



Mercury QuickTest Professional™

USER'S GUIDE

Mercury QuickTest Professional™

User's Guide

Version 8.0



Mercury QuickTest Professional User's Guide, Version 8.0

This manual, and the accompanying software and other documentation, is protected by U.S. and international copyright laws, and may be used only in accordance with the accompanying license agreement. Features of the software, and of other products and services of Mercury Interactive Corporation, may be covered by one or more of the following patents: U.S. Patent Nos. 5,701,139; 5,657,438; 5,511,185; 5,870,559; 5,958,008; 5,974,572; 6,138,157; 6,144,962; 6,205,122; 6,237,006; 6,341,310; 6,360,332; 6,449,739; 6,470,383; 6,477,483; 6,549,944; 6,560,564; and 6,564,342; 6,564,342; 6,587,969; 6,631,408; 6,631,411; 6,633,912 and 6,694,288. Other patents pending. All rights reserved.

Mercury, Mercury Interactive, the Mercury Interactive logo, LoadRunner, LoadRunner TestCenter, QuickTest Professional, SiteScope, SiteSeer, TestDirector, Topaz and WinRunner are trademarks or registered trademarks of Mercury Interactive Corporation or its subsidiaries, in the United States and/or other countries. The absence of a trademark from this list does not constitute a waiver of Mercury Interactive's intellectual property rights concerning that trademark.

All other company, brand and product names are registered trademarks or trademarks of their respective holders. Mercury Interactive Corporation disclaims any responsibility for specifying which marks are owned by which companies or which organizations.

Mercury Interactive Corporation
379 North Whisman Road
Mountain View, CA 94043
Tel: (650) 603-5200
Toll Free: (800) TEST-911
Customer Support: (877) TEST-HLP
Fax: (650) 603-5300

© 2004 Mercury Interactive Corporation, All rights reserved

If you have any comments or suggestions regarding this document, please send them via e-mail to documentation@mercury.com.

Table of Contents

Welcome to QuickTest	xiii
Using This Guide	xiii
Documentation Updates	xvii
Typographical Conventions.....	xviii

PART I: STARTING THE TESTING PROCESS

Chapter 1: Introduction	3
Testing with QuickTest.....	4
Understanding the Testing Process	5
Programming in the Expert View.....	8
Managing the Testing Process Using Quality Center	8
Using the Sample Sites	9
Modifying License Information	9
Chapter 2: QuickTest at a Glance.....	11
Starting QuickTest	12
The QuickTest Window.....	14
Test Pane.....	16
Active Screen	18
Data Table.....	19
Debug Viewer Pane.....	19
Using QuickTest Commands.....	20
Browsing the QuickTest Professional Program Folder	29

PART II: WORKING WITH TEST OBJECTS

Chapter 3: Understanding the Test Object Model.....	33
About Understanding the Test Object Model.....	33
Applying the Test Object Model Concept.....	37
Viewing Object Properties Using the Object Spy.....	42
Viewing Object Methods and Method Syntax Using the Object Spy	46

Table of Contents

Chapter 4: Managing Test Objects	49
About Managing Test Objects	50
Understanding the Object Repository Dialog Box.....	51
Understanding the Object Properties Dialog Box.....	57
Modifying Test Object Properties While Editing Your Test or Component.....	60
Working with Test Objects During a Run Session	66
Modifying Object Descriptions	67
Adding Objects to the Object Repository	73
Deleting an Object from the Object Repository	80

PART III: CREATING TESTS OR COMPONENTS

Chapter 5: Designing Tests	85
About Designing Tests.....	85
Planning a Test	87
Recording a Test or Component	88
Understanding Your Recorded Test or Component	92
Enhancing Your Test	93
Managing Your Test	95
Choosing the Recording Mode	101
Changing the Active Screen	108
Creating, Opening, and Saving Tests with Locked Resources	109
Chapter 6: Understanding Checkpoints	115
About Understanding Checkpoints	115
Adding Checkpoints to a Test or Component	116
Understanding Types of Checkpoints.....	117
Chapter 7: Checking Object Property Values	123
About Checking Object Properties	123
Creating Standard Checkpoints	124
Understanding the Checkpoint Properties Dialog Box	126
Understanding the Image Checkpoint Properties Dialog Box.....	130
Modifying Checkpoints.....	132
Chapter 8: Checking Tables and Databases	133
About Checking Tables and Databases	133
Creating a Table Checkpoint	134
Creating a Check on a Database	135
Understanding the Table/Database Checkpoint Properties Dialog Box	140
Modifying a Table Checkpoint	149
Modifying a Database Checkpoint.....	149

Chapter 9: Checking Text.....	151
About Checking Text.....	151
Creating a Text Checkpoint	153
Creating a Standard Checkpoint for Checking Text.....	155
Creating a Text Area Checkpoint.....	156
Understanding the Text/Text Area Checkpoint Properties	
Dialog Box	159
Modifying a Text or Text Area Checkpoint	168
Chapter 10: Checking Bitmaps.....	169
About Checking Bitmaps.....	169
Checking a Bitmap	170
Modifying a Bitmap Checkpoint.....	177
Chapter 11: Checking XML	181
About Checking XML.....	181
Creating XML Checkpoints.....	183
Modifying XML Checkpoints.....	201
Reviewing XML Checkpoint Results	201
Using XML Objects and Methods to Enhance Your Test or Component.....	202
Chapter 12: Parameterizing Values	203
About Parameterizing Values	203
Parameterizing Values in Steps and Checkpoints.....	205
Using Test, Action, and Component Input Parameters.....	211
Using Data Table Parameters.....	215
Using Environment Variable Parameters	222
Using Random Number Parameters	233
Example of a Parameterized Test.....	235
Using the Data Driver to Parameterize Your Test	241
Chapter 13: Outputting Values	247
About Outputting Values	247
Creating Output Values.....	248
Outputting Property Values	255
Specifying the Output Type and Settings	261
Outputting Text Values	268
Outputting Database Values.....	277
Outputting XML Values	280

