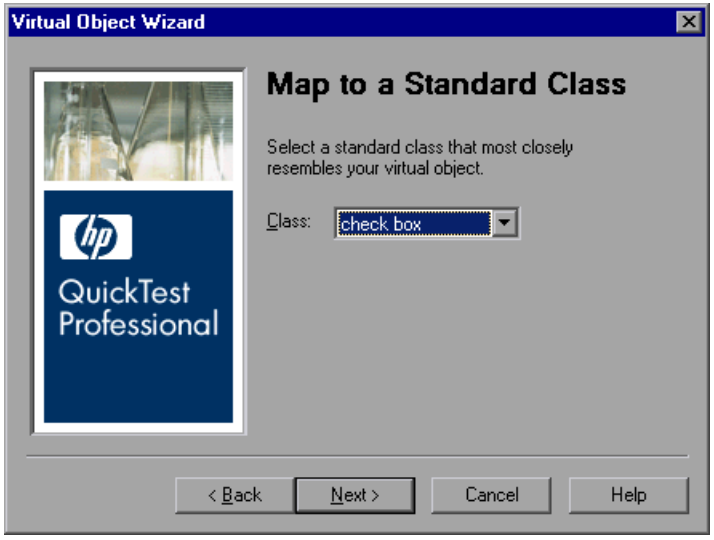
Additional References

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.

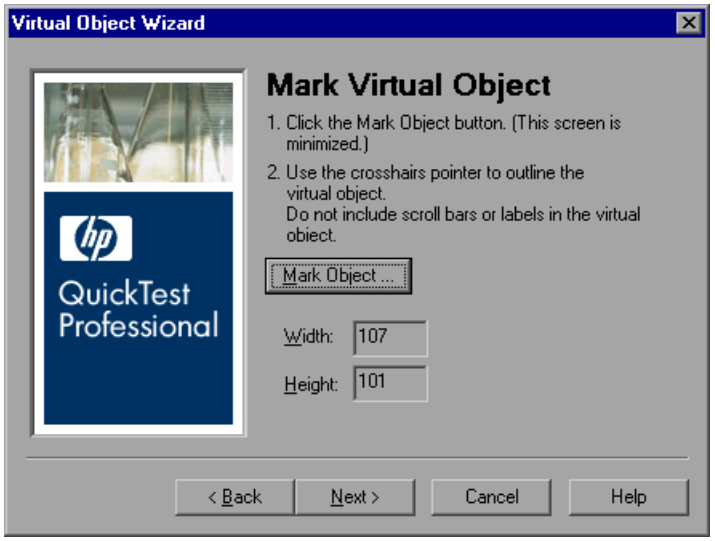
Additional References

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.

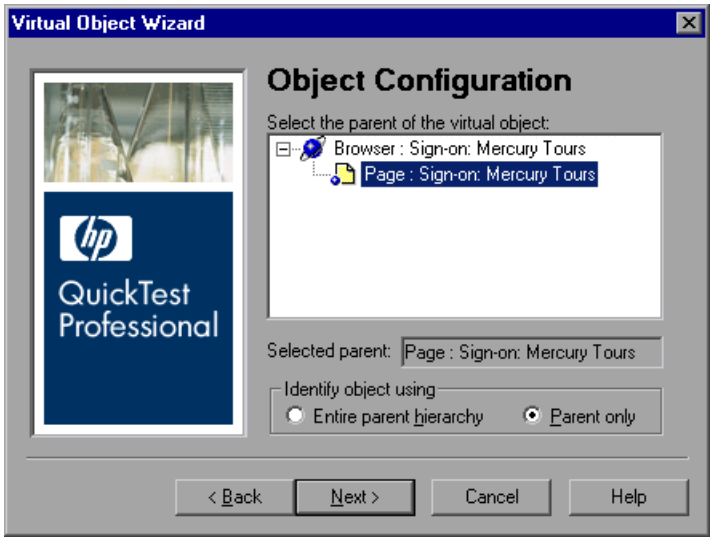
Additional References

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.

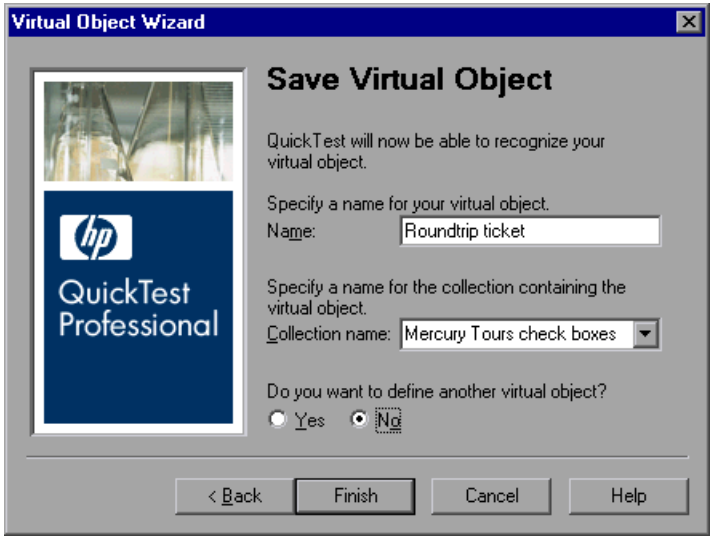
Additional References

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.

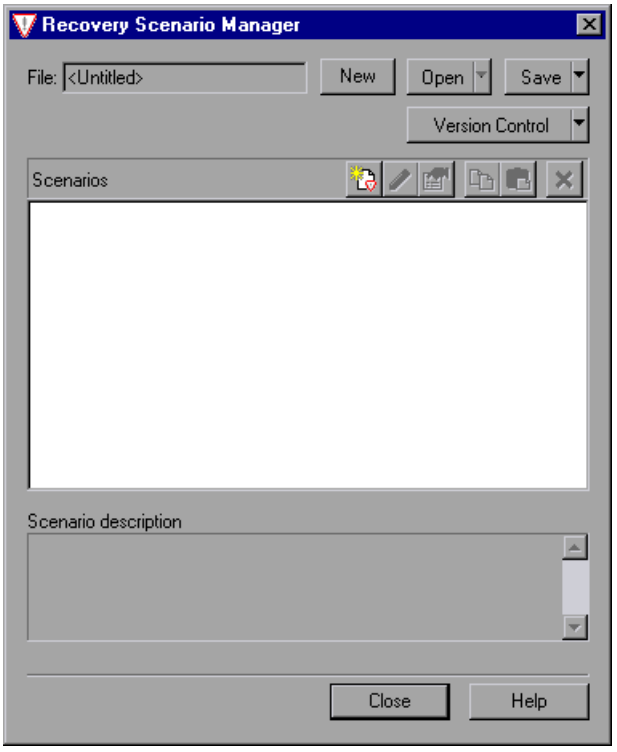
Using the Recovery Scenario Manager dialog box, you can then select any recovery file to manage all of the recovery scenarios stored in that file. This enables you to edit a selected recovery scenario, associate specific recovery scenarios with specific tests to instruct QuickTest to implement the recovery 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.

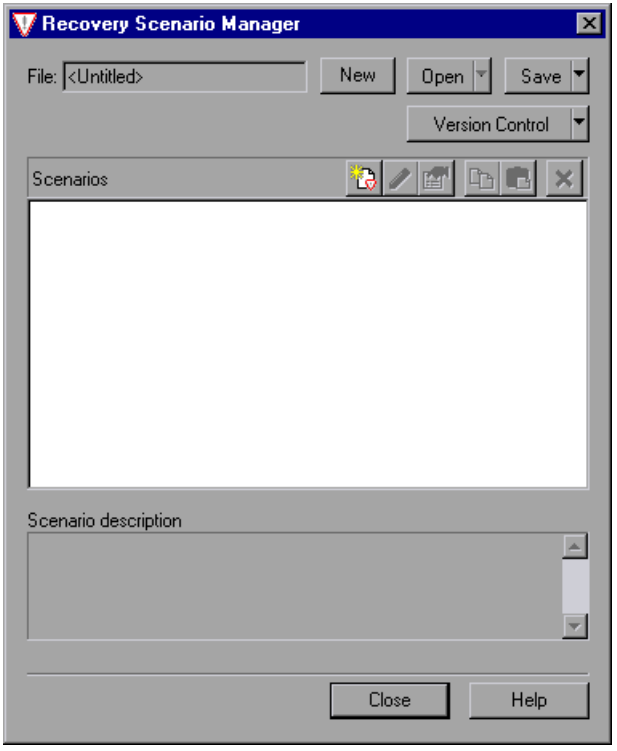
c Browse to and select the recovery scenario file you want to open and click **Open**. If the recovery file is stored in a version-control-enabled project in Quality Center, you can click the **Open** down arrow and select **Open and Check out** to check out the file.

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

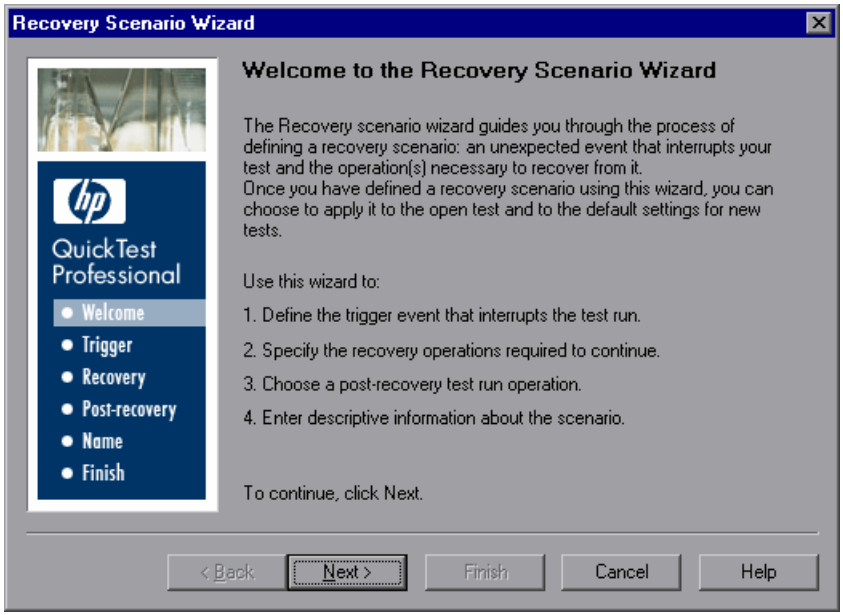
The Recovery Scenario Wizard leads you, step-by-step, through the process of creating a recovery scenario. The Recovery Scenario Wizard contains the following main steps:

- 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

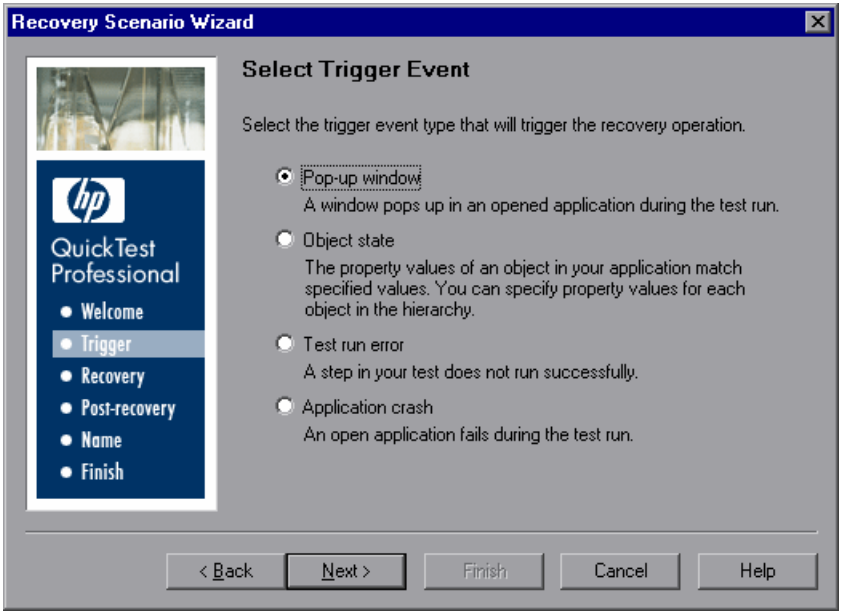
The Welcome to the Recovery Scenario Wizard screen provides general information on the different options in the Recovery Scenario Wizard, and provides an overview of the stages involved in defining a recovery scenario.