Table of Contents

Chapter 14: Configuring Values.....	285
About Configuring Values.....	285
Configuring Constant and Parameter Values	286
Understanding and Using Regular Expressions	290
Defining Regular Expressions.....	293
Chapter 15: Working with the Keyword View.....	303
About Working with the Keyword View.....	304
Understanding the Keyword View	305
Working with Steps in the Keyword View	309
Setting Keyword View Display Options	323
Viewing Properties of Step Elements in the Keyword View.....	326
Using Conditional and Loop Statements in the Keyword View.....	327
Working with Breakpoints in the Keyword View	328
Chapter 16: Learning Virtual Objects	329
About Learning Virtual Objects	329
Understanding Virtual Objects	330
Understanding the Virtual Object Manager	331
Defining a Virtual Object	333
Removing or Disabling Virtual Object Definitions.....	338
Chapter 17: Working with Actions	341
About Working with Actions	342
Using Global and Action Data Sheets	345
Using the Action Toolbar	347
Creating New Actions.....	348
Inserting Calls to Existing Actions	352
Nesting Actions	360
Splitting Actions.....	362
Using Action Parameters	364
Setting Action Properties	366
Setting Action Call Properties	375
Sharing Action Information.....	382
Exiting an Action	384
Removing Actions from a Test	385
Renaming Actions	389
Creating an Action Template	391
Guidelines for Working with Actions	392

Chapter 18: Working with Data Tables	395
About Working with Data Tables.....	395
Working with Global and Action Sheets	397
Saving the Data Table.....	399
Editing the Data Table.....	400
Importing Data from a Database.....	409
Using Formulas in the Data Table.....	412
Using Data Table Scripting Methods.....	417
Chapter 19: Defining and Using Recovery Scenarios	419
About Defining and Using Recovery Scenarios.....	419
Deciding When to Use Recovery Scenarios.....	421
Defining Recovery Scenarios	422
Understanding the Recovery Scenario Wizard	426
Managing Recovery Scenarios	451
Setting the Recovery Scenarios List for Your Tests or Components.	456
Programmatically Controlling the Recovery Mechanism	462
Chapter 20: Adding Steps Containing Programming Logic.....	465
About Adding Steps Containing Programming Logic	466
Inserting Steps Using the Step Generator	467
Using Conditional Statements	484
Using Loop Statements.....	488
Generating "With" Statements for Your Test or Component	491
Sending Messages to Your Test Results	496
Adding Comments	498
Synchronizing Your Test or Component	499
Measuring Transactions	504

PART IV: RUNNING AND DEBUGGING TESTS AND COMPONENTS

Chapter 21: Running Tests and Components.....	511
About Running Tests and Components.....	511
Running Your Entire Test or Component.....	512
Running Part of Your Test or Component.....	517
Updating a Test or Component	519
Using Optional Steps.....	525
Running a Test Batch	527

Table of Contents

Chapter 22: Debugging Tests and Components	529
About Debugging Tests and Components	530
Using the Step Commands.....	530
Pausing a Run Session	532
Setting Breakpoints.....	533
Removing Breakpoints	534
Using the Debug Viewer.....	535
Handling Run Errors.....	537
Practicing Debugging a Test	538
Chapter 23: Analyzing Test Results	541
About Analyzing Test Results	541
Understanding the Test Results Window.....	542
Viewing the Results of a Run Session.....	546
Viewing Checkpoint Results	557
Viewing Parameterized Values and Output Value Results.....	578
Analyzing Smart Identification Information in the Test Results.....	587
Deleting Test Results	590
Submitting Defects Detected During a Run Session	597
Viewing WinRunner Test Steps in the Test Results	603
Customizing the Test Results Display	606

PART V: CONFIGURING QUICKTEST

Chapter 24: Setting Global Testing Options	611
About Setting Global Testing Options	611
Using the Options Dialog Box	612
Setting General Testing Options	614
Setting Folder Testing Options.....	616
Setting Active Screen Options.....	618
Setting Run Testing Options	625
Setting Windows Application Testing Options	628
Setting Web Testing Options	639

Chapter 25: Setting Options for Individual Tests or Components	651
About Setting Options for Individual Tests or Components.....	652
Using the Test Settings Dialog Box	654
Using the Business Component Settings Dialog Box	656
Defining Properties for Your Test.....	659
Defining Properties for Your Component.....	660
Defining Run Settings for Your Test	665
Defining a Snapshot for Your Component.....	669
Defining Application Settings for Your Component.....	670
Defining Resource Settings for Your Test	674
Defining Resource Settings for Your Component.....	680
Defining Parameters for Your Test or Component	682
Defining Environment Settings for Your Test or Component	685
Defining Web Settings for Your Test or Component.....	692
Defining Recovery Scenario Settings for Your Test or Component..	694
Chapter 26: Setting Record and Run Options	699
About Setting Record and Run Options.....	699
Using the Record and Run Settings Dialog Box	700
Setting Web Record and Run Options	703
Setting Windows Applications Record and Run Options	705
Using Environment Variables to Specify the Application Details for Your Test	709
Chapter 27: Customizing the Expert View	713
About Customizing the Expert View.....	713
Customizing Expert View Behavior	714
Customizing Script Element Appearance.....	717
Personalizing Editing Commands.....	719
Chapter 28: Setting Testing Options During the Run Session	723
About Setting Testing Options During the Run Session	723
Setting Testing Options.....	724
Retrieving Testing Options.....	725
Controlling the Test Run.....	726
Adding and Removing Run-Time Settings.....	727
PART VI: WORKING WITH SUPPORTED ENVIRONMENTS	
Chapter 29: Working with QuickTest Add-Ins	731
About Working with QuickTest Add-ins.....	731
Loading QuickTest Add-ins	732
Tips for Working with QuickTest Add-ins	736

Table of Contents

Chapter 30: Testing Web Objects	739
About Testing Web Objects.....	739
Working with Web Browsers.....	742
Checking Web Objects	744
Checking Web Pages	747
Checking Web Content Accessibility.....	760
Accessing Password-Protected Resources in the Active Screen	764
Activating Methods Associated with a Web Object.....	770
Using Scripting Methods with Web Objects.....	771
Chapter 31: Testing Visual Basic Applications	773
About Testing Visual Basic Applications.....	774
Recording and Running on Visual Basic Applications	775
Checking Visual Basic Objects	776
Using Visual Basic Objects and Methods to Enhance Your Test or Component.....	778
Chapter 32: Testing ActiveX Controls	779
About Testing ActiveX Controls	779
Recording and Running on ActiveX Controls	781
Checking ActiveX Controls.....	783
Activating an ActiveX Control Method	785
Using Scripting Methods with ActiveX Controls.....	785