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.

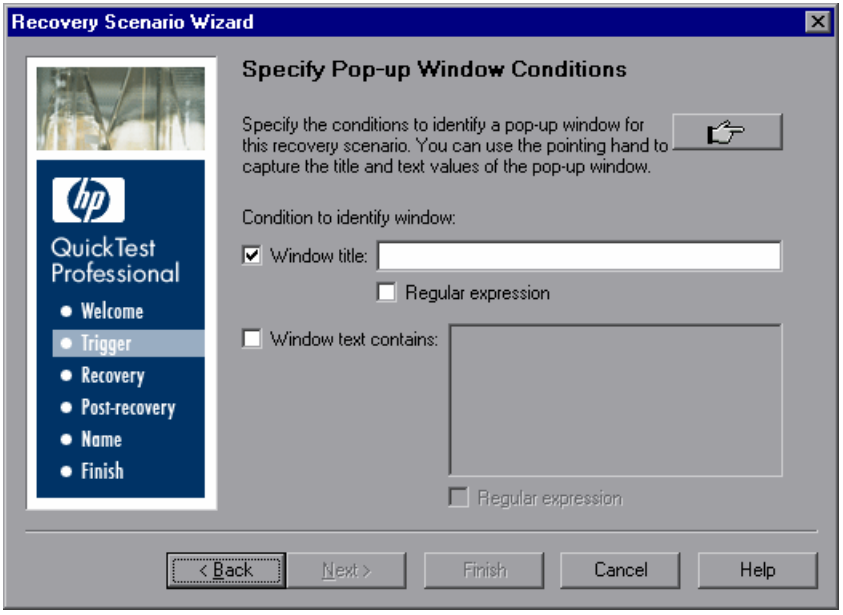
Select this option and click **Next** to continue to the Recovery Operations Screen (described on page 1352).

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.

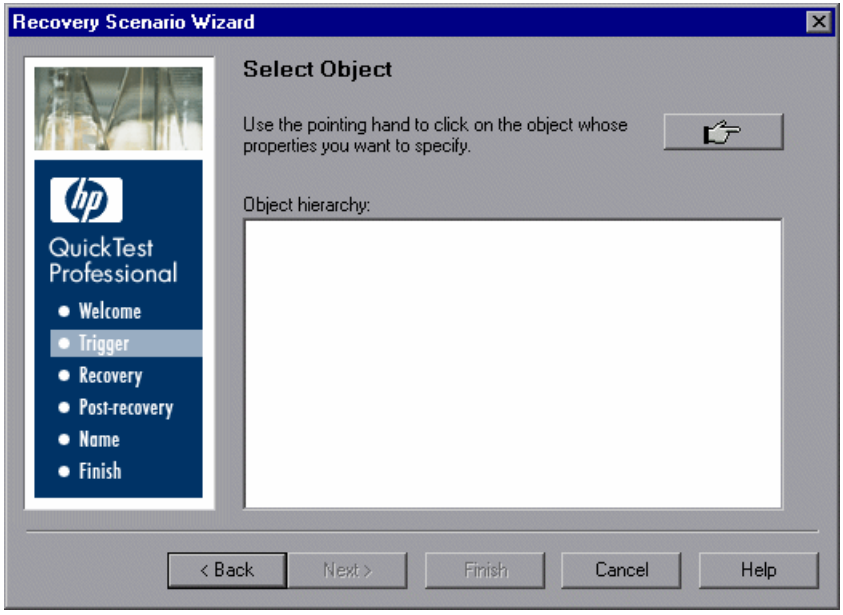
Click **Next** to continue to the Recovery Operations Screen (described on page 1352).

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.

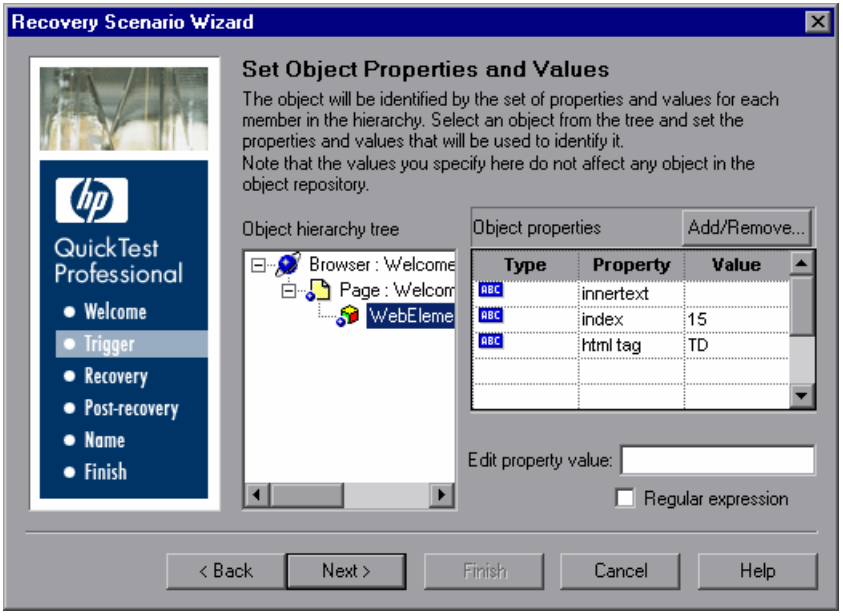
Click **Next** to continue to the Set Object Properties and Values Screen (described on page 1348).

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.



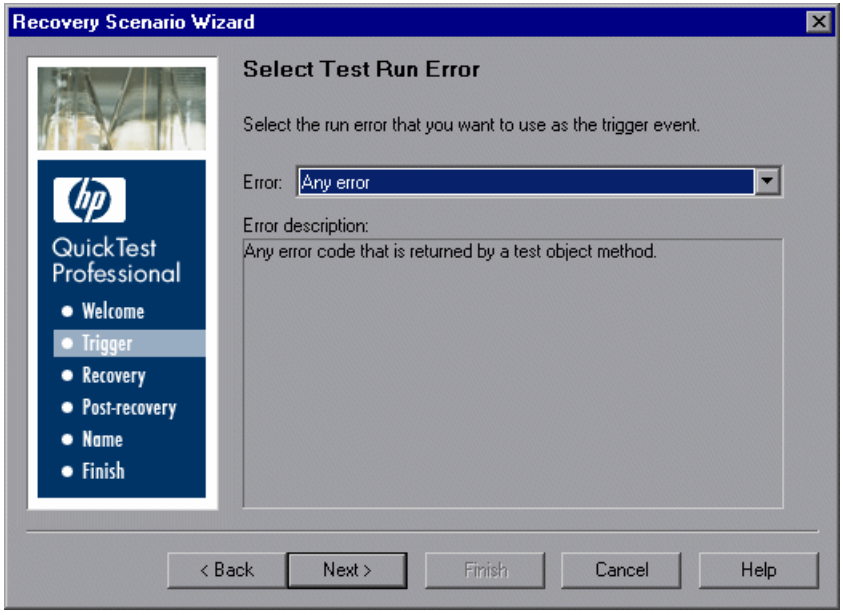
For each object in the hierarchy, in the **Edit property value** box, you can modify the property values used to identify the object. You can also click the **Add/Remove** button to add or remove object properties from the list of property values to check. Note that an object is identified only by its 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:

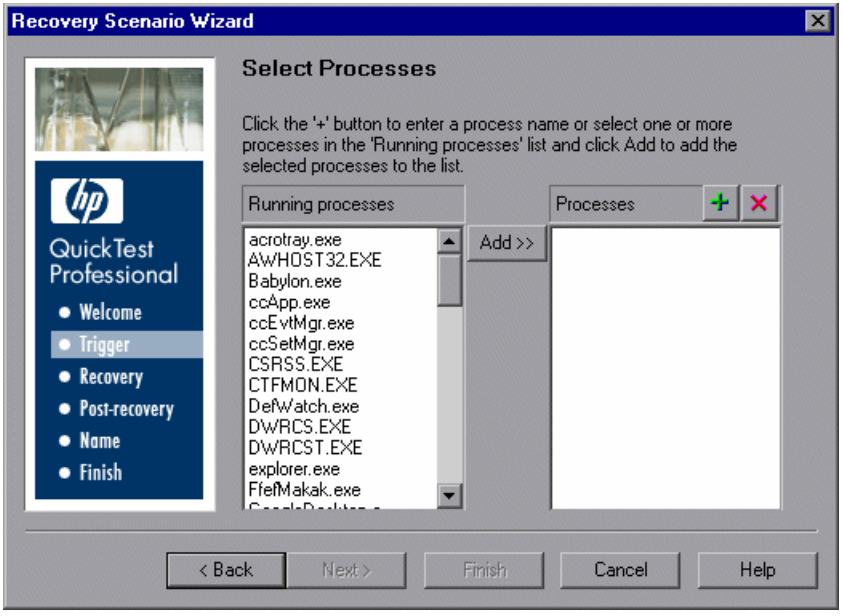
- **Any error.** Any error code that is returned by a test object method.
- **Item in list or menu is not unique.** Occurs when more than one item in the list, menu, or tree has the name specified in the method argument.
- **Item in list or menu not found.** Occurs when QuickTest cannot identify the 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.

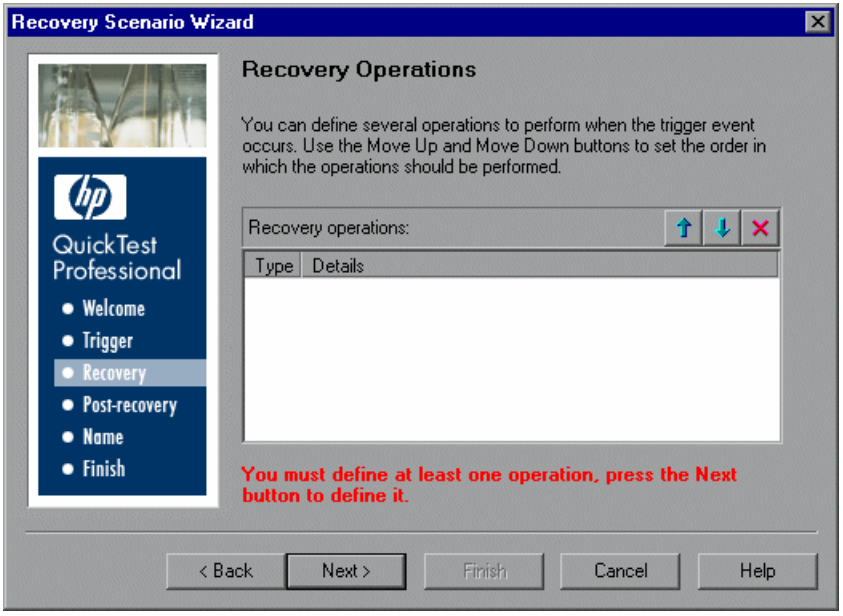
- To remove a process from the **Processes** list, select it and click the **Remove Process** button.

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.

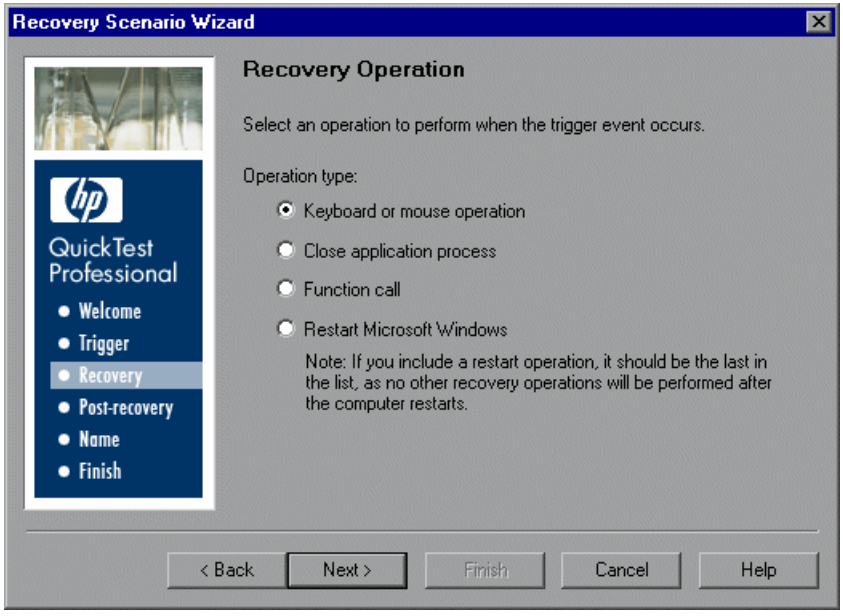
After you have defined at least one recovery operation, the **Add another recovery operation** check box is displayed.

➤ 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.

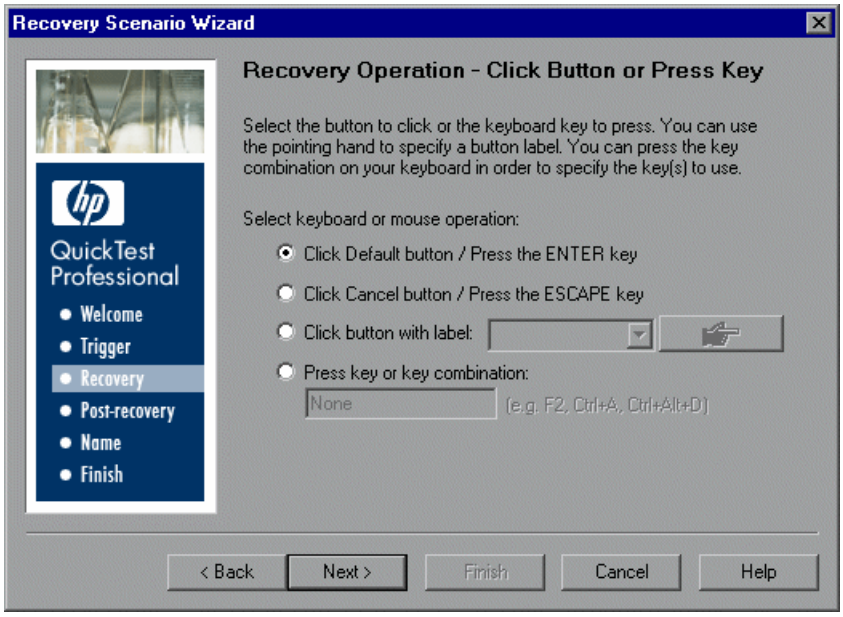
You can define the following types of recovery operations:

- **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.

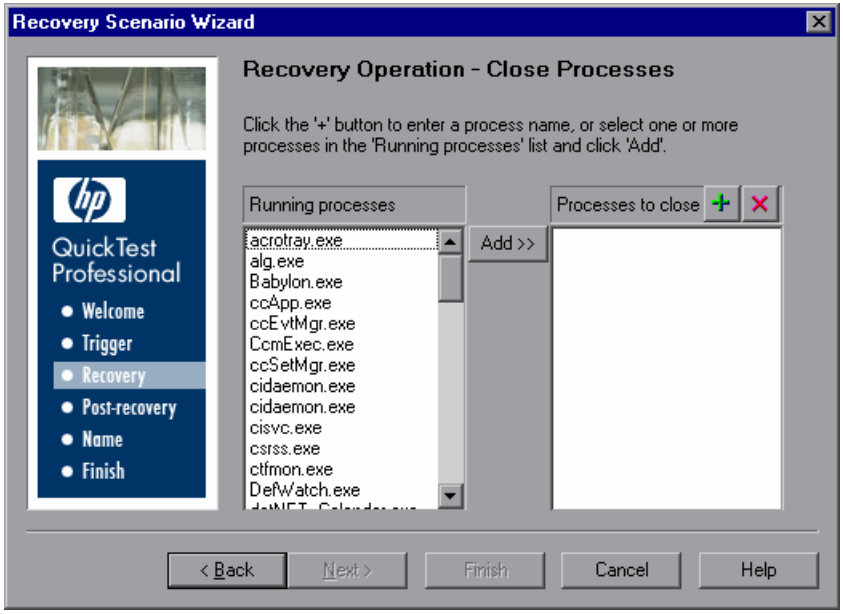
Tips for Using the Pointing Hand

- 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.

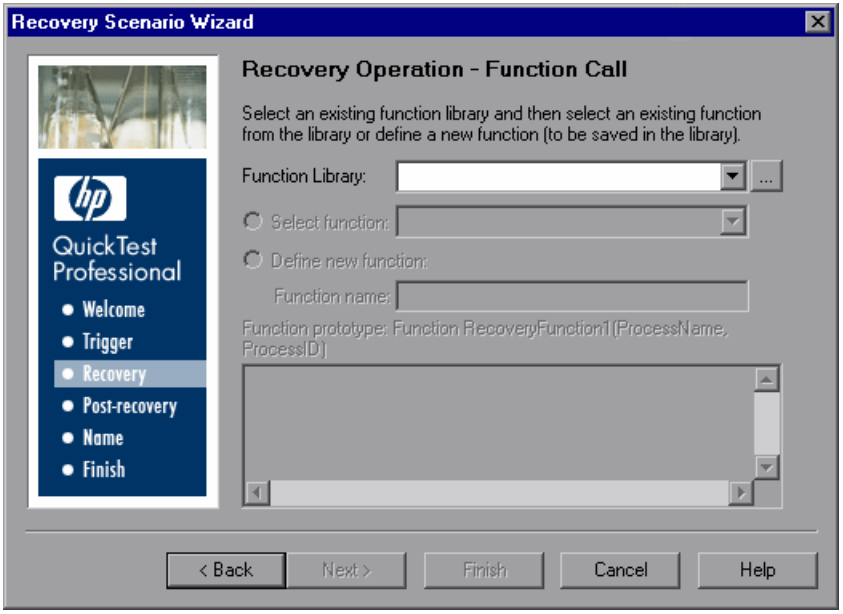
➤ To remove a process from the **Processes to close** list, select it and click the **Remove Process** button.