PART VII: ADVANCED FEATURES

Chapter 33: Configuring Object Identification	789
About Configuring Object Identification	789
Understanding the Object Identification Dialog Box.....	791
Configuring Smart Identification.....	803
Mapping User-Defined Test Object Classes	812
Chapter 34: Choosing the Object Repository Mode	815
About Choosing the Object Repository Mode	815
Deciding Which Object Repository Mode to Choose	817
Setting the Object Repository Mode	828
Chapter 35: Configuring Web Event Recording	835
About Configuring Web Event Recording	835
Selecting a Standard Event Recording Configuration.....	837
Customizing the Event Recording Configuration	839
Saving and Loading Custom Event Configuration Files.....	849
Resetting Event Recording Configuration Settings.....	850

Chapter 36: Working with the Expert View.....	851
About Working with the Expert View.....	852
Understanding and Using the Expert View	853
Navigating in the Expert View	864
Understanding Basic VBScript Syntax.....	873
Using Programmatic Descriptions.....	878
Running and Closing Applications Programmatically	887
Using Comments, Control-Flow, and Other VBScript Statements.....	888
Retrieving and Setting Test Object Property Values	898
Accessing Run-Time Object Properties and Methods	899
Running DOS Commands.....	901
Enhancing Your Tests using the Windows API.....	902
Choosing Which Steps to Report During the Run Session	904
Chapter 37: Working with User-Defined Functions	907
About Working with User-Defined Functions	907
Working with Associated Library Files.....	908
Executing Externally-Defined Functions from Your Test or Component	910
Using User-Defined Test Object Methods	912
Chapter 38: Automating QuickTest Operations	919
About Automating QuickTest Operations	920
Deciding When to Use QuickTest Automation Programs	921
Choosing a Language and Development Environment for Designing and Running Automation Programs.....	923
Learning the Basic Elements of a QuickTest Automation Program..	925
Generating Automation Scripts.....	926
Using the QuickTest Automation Object Model Reference.....	927

PART VIII: WORKING WITH OTHER MERCURY PRODUCTS

Chapter 39: Working with WinRunner	931
About Working with WinRunner	931
Calling WinRunner Tests	932
Calling WinRunner Functions	937

Table of Contents

Chapter 40: Working with Quality Center.....	943
About Working with Quality Center	944
Connecting to and Disconnecting from Quality Center.....	946
Saving Tests to a Quality Center Project.....	951
Opening Tests from a Quality Center Project.....	952
Running a Test Stored in a Quality Center Project.....	957
Managing Test Versions in QuickTest.....	959
Setting Preferences for Quality Center Test Runs	967
Chapter 41: Working with Business Process Testing.....	973
About Working with Business Process Testing	973
Understanding Components.....	980
Creating Components	982
Opening Existing Components.....	983
Saving Components	985
Working with Component Templates	987
Recording Components	991
Running Components.....	993
Chapter 42: Working with Mercury Performance Testing and Application Management Products	995
About Working with Mercury Performance Testing and Application Management Products	996
Using QuickTest's Performance Testing and Application Management Features	997
Designing QuickTest Tests for Use with LoadRunner or the Business Process Monitor.....	999
Inserting and Running Tests in LoadRunner or Mercury Application Management.....	1000
PART IX: APPENDIX	
Appendix A: Working with QuickTest— Frequently Asked Questions.....	1003
Recording and Running Tests	1003
Programming in the Expert View.....	1004
Working with Dynamic Content	1005
Advanced Web Issues	1006
Test Maintenance	1007
Testing Localized Applications.....	1009
Improving QuickTest Performance	1010
Index	1015

Welcome to QuickTest

Welcome to QuickTest Professional, the Mercury automated keyword-driven testing solution. QuickTest provides everything you need to quickly create and run tests and components.

Using This 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 components, and report defects detected during the testing process.

It contains 8 parts:

Part I Starting the Testing Process

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

Part II Working with Test Objects

Explains how QuickTest identifies objects in your application and how to work with the object repository.

Part III Creating Tests or Components

Describes how to create tests and components, insert checkpoints, parameters, and output values, use regular expressions, work with actions, and handle unexpected events that occur during a run session.

Part IV Running and Debugging Tests and Components

Describes how to run tests and components, analyze results, and control run sessions to identify and isolate bugs in test or component scripts.

Part V Configuring QuickTest

Describes how to modify QuickTest settings to match your testing needs.

Part VI Working with Supported Environments

Explains how to work with QuickTest core add-ins, and includes environment-specific information for testing Web sites, ActiveX controls, and Visual Basic applications.

Part VII Advanced Features

Describes how to choose an object repository mode, configure object identification and create Smart Identification definitions, and enhance your test or component in Expert View mode. It also introduces several programming techniques to create a more powerful script. **This section is recommended for advanced QuickTest users.**

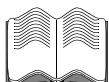
Part VIII Working with Other Mercury Products

Describes how you can run tests and components and call functions in compiled modules from WinRunner, the Mercury enterprise functional testing tool for Microsoft Windows applications. This section also describes how QuickTest can be used with Business Process Testing, and how to create components in QuickTest. In addition, this section describes how QuickTest interacts with Mercury Quality Center, the Mercury centralized quality solution (formerly TestDirector), and details considerations for designing QuickTest tests for use with Mercury performance testing and application management products.

Appendix

Provides information on frequently asked questions.

QuickTest Documentation Set



In addition to this User's Guide, QuickTest Professional comes with the following printed documentation:

QuickTest Professional Installation Guide explains how to install QuickTest Professional.

QuickTest Professional Tutorial teaches you basic QuickTest skills and shows you how to design tests for your applications.

QuickTest Professional Shortcut Key Reference Card provides a list of commands that you can execute using shortcut keys.

Online Resources



QuickTest Professional includes the following online resources:

Readme (available from the QuickTest Professional Start menu program folder) provides the latest news and information about QuickTest Professional.

What's New in QuickTest Professional (available from **Help > What's New in QuickTest**) describes the newest features, enhancements, and supported environments in this latest version of QuickTest Professional.

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

QuickTest Professional Tutorial (available from the QuickTest Professional Welcome window, the **Help** menu, and the QuickTest Professional Start menu program folder) teaches you basic QuickTest skills and shows you how to start designing tests for your applications.

QuickTest Professional Context-Sensitive Help (available from specific dialog boxes and windows) describes QuickTest dialog boxes and windows.

QuickTest Professional User's Guide (available from **Help > QuickTest Professional Help**) provides step-by-step instructions for using QuickTest Professional to test your applications.

QuickTest Professional Object Model Reference (available from **Help > QuickTest Professional Help**) describes QuickTest Professional test objects, lists the methods and properties associated with each object, and provides syntax information and examples for the methods.

QuickTest Professional Automation Object Model Reference (available from the QuickTest Professional Start menu program folder and from **Help > QuickTest Automation Object Model Reference**) 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 or component management, by providing objects, methods and properties that enable you to control virtually every QuickTest feature and capability.

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

Mercury Tours sample Web site (available from the QuickTest Professional Start menu program folder and also available from the QuickTest Professional Record and Run Settings dialog box) and the **Mercury Tours** Windows sample flight application (available from the QuickTest Professional Start menu program folder) are the basis for many examples in this book. The URL for this Web site is <http://newtours.mercuryinteractive.com>.

Customer Support Online (available from **Help > Customer Support Online**) uses your default Web browser to open the Mercury Customer Support Web site. This site enables you to browse the knowledge base and add your own articles, post to and search user discussion forums, submit support requests, download patches and updated documentation, and more. The URL for this Web site is <http://support.mercury.com>.

Send Feedback (available from **Help > Send Feedback**) enables you to send online feedback about QuickTest Professional to the product team.

Mercury Interactive on the Web (available from **Help > Mercury Interactive on the Web**) uses your default Web browser to open the Mercury home page. This site provides you with the most up-to-date information on Mercury and its products. This includes new software releases, seminars and trade shows, customer support, educational services, and more. The URL for this Web site is <http://www.mercury.com>.

Documentation Updates

Mercury Interactive is continuously updating its product documentation with new information. You can download the latest version of this document from the Customer Support Web site (<http://support.mercury.com>).

To download updated documentation:

- 1** In the Customer Support Web site, click the **Documentation** link.
- 2** Under **Select Product Name**, select **QuickTest Professional**.

Note that if **QuickTest Professional** does not appear in the list, you must add it to your customer profile. Click **My Account** to update your profile.

- 3** Click **Retrieve**. The Documentation page opens and lists the documentation available for the current release and for previous releases. If a document was recently updated, **Updated** appears next to the document name.
- 4** Click a document link to download the documentation.

Typographical Conventions

This book uses the following typographical conventions:

1, 2, 3	Bold numbers indicate steps in a procedure.
>	The greater-than sign separates menu levels (for example, File > Open).
Stone Sans	The Stone Sans font indicates names of interface elements (for example, the Run button) and other items that require emphasis.
Bold	Bold text indicates method or function names.
<i>Italics</i>	<i>Italic</i> text indicates method or function arguments, file names in syntax descriptions, and book titles. It is also used when introducing a new term.
<>	Angle brackets enclose a part of a file path or URL address that may vary from user to user (for example, < MyProduct installation folder >\bin).
Arial	The Arial font is used for examples and text that is to be typed literally.
Arial bold	The Arial bold font is used in syntax descriptions for text that should be typed literally.
SMALL CAPS	The SMALL CAPS font indicates keyboard keys.
...	In a line of syntax, an ellipsis indicates that more items of the same format may be included. In a programming example, an ellipsis is used to indicate lines of a program that were intentionally omitted.
[]	Square brackets enclose optional arguments.
	A vertical bar indicates that one of the options separated by the bar should be selected.
	Indicates information that is relevant only for action-based tests.
	Indicates information that is relevant only for business components.

Part I

Starting the Testing Process

1

Introduction

Welcome to QuickTest Professional, the Mercury advanced keyword-driven testing solution.

QuickTest Professional enables you to test standard Windows applications, Web objects, ActiveX controls, and Visual Basic applications. You can also acquire additional QuickTest add-ins for a number of special environments (such as Java, Oracle, SAP Solutions, .NET Windows and Web Forms, Siebel, PeopleSoft, Web services, and terminal emulator applications).

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, to allow the processing and display of many diverse languages and character sets.

This introductory section provides you with an overview of the following QuickTest Professional features and testing procedures:

- Testing with QuickTest
- Understanding the Testing Process
- Programming in the Expert View
- Managing the Testing Process Using Quality Center
- Using the Sample Sites
- Modifying License Information

Testing with QuickTest

QuickTest Professional facilitates creating tests and business components by recording operations as you perform them on your application. Tests and business components are two different types of documents that you can use to test that your application or Web site works as expected:

 **Test**—A collection of steps organized into one or more actions, which are used to verify that your application performs as expected.

 **Business Component**—A collection of steps representing a single task in your application. Business components (also known as components) are combined into specific scenarios to build business process tests in Mercury Quality Center with Business Process Testing.

As you navigate through your application, QuickTest records each step you perform and generates a test or component that graphically displays these steps in a table-based Keyword View. For example, clicking a link, selecting a check box, or submitting a form are all recorded in your test or component.

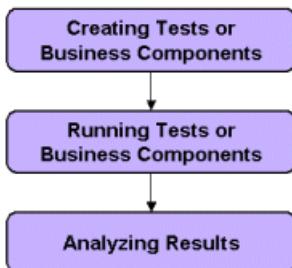
After you have finished recording, you can instruct QuickTest to check the properties of specific objects in your application. For example, you can instruct QuickTest to check that a specific text string is displayed in a particular location on a dialog box, or you can check that a hypertext link on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

You can further enhance your test or component by adding and modifying steps in the Keyword View. When you perform a run session, QuickTest performs each step in your test or component. After the run sessions ends, you can view a report detailing which steps were performed, and which ones succeeded or failed.