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.
)
```

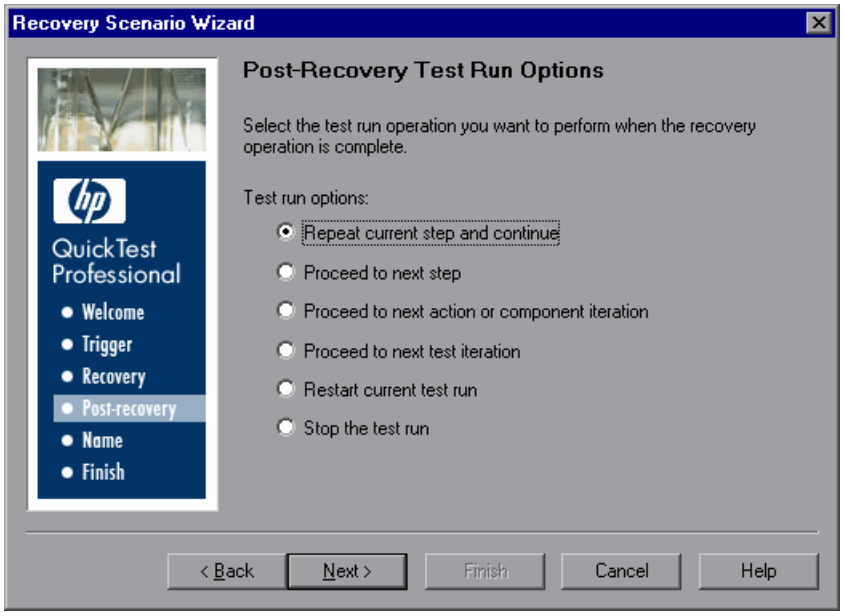
➤ **Define new function.** Create a new function by specifying a unique name for it, and defining the function in the **Function Name** box according to the displayed function prototype. The new function is added to the function library you selected.

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.

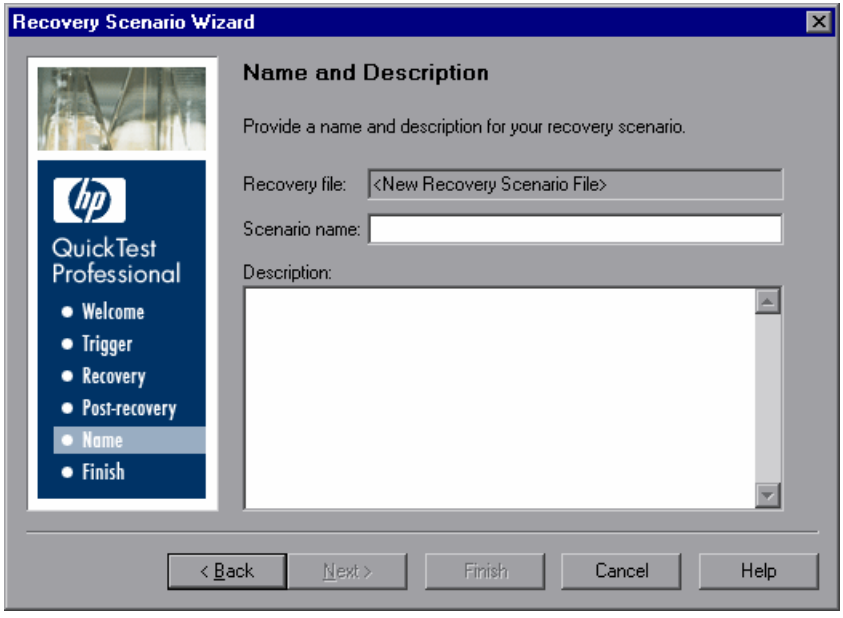
Note: If you chose **Restart Microsoft Windows** as a recovery operation, you can choose from only the last two test run options listed above.

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.

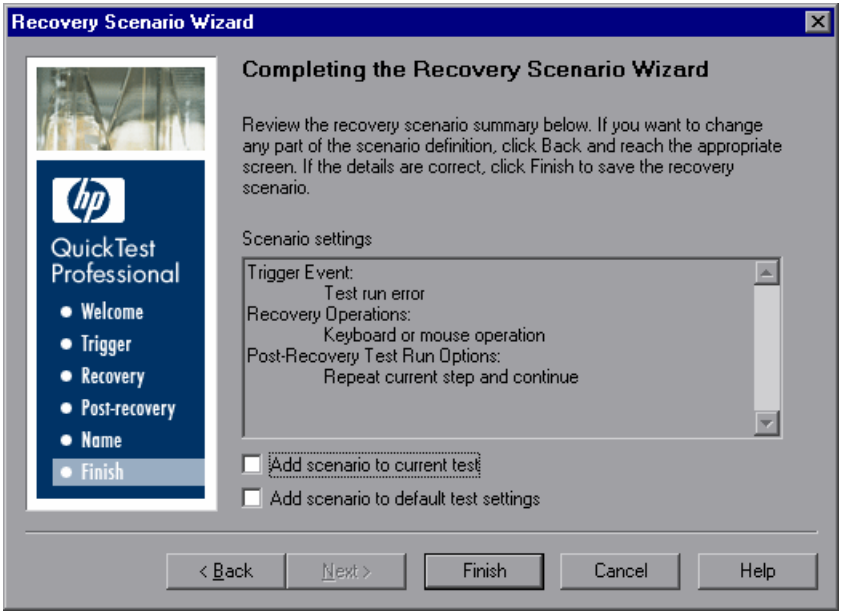


Enter a name and a textual description for your recovery scenario, and click **Next** to continue to the Completing the Recovery Scenario Wizard Screen (described on page 1364).

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**.

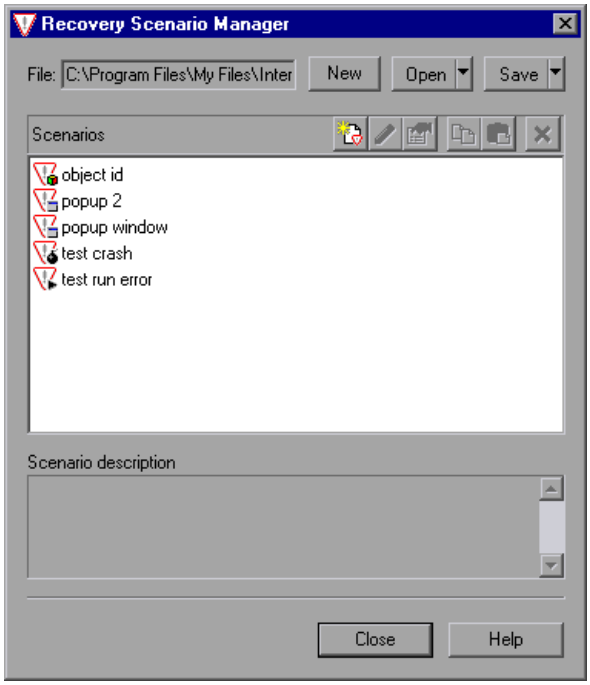
Note: When you specify a path to a resource in the file system or in Quality Center 9.x, QuickTest checks if the path, or a part of the path, exists in the Folders pane of the Options dialog box (**Tools** > **Options** > **Folders** node). If the path exists, you are prompted to define the path using 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

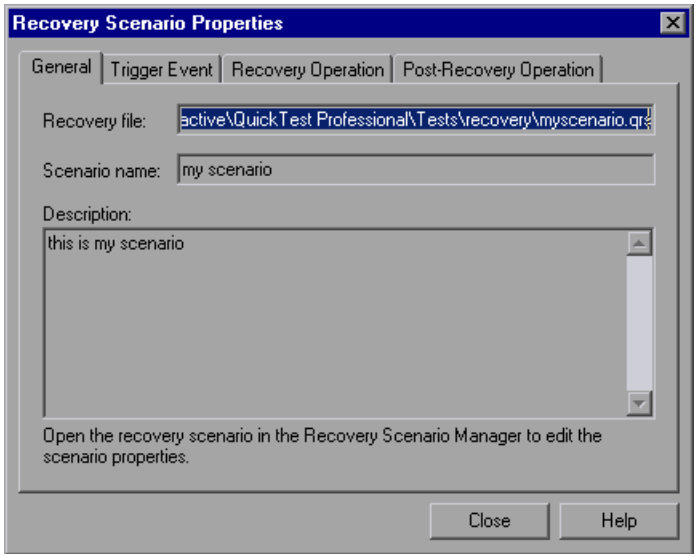
Viewing Recovery Scenario Properties

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.

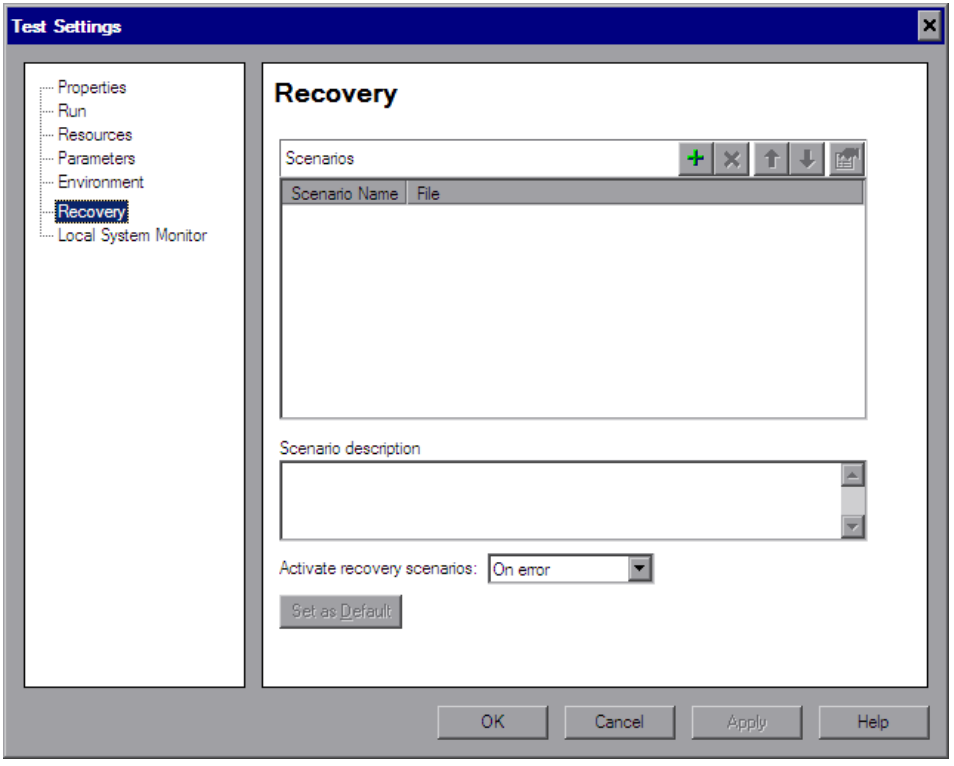
Adding Recovery Scenarios to Your Test

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.

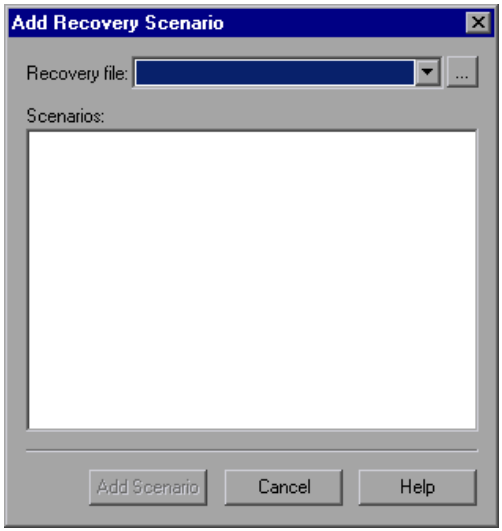
Tip: When a trigger event occurs, QuickTest checks for applicable recovery scenarios in the order in which they are displayed in the Recovery pane. You can change this order as described in "Setting Recovery Scenario Priorities" on page 1376.

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

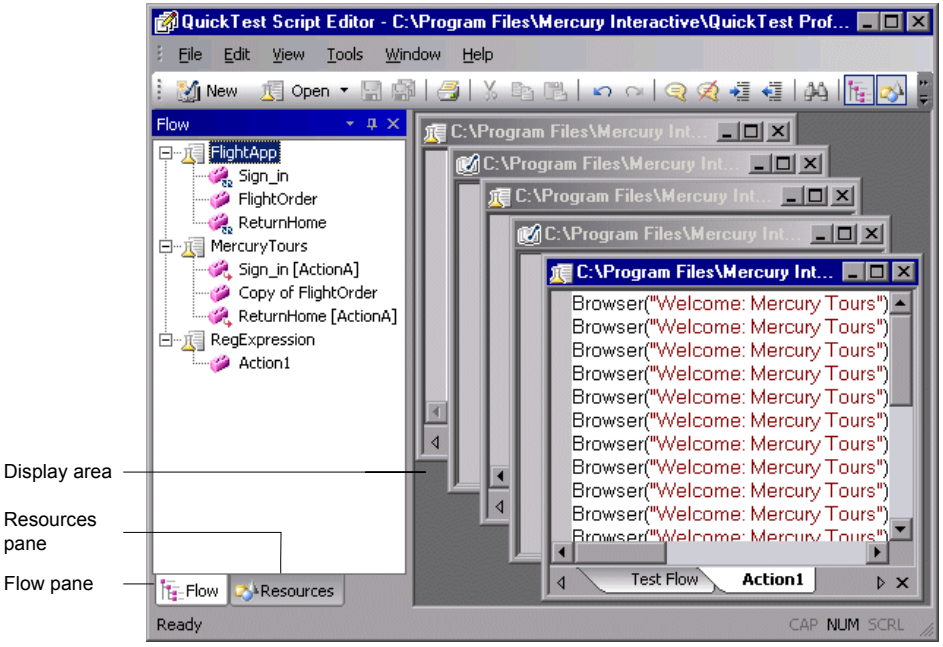
- The QuickTest Script Editor enables you to work with QuickTest tests and function libraries only. To work with components or scripted components, see Chapter 56, "Working with Business Process Testing."
- You cannot use the Script Editor to modify version control-enabled files stored in Quality Center, 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:

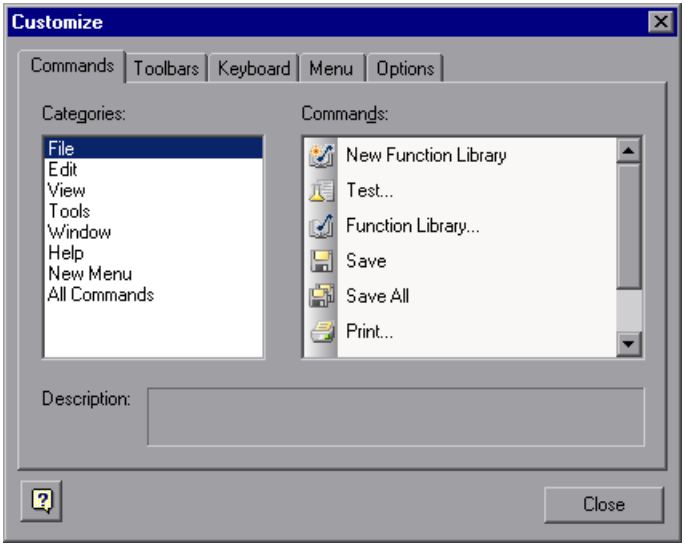
- "Customizing the QuickTest Script Editor Window" on page 1384
- "Understanding the Flow Pane" on page 1386
- "Understanding the Resources Pane" on page 1388
- "Understanding the Display Area" on page 1391

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

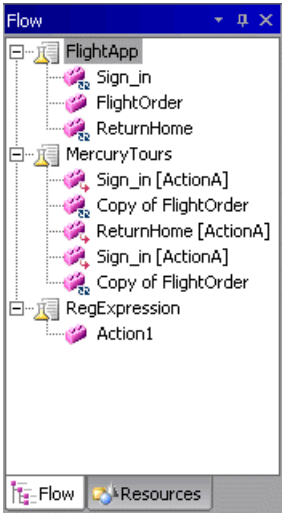
You can select which of the available menus to display in the Script Editor window, and the commands that appear in the context menus. You can choose how the menus are animated, and whether they are displayed with a shadow. You can also reset the displayed menus to the default.

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.

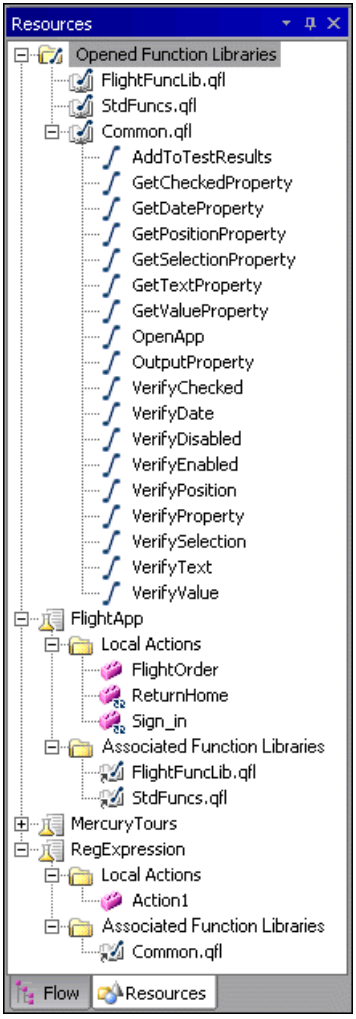
Tips:

- 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.

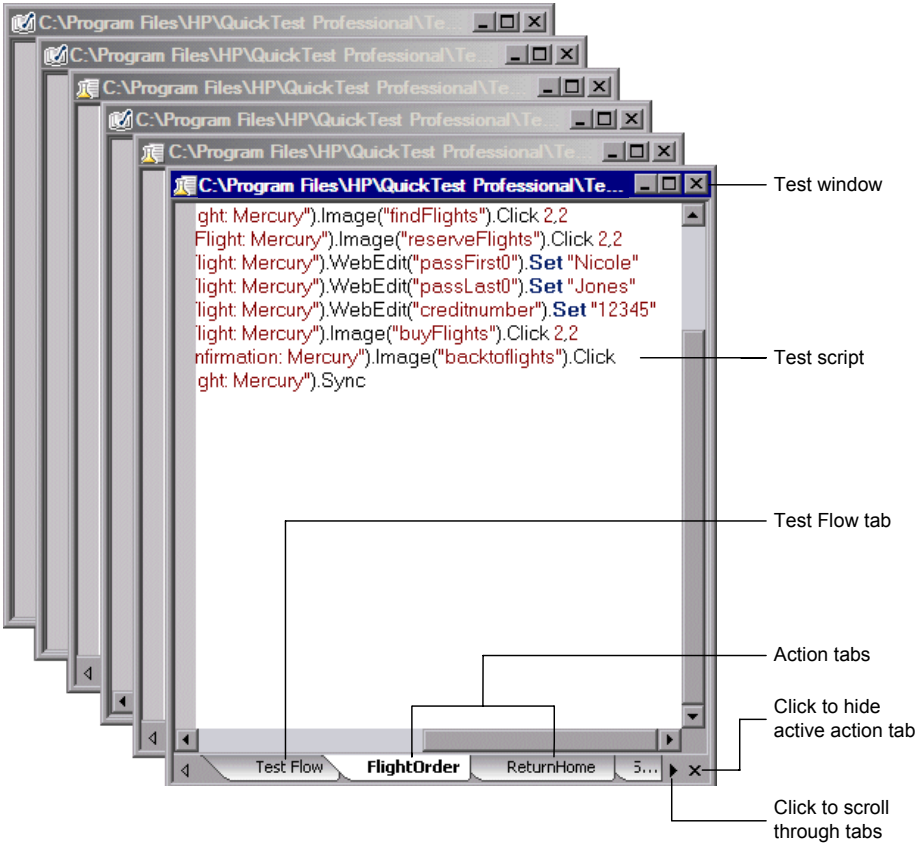
Tips:

- 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.

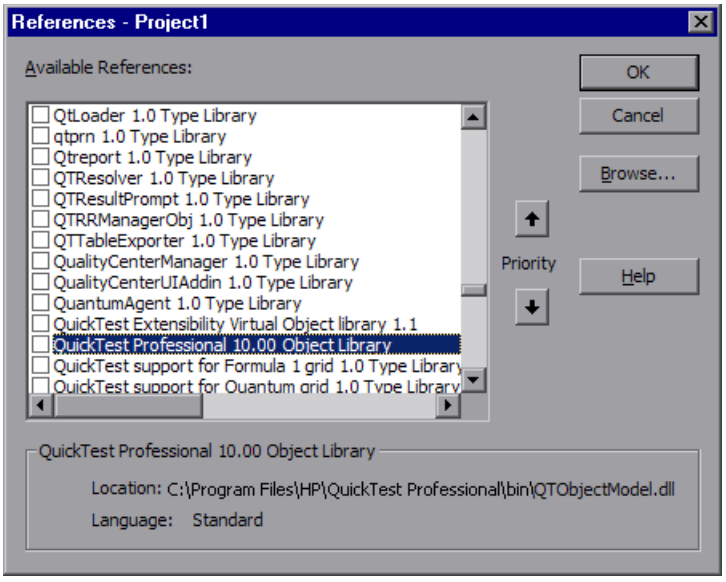
Writing Your Automation Script

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.

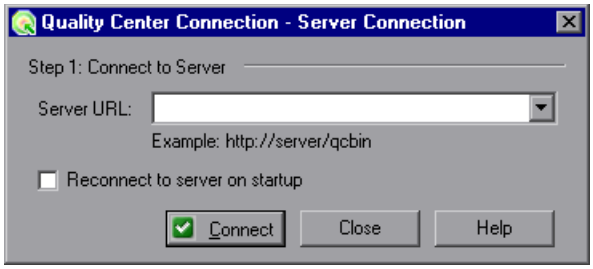
Connecting QuickTest to Quality Center

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.



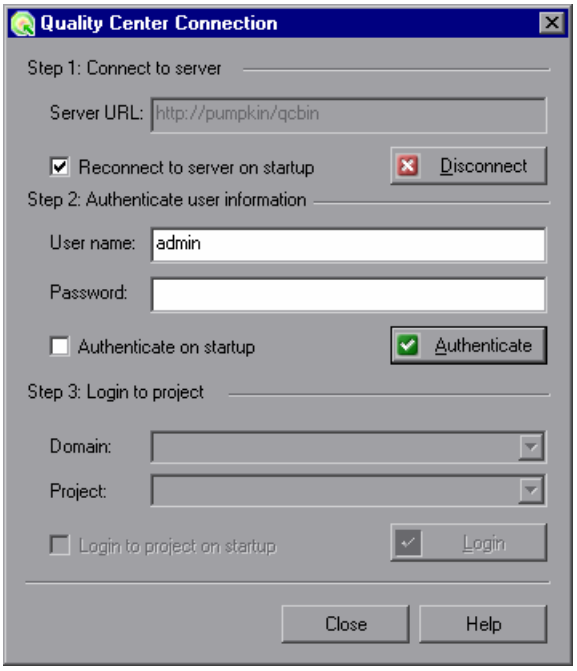
2 In the **Server URL** box, type the URL address of the Web server where Quality Center is installed.

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.

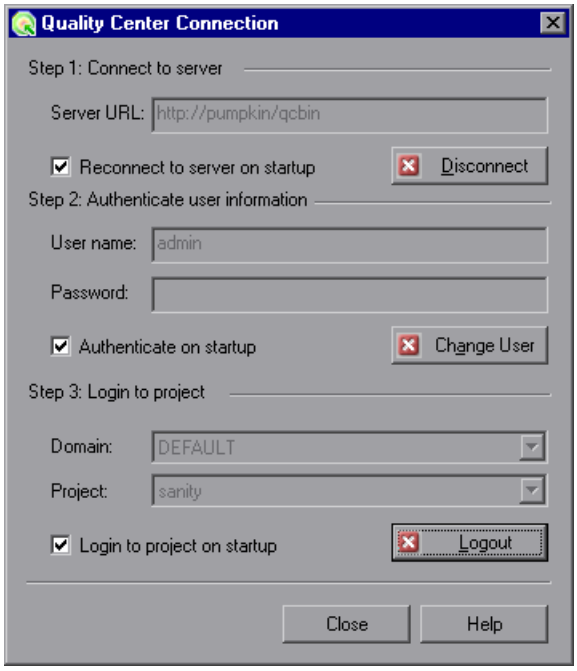
Disconnecting QuickTest from Quality Center

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

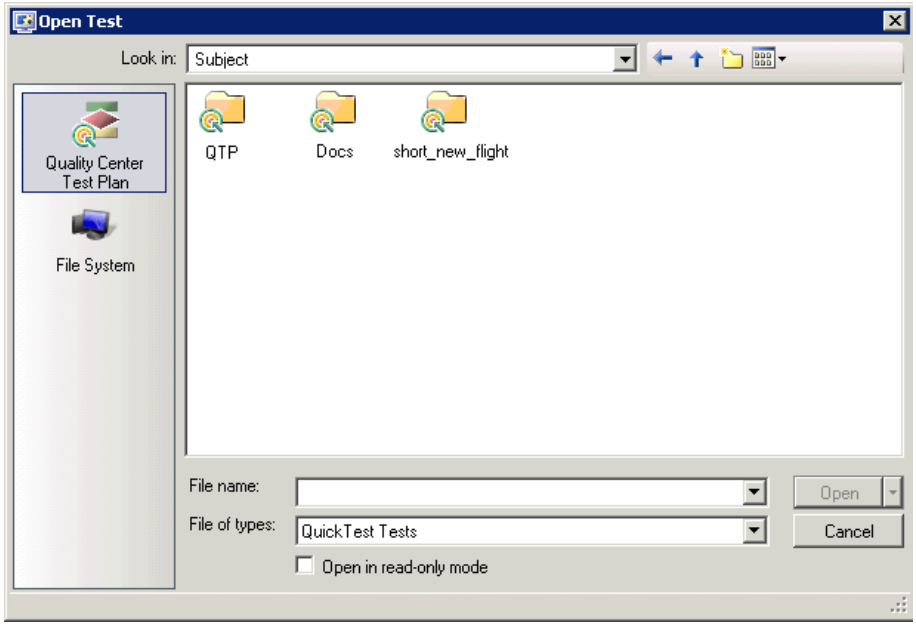
When QuickTest is connected to a Quality Center project, you can open QuickTest tests that are a part of your Quality Center project. You locate tests according to their position in the test plan tree, rather than by their actual location in the file system. You can also open tests from the recent tests 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

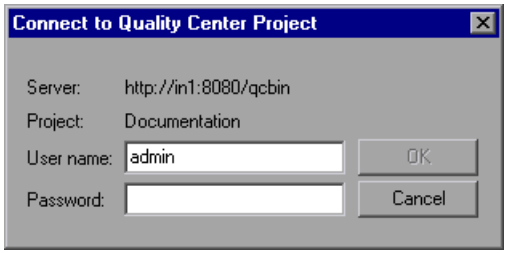
The test opens in read-only mode if:

- 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.

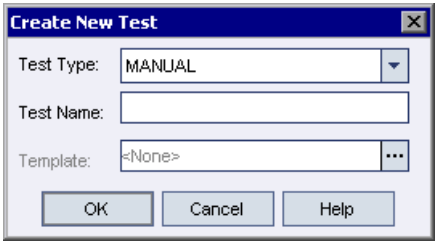
Note: The following procedure describes how to create a test in Quality Center using a template test. This procedure may be different depending on your version of Quality Center. For the most updated instructions on creating a new test in Quality Center, see the *HP Quality Center User Guide*.

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.



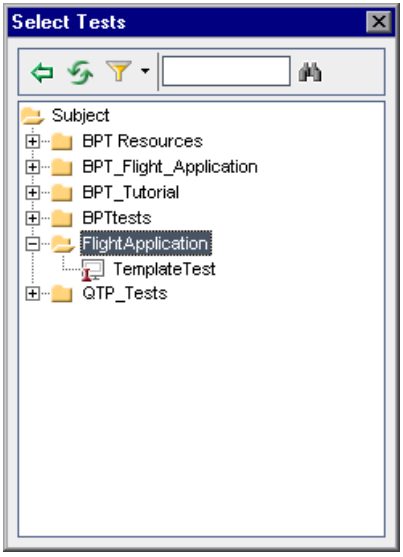
Note: The **Template** box is displayed only if the **QuickTest Professional Add-in for Quality Center** is installed on your computer. If the **Template** box is not displayed, you must install the **QuickTest Professional Add-in for Quality Center** from the QuickTest Professional DVD or from the More 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*.

---

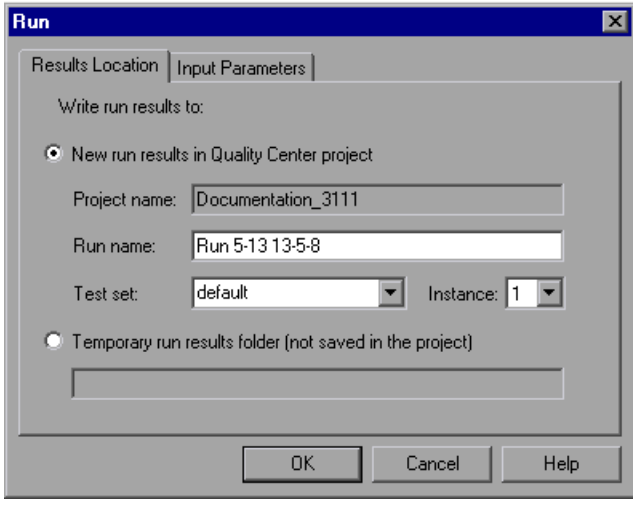
11 Continue creating the test. For more information on creating tests in Quality Center, see the *HP Quality Center User Guide*.

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

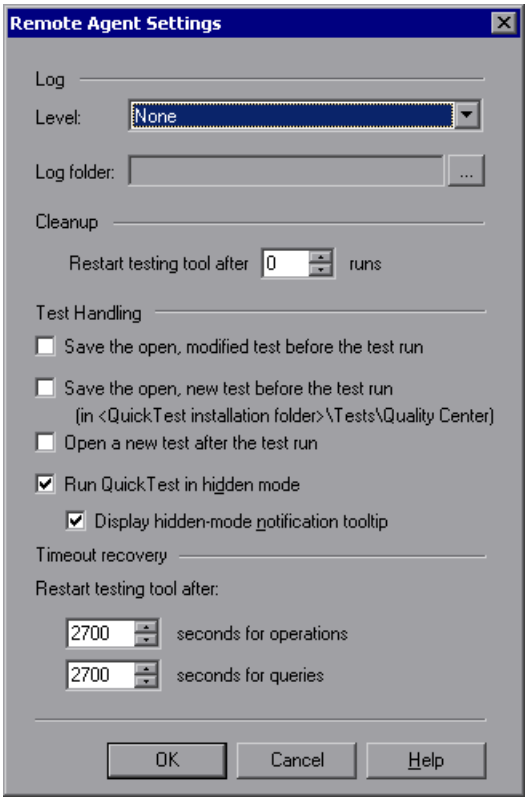
When you run a QuickTest test or business process test from Quality Center, the QuickTest Remote Agent opens on the QuickTest computer. The QuickTest Remote Agent determines how QuickTest behaves when a test is run by a remote application such as Quality Center.

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.

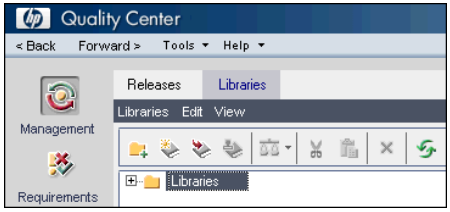
This section provides a general overview of the tabs that are relevant for QuickTest tests. For information on using any of these tabs, see the relevant section in the *HP Quality Center User Guide*.

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.



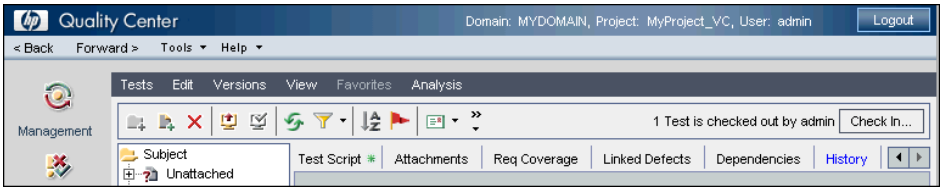
The History Tab

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.

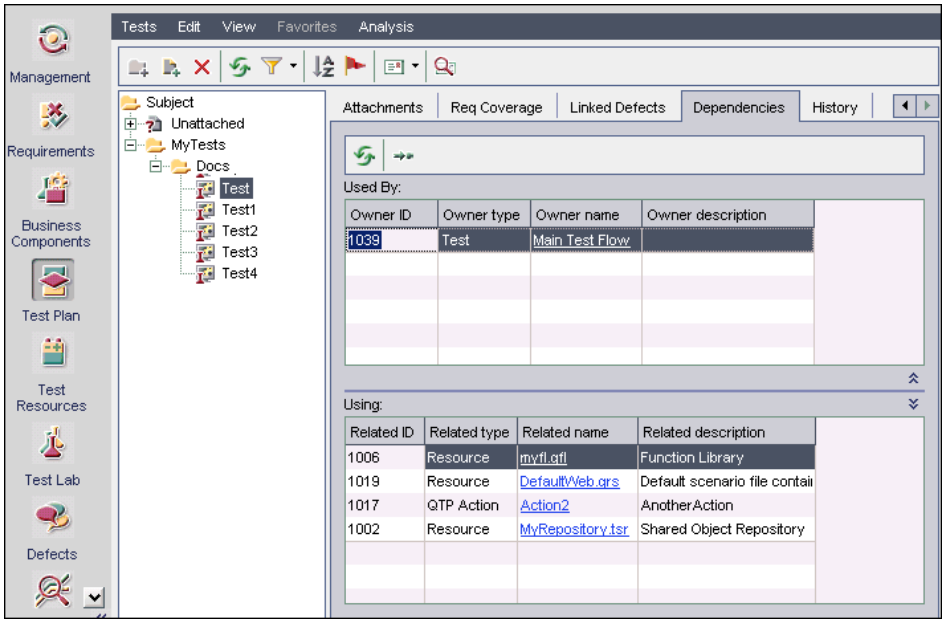
For example, suppose you want to modify the objects in a shared object repository. You can navigate to the shared object repository in the Test Resources module to view a list of the tests with which it is associated. This enables you to determine which assets this resource file is used by and helps 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.)

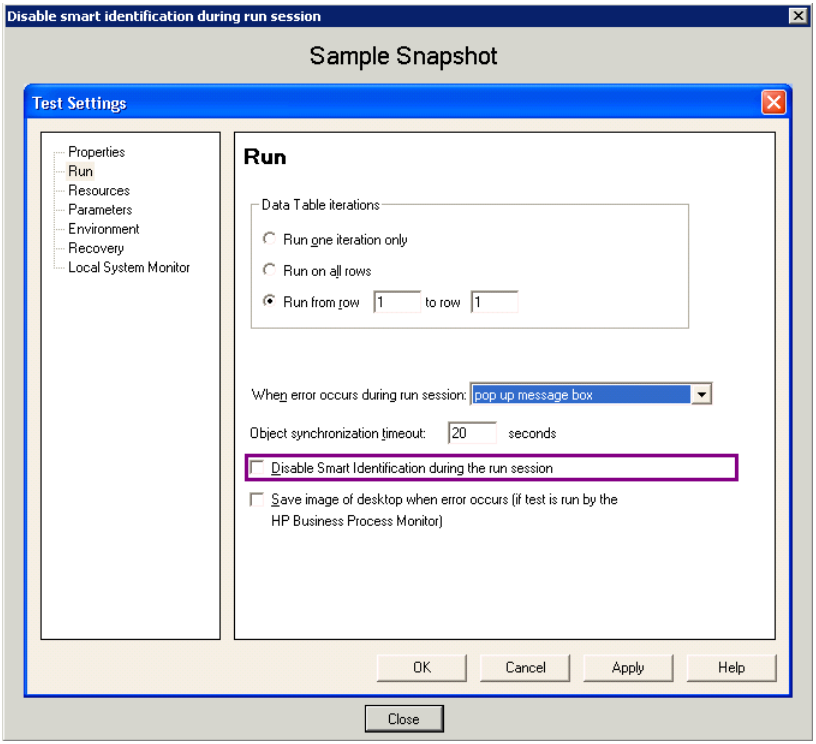
To drill down, do one of the following:

- 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"
```