 A test is composed of actions. The steps you add to a test are included within the test's actions. Note that by default, each test begins with a single action. You can divide your test into multiple actions to organize your test. Most of the chapters in this guide provide information on how to work within a single action. For information on when and how to work with multiple actions in a test, see Chapter 17, “Working with Actions.” A component does not contain actions—you add steps directly to a component.

Understanding the Testing Process

Testing with QuickTest involves three main stages:



Creating Tests or Components

You create a test or component by either recording a session on your application or Web site or by building an object repository and adding steps manually to the Keyword View using keyword-driven functionality. You can then modify your test or component with special testing options and/or with programming statements.

To create a test or component:

- Add steps to your test, in either or both of the following ways:
 - Record a session on your application or site.

As you navigate through your application or site, QuickTest graphically displays each *step* you perform as a row in the Keyword View. The **Documentation** column of the Keyword View also displays a description of each step in easy-to-understand sentences. A step is something that causes or makes a change in your site or application, such as clicking a link or image, or submitting a data form. For more information, see Chapter 5, “Designing Tests.”

- Build an object repository and use these objects to add steps manually in the Keyword View or Expert View.

Build an object repository that contains all the objects that you want to test in your application or Web site. For more information on building an object repository, see Chapter 4, “Managing Test Objects.”

Create steps by selecting items and operations in the Keyword View and entering information as required. For more information, see Chapter 15, “Working with the Keyword View”. Advanced users can add steps in the Expert View. For more information, see Chapter 36, “Working with the Expert View.”

- Insert checkpoints into your test or component.

A *checkpoint* checks specific values or characteristics of a page, object, or text string and enables you to identify whether or not your Web site or application is functioning correctly. For more information, see Chapter 6, “Understanding Checkpoints.”

- Broaden the scope of your test or component by replacing fixed values with parameters.

When you test your site or application, you can *parameterize* your test or component 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, component, or action parameters and values, or have QuickTest generate random numbers or current user and test data. When you parameterize your test or component, QuickTest substitutes the fixed values in your test or component with 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. For components, the data for each iteration is defined in Quality Center. Each run session that uses a different set of parameterized data is called an iteration. For more information, see Chapter 12, “Parameterizing Values.”

You can also use output values to extract data from your test or component. An *output value* is a value retrieved during the run session and entered into your Data Table or saved as a variable or a parameter. You can subsequently use this output value as input data in your test or component. This enables you to use data retrieved during a run session in other parts of the test or component. For more information, see Chapter 13, “Outputting Values.”

- Use the many functional testing features included in QuickTest to enhance your test or component and/or add programming statements to achieve more complex testing goals.

Running Tests or Components

After you create your test or component, you run it.

- Run your test or component to check your site or application.

The test or component runs from the first line in your test or component and stops at the end of the test or component. While running, QuickTest connects to your Web site or application and performs each operation in your test or component, checking any text strings, objects, or tables you specified. 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 you defined. For more information, see Chapter 21, “Running Tests and Components.”

- Run your test or component to debug it.

You can control your run session to help you identify and eliminate defects in your test or component. You can use the **Step Into**, **Step Over**, and **Step Out** commands to run your test or component step by step. You can also set breakpoints to pause your test or component at pre-determined points. You can view the value of variables in your test or component each time it stops at a breakpoint in the Debug Viewer. For more information, see Chapter 22, “Debugging Tests and Components.”

Analyzing Results

After you run your test or component, you can view the results.

- View the results in the Results window.

After you run your test or component, 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. For more information, see Chapter 23, “Analyzing Test Results.”

- Report defects detected during a run session.

If you have Quality Center installed, you can report the defects you discover to a database. You can instruct QuickTest to automatically report each failed step in your test or component, or you can report them manually from the Test Results window. Quality Center is the Mercury centralized quality solution. For more information, see Chapter 40, “Working with Quality Center.”

Programming in the Expert View

You can use the Expert View tab to view a text-based version of your test or component. The test or component 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 36, “Working with the Expert View.”

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

Managing the Testing Process Using Quality Center

You can use QuickTest together with Quality Center (formerly TestDirector), the Mercury centralized quality solution. You can use Quality Center to create a project (central repository) of manual and automated tests and components, build test cycles, run tests and components, 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.

When you work in QuickTest, you can create and save tests and components directly to your Quality Center project. For more information, see Chapter 40, “Working with Quality Center.”

You can run QuickTest tests or components from Quality Center and then use Quality Center to review and manage the results. You can also use Quality Center with Business Process Testing support to create business process tests, comprised of the components you create in either QuickTest or Quality Center with Business Process Testing support. For more information, see Chapter 41, “Working with Business Process Testing.”

Using the Sample Sites

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

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

You can also use the Mercury Tours sample Windows application available from the QuickTest Professional Start menu program folder.

Modifying License Information

After you install QuickTest, you are prompted to install your license code. You can modify your license at any time to change your license type. You can request a new license on the Mercury Interactive Customer Support Web site. The URL for the License Request Web site is <http://support.mercury.com/license>.

If you purchase one or more external add-ins, you need to install an add-in license. For more information, refer to your add-in documentation.

To modify the license information, refer to the *QuickTest Professional Installation Guide*.