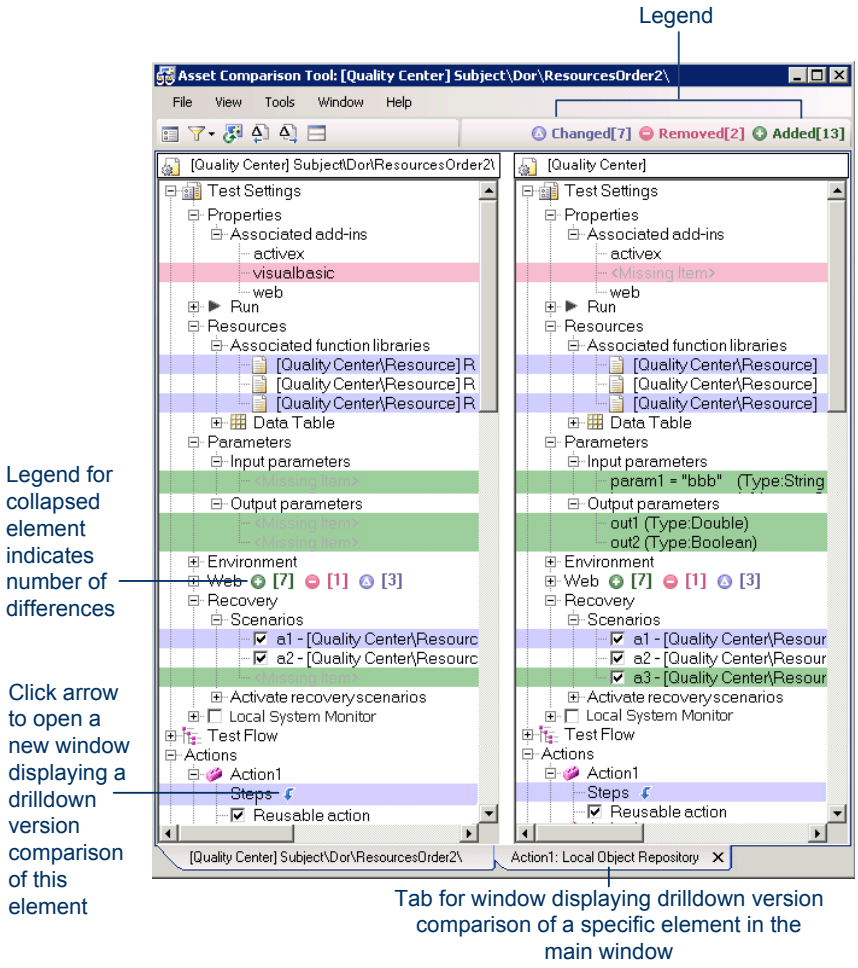
Understanding the Asset Comparison Tool Commands and Options

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:

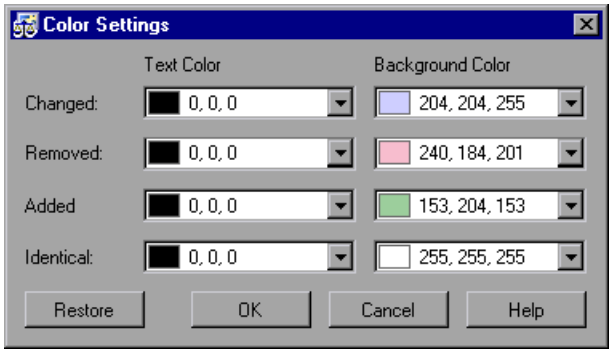
QuickTest Asset Comparison Tool - Context Menu Commands

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.

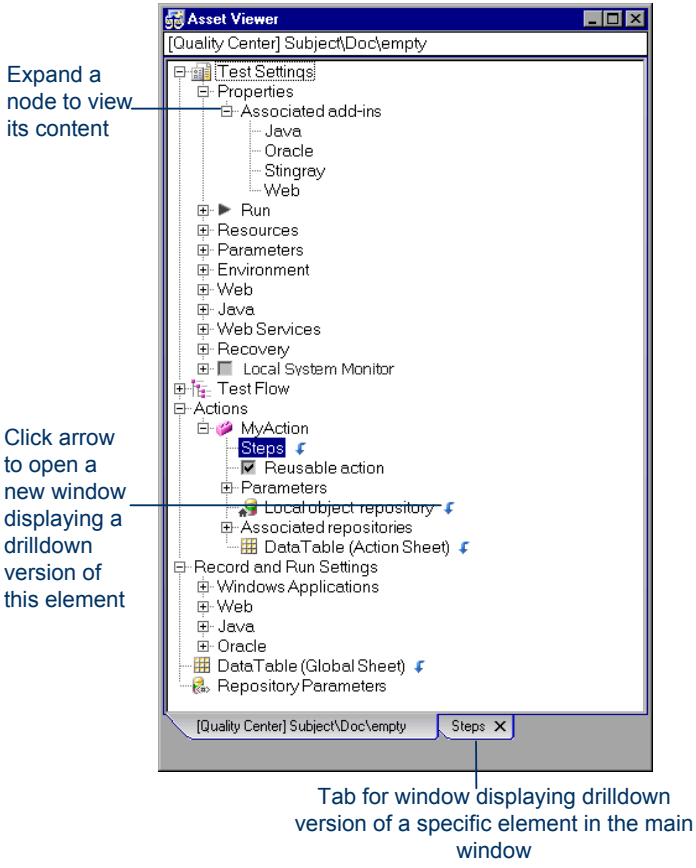
Example:

```
"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.

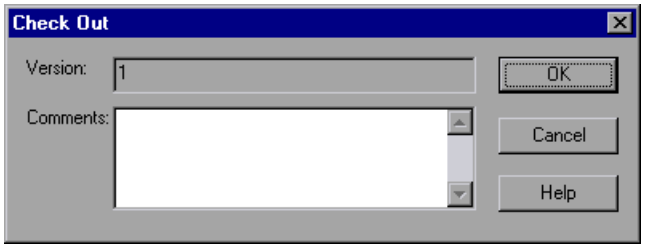
2 Browse to and select the asset.

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.

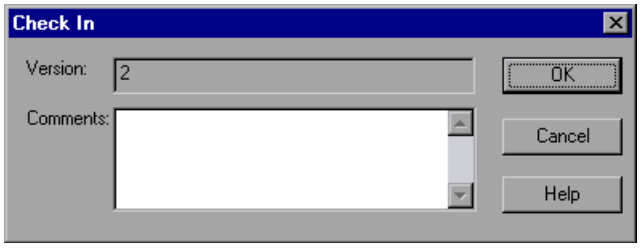
To check in the currently open asset:

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.

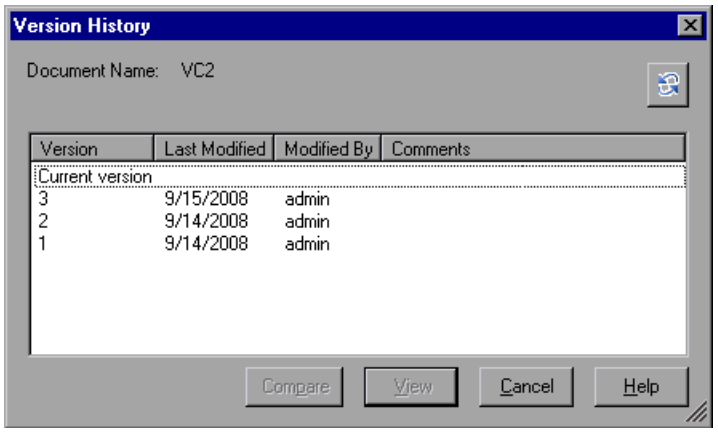
Viewing Version History for an Asset

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.

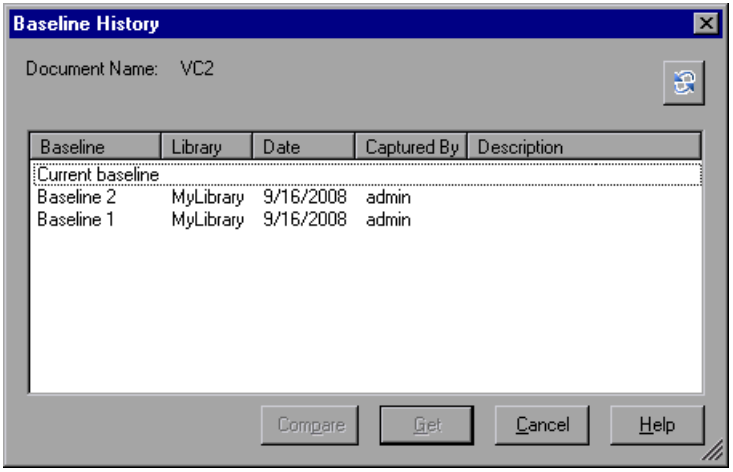
In QuickTest, you can view and compare the assets saved in these baselines. This enables you to review the content of an asset at a specific phase in the project time line.

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.

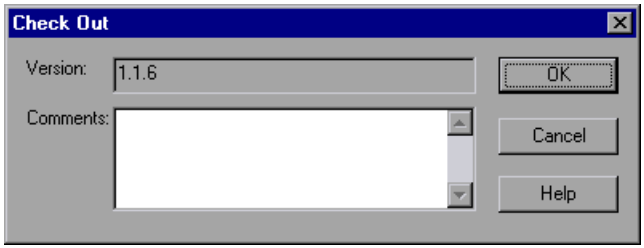
To check out the latest version of a test:

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.

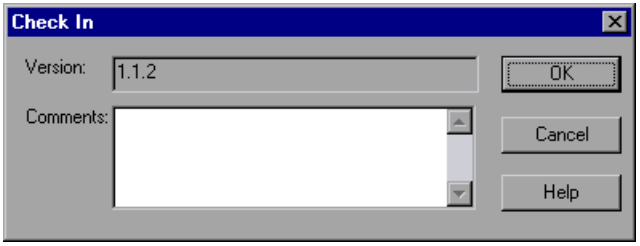
To check in the currently open test:

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.

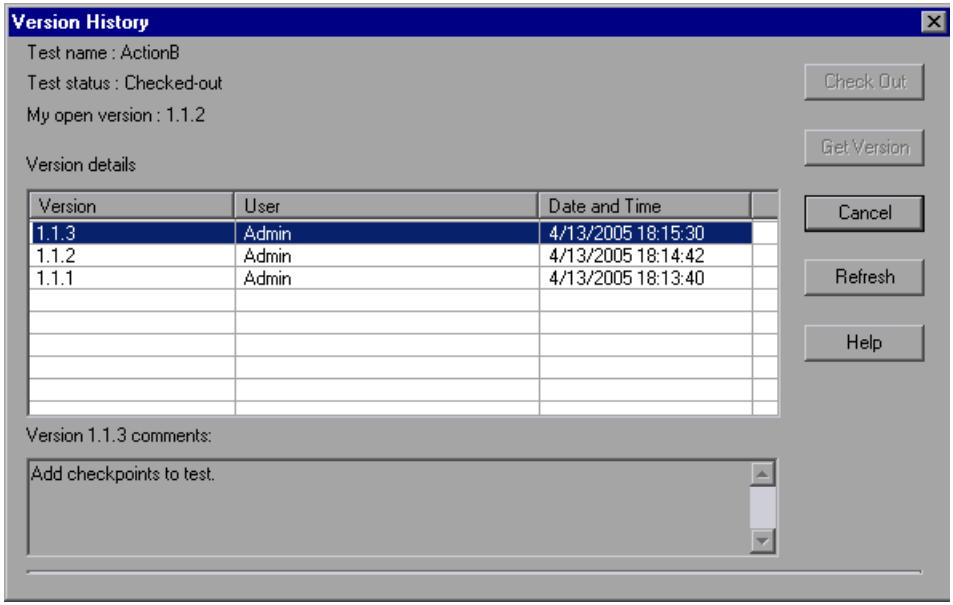
Using the Version History Dialog Box

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:

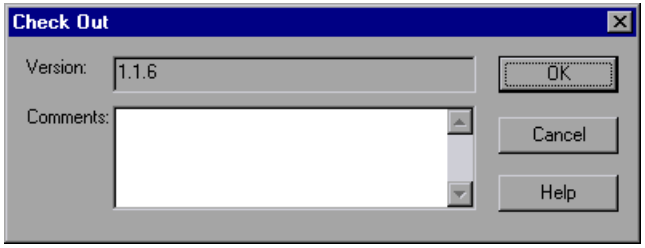
1 Open the Quality Center test. The latest version of the test opens. For more information, see "Opening Tests from a Quality Center 9.x Project with Version Control Support" on page 1496.

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.

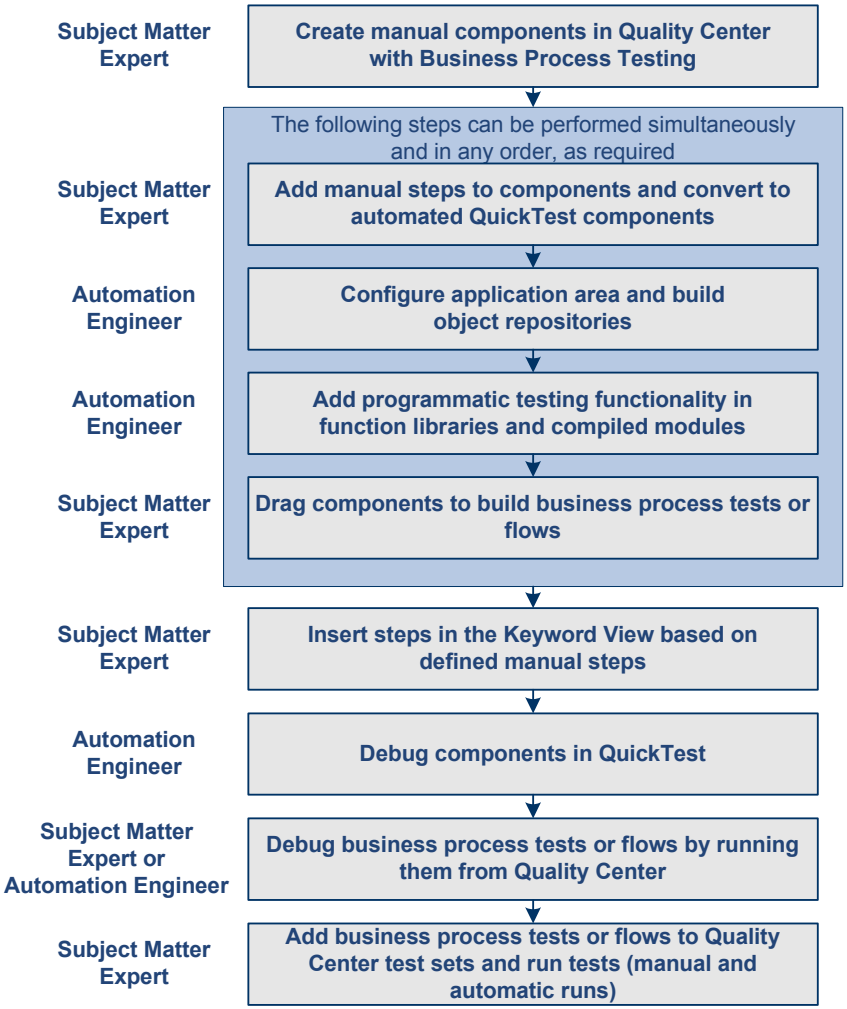
The Automation Engineer uses QuickTest features and functionality to create these resources from within QuickTest. For example, in QuickTest, the Automation Engineer can create and populate various object repositories with test objects that represent the different objects in the application being tested, 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.

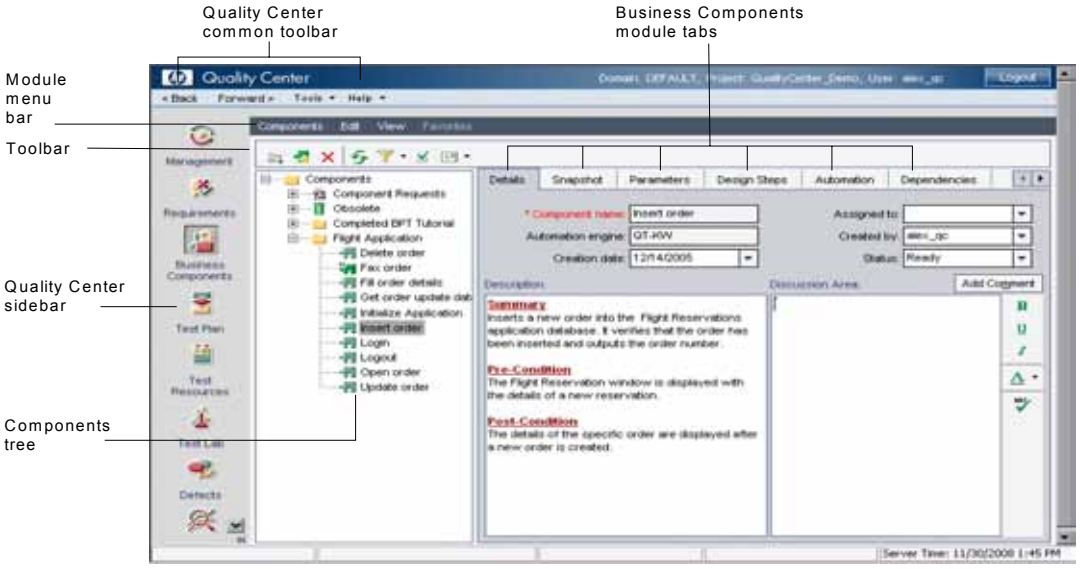
You can also add checkpoint steps and output values to your component.

- 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

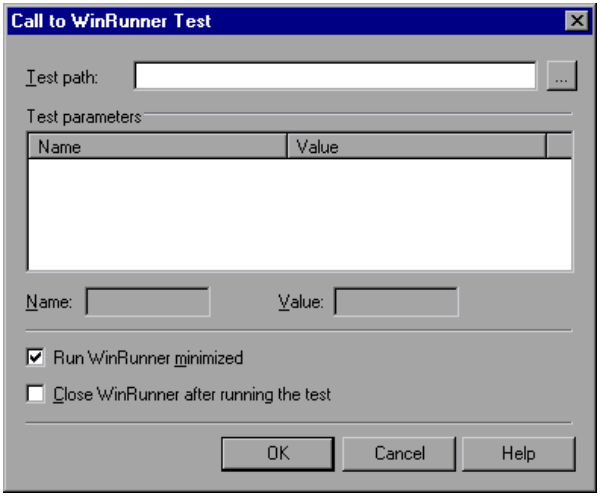
When QuickTest links to WinRunner to run a test, it starts WinRunner, opens the test, and runs it. Information about the WinRunner test run is displayed in the QuickTest Test Results window.

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.

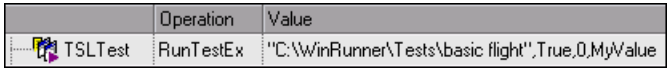
5 Select **Close WinRunner after running the test** if you want the WinRunner application to close when the step calling the WinRunner test is complete.

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

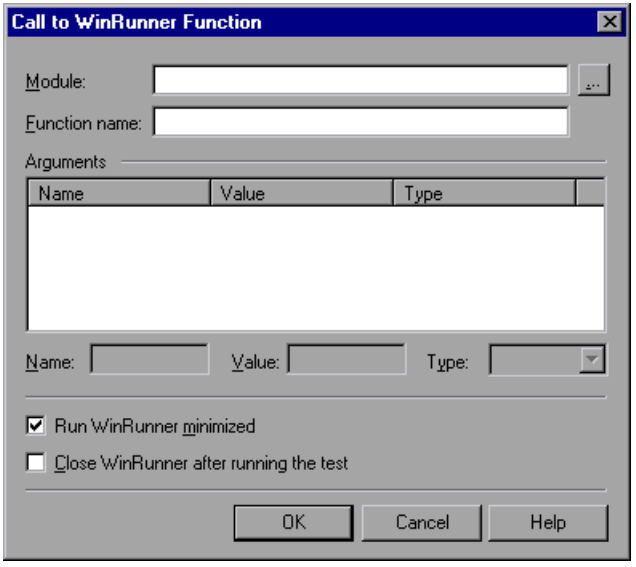
When QuickTest links to WinRunner to call a function, it starts WinRunner, loads the compiled module, and calls the function. This is useful when you want to use a user-defined function from WinRunner in QuickTest.

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*.

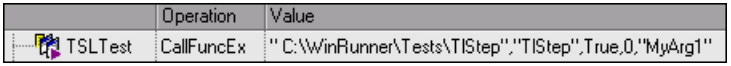
6 Select **Run WinRunner minimized** if you do not want to view the WinRunner window while the function runs.

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:

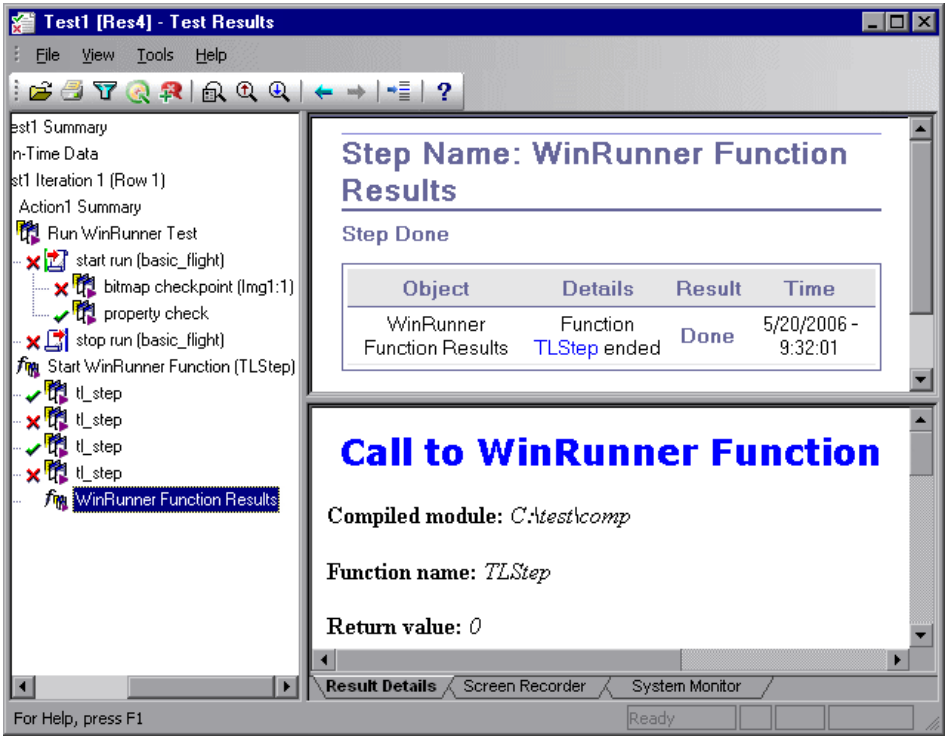
```
TSLTest.CallFuncEx "D:\flightfuncs", "run_flight", TRUE, FALSE,
DataTable("FlightUserName", dtGlobalSheet), DataTable("FlightPwd",
dtGlobalSheet)
```

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.

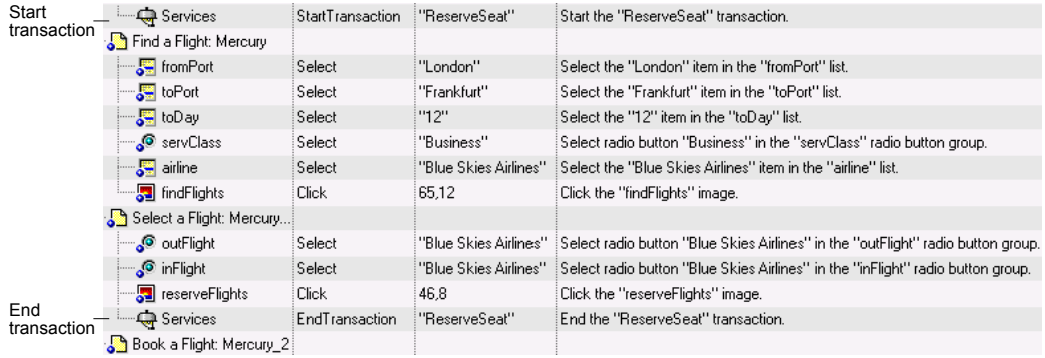
During the test run, the StartTransaction step signals the beginning of the time measurement. The time measurement continues until the EndTransaction step is reached. The test report displays the time it took to perform the transaction.

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:



The same part of the test is displayed in the Expert View as follows:

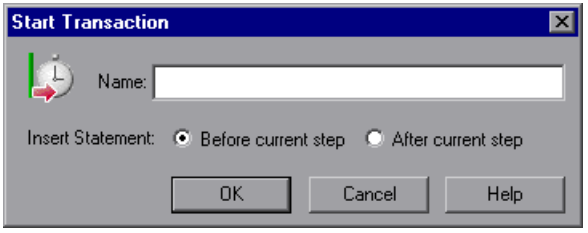
```
Services.StartTransaction "ReserveSeat"
Browser("Welcome: Mercury Tours").Page("Find a Flight: Mercury").
    WebList("fromPort").Select "London"
Browser("Welcome: Mercury Tours").Page("Find a Flight: Mercury").
    WebList("toPort").Select "Frankfurt"
Browser("Welcome: Mercury Tours").Page("Find 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