Part I • Starting the Testing Process

2

QuickTest at a Glance

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

This chapter describes:

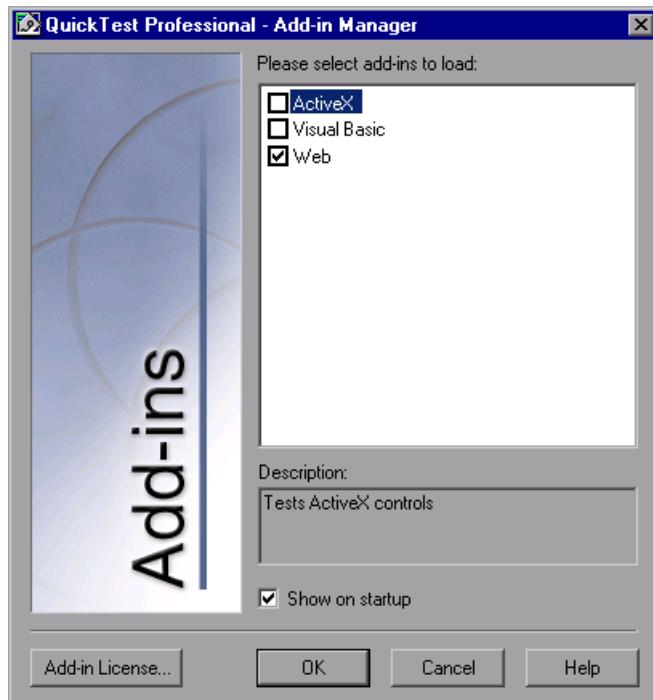
- Starting QuickTest
- The QuickTest Window
- Test Pane
- Active Screen
- Data Table
- Debug Viewer Pane
- Using QuickTest Commands
- Browsing the QuickTest Professional Program Folder

Starting QuickTest



To start QuickTest, choose **Programs > QuickTest Professional > QuickTest Professional** in the **Start** menu.

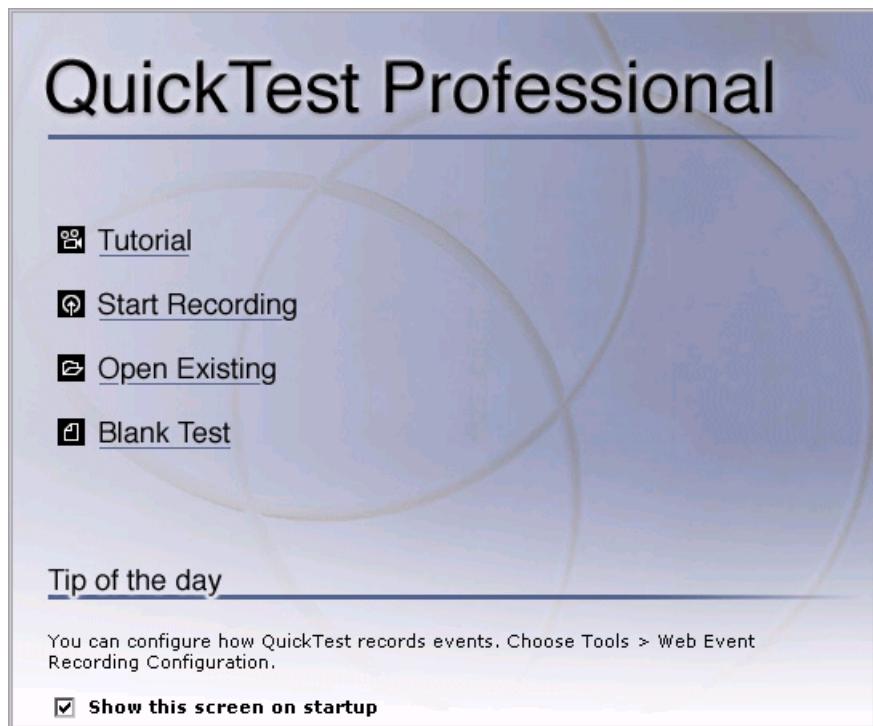
The first time you start QuickTest, the Add-in Manager dialog box opens.



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 about loading add-ins, see “Loading QuickTest Add-ins” on page 732.

Click **OK**. The QuickTest Professional window opens. You can choose to open the QuickTest tutorial, start recording a new test, open an existing test, or open a blank new test.



Tips:

 You can press the ESC key to close the window and open a blank test.

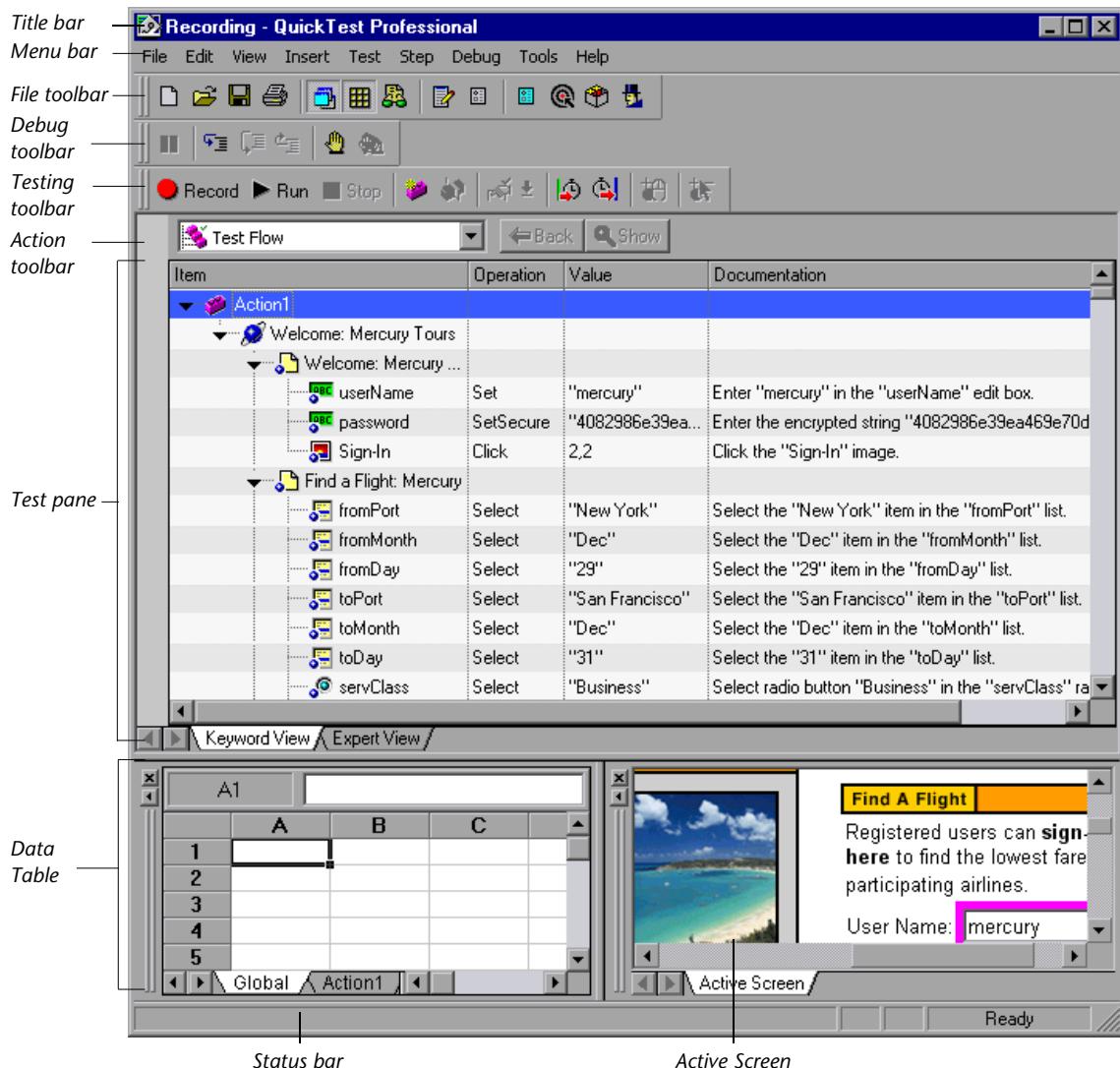
You can click **Tip of the Day** to browse through all the available tips.

If you do not want this window to be displayed the next time you start QuickTest, clear the **Show this screen on startup** check box.

The QuickTest Window

The QuickTest window contains the following key elements:

- **QuickTest title bar**—Displays the name of the currently open test or component.
- **Menu bar**—Displays menus of QuickTest commands.
- **File toolbar**—Contains buttons to assist you in managing your test or component.
- **Testing toolbar**—Contains buttons to assist you in the testing process.
- **Debug toolbar**—Contains buttons to assist you in debugging your test or component (not displayed by default).
- **Action toolbar**—Contains buttons and a list of actions, enabling you to view the details of an individual action or the entire test flow.
- **Test pane**—Contains the Keyword View and Expert View tabs.
- **Active Screen**—Provides a snapshot of your application as it appeared when you performed a certain step during the recording session.
- **Data Table**—Assists you in parameterizing your test or component. For a test, the Data Table contains the **Global** tab and a tab for each action. For a component, the Data Table contains single tab.
- **Debug Viewer pane**—Assists you in debugging your test or component. The Debug Viewer pane contains the **Watch Expressions**, **Variables**, and **Command** tabs (not displayed by default).
- **Status bar**—Displays the status of the QuickTest application.



Test Pane

The Test pane contains two tabs to view your test or component—the Keyword View and the Expert View.

Keyword View

The Keyword View enables you to create and view the steps of your test or component in a keyword-driven, modular, table format. Each step in your test or component is a row in the Keyword View, comprised of individual parts which you can easily modify. You create and modify tests or components 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 or component steps in understandable English.

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

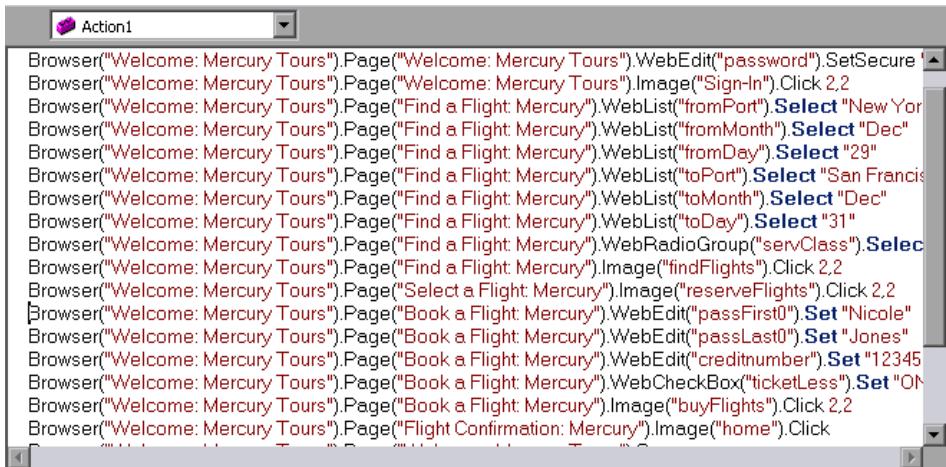
Item	Operation	Value	Documentation
Action1			
Welcome: Mercury Tours			
Welcome: Mercury ...			
userNmae	Set	"mercury"	Enter "mercury" in the "userNmae" edit box.
password	SetSecure	"4082986e39ea469e70dbf8c5..."	Enter the encrypted string "4082986e39ea469e70dbf8c5..."
Sign-In	Click	2,2	Click the "Sign-In" image.
Find a Flight: Mercury			
fromPort	Select	"New York"	Select the "New York" item in the "fromPort" list.
fromMonth	Select	"Dec"	Select the "Dec" item in the "fromMonth" list.
fromDay	Select	"29"	Select the "29" item in the "fromDay" list.
toPort	Select	"San Francisco"	Select the "San Francisco" item in the "toPort" list.
toMonth	Select	"Dec"	Select the "Dec" item in the "toMonth" list.
toDay	Select	"31"	Select the "31" item in the "toDay" list.
servClass	Select	"Business"	Select radio button "Business" in the "servClass" radio button group.
findFlights	Click	2,2	Click the "findFlights" image.

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 or component. For more information on using the Keyword View, see Chapter 15, “Working with the Keyword View.”

Note: The Keyword View replaces the Tree View found in earlier versions of QuickTest. Many of the operations you could perform from the Tree View can be performed in a similar manner from the Keyword View. For example, right-click on a step to access context-sensitive options for that step, such as checkpoint, output value and action-related operations.