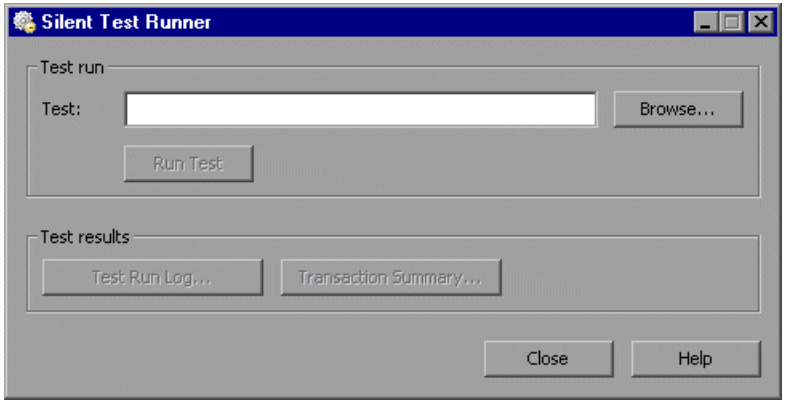
Silent Test Runner enables you to simulate the way a QuickTest test runs from LoadRunner, Performance Center, and Business Availability Center. When you run a test using Silent Test Runner, it runs without opening the QuickTest user interface, and the test runs at the same speed as when it is run from 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**.

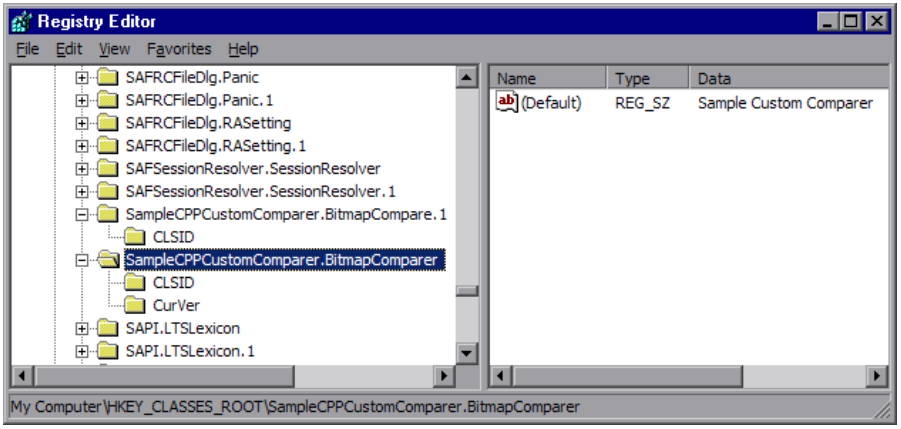
When you design your custom comparer, you must ensure that when it is installed on the QuickTest computer, it is also registered to the component category for QuickTest bitmap comparers. This can be achieved in different ways.

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.

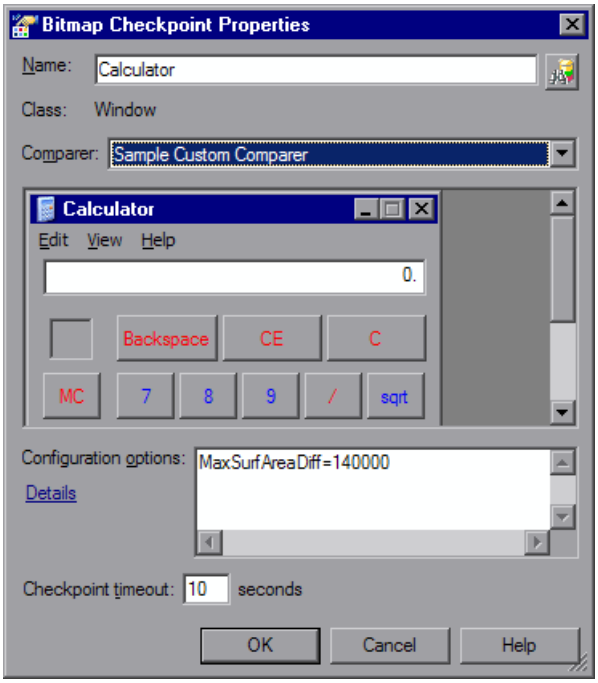
For more information on how to work with bitmap checkpoints, see Chapter 19, "Checking Bitmaps."

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.

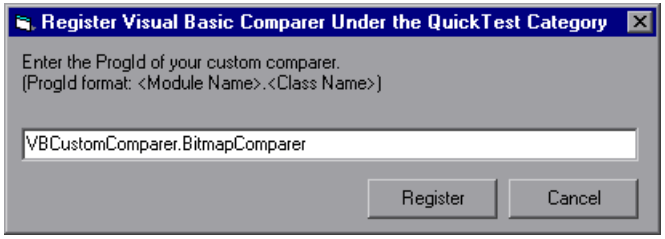
Sample Custom Comparer Registration

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

B

C

D

E

F

G

H

I

J

K

L

M

N

O

Q

S

T

U

V

W

X

Z