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. For more information on using the Expert View, see Chapter 36, “Working with the Expert View.”



```
Action1

Browser("Welcome: Mercury Tours").Page("Welcome: Mercury Tours").WebEdit("password").SetSecure '▲
Browser("Welcome: Mercury Tours").Page("Welcome: Mercury Tours").Image("Sign-In").Click 2,2
Browser("Welcome: Mercury Tours").Page("Find a Flight: Mercury").WebList("fromPort").Select "New York"
Browser("Welcome: Mercury Tours").Page("Find a Flight: Mercury").WebList("fromMonth").Select "Dec"
Browser("Welcome: Mercury Tours").Page("Find a Flight: Mercury").WebList("fromDay").Select "29"
Browser("Welcome: Mercury Tours").Page("Find a Flight: Mercury").WebList("toPort").Select "San Francisco"
Browser("Welcome: Mercury Tours").Page("Find a Flight: Mercury").WebList("toMonth").Select "Dec"
Browser("Welcome: Mercury Tours").Page("Find a Flight: Mercury").WebList("toDay").Select "31"
Browser("Welcome: Mercury Tours").Page("Find a Flight: Mercury").WebRadioGroup("servClass").Select "First"
Browser("Welcome: Mercury Tours").Page("Find a Flight: Mercury").Image("findFlights").Click 2,2
Browser("Welcome: Mercury Tours").Page("Select a Flight: Mercury").Image("reserveFlights").Click 2,2
Browser("Welcome: Mercury Tours").Page("Book a Flight: Mercury").WebEdit("passFirst0").Set "Nicole"
Browser("Welcome: Mercury Tours").Page("Book a Flight: Mercury").WebEdit("passLast0").Set "Jones"
Browser("Welcome: Mercury Tours").Page("Book a Flight: Mercury").WebEdit("creditnumber").Set "12345"
Browser("Welcome: Mercury Tours").Page("Book a Flight: Mercury").WebCheckBox("ticketLess").Set "ON"
Browser("Welcome: Mercury Tours").Page("Book a Flight: Mercury").Image("buyFlights").Click 2,2
Browser("Welcome: Mercury Tours").Page("Flight Confirmation: Mercury").Image("home").Click
```

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 choose **View > Active Screen**.

The Active Screen enables you to easily parameterize object values and insert checkpoints, methods, and output values for any object in the page, even if your application is not available or you do not have a step in your test or component corresponding to the selected object.

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 or component. Therefore, you may need to provide login information to view password-protected resources.

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.

For information on Active Screen customization options, see “Setting Active Screen Options” on page 618.

For information on accessing password-protected resources in the Active Screen of a Web-based application, see “Accessing Password-Protected Resources in the Active Screen” on page 764.

Data Table



In a new test, the Data Table contains one Global tab plus an additional tab for each action, or test step grouping, in your test. In a new component, the Data Table contains a single tab. The Data Table assists you in parameterizing your test or component. To view the Data Table, click the **Data Table** toolbar button or choose **View > Data Table**. The Data Table is a Microsoft Excel-like sheet with columns and rows representing the data applicable to your test or component. For more information, see Chapter 18, “Working with Data Tables.”

Debug Viewer Pane



The Debug Viewer pane contains three tabs to assist you in debugging your test or component—Watch Expressions, Variables, and Command. To view the Debug Viewer pane, click the **Debug Viewer** button or choose **View > Debug Viewer**.

Watch Expressions

The Watch Expressions tab enables you to view the current value of any variable or other VBScript expression.

Variables

The Variables tab enables you to view the current value of all variables that have been recognized up to the last step performed in the run session.

Command

The Command tab enables you to execute a line of script in order to set or modify the current value of a variable or VBScript object in your test or component. When you continue the run session, QuickTest uses the new value that was set in the command.

For more information on using the Debug Viewer pane, see Chapter 22, “Debugging Tests and Components.”

Using QuickTest Commands

You can select QuickTest commands from the menu bar or from a toolbar. Certain QuickTest commands can also be executed by pressing shortcut keys or selecting commands from context-sensitive (right-click) menus.

Choosing Commands on a Menu

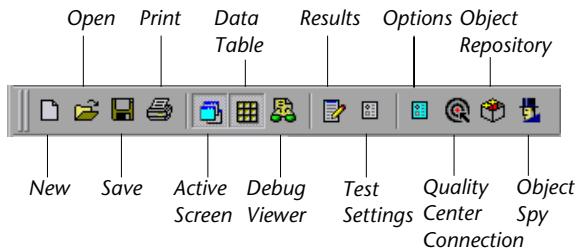
You can choose all QuickTest commands from the menu bar.

Clicking Commands on a Toolbar

You can execute some QuickTest commands by clicking buttons on the toolbars. QuickTest has four built-in toolbars—the File toolbar, the Testing toolbar, the Debug toolbar, and the Action toolbar.

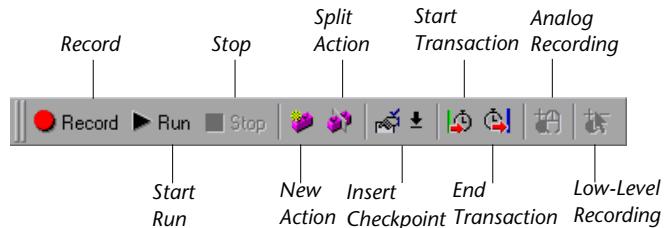
File Toolbar

The File toolbar contains buttons for managing a test or component. For more information on managing your test, see Chapter 5, “Designing Tests.” For more information on managing components, see Chapter 41, “Working with Business Process Testing.” The following buttons are displayed on the File toolbar:



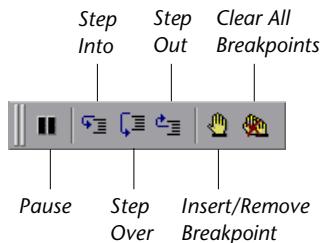
Testing Toolbar

The Testing toolbar contains buttons for the commands used when creating and maintaining your test or component. The following buttons are displayed on the Testing toolbar:



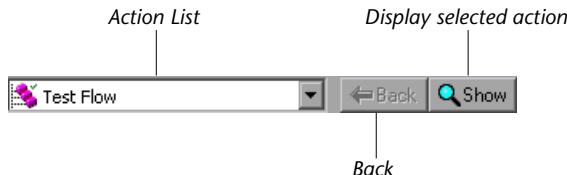
Debug Toolbar

The Debug toolbar contains buttons for the commands used when debugging the steps in your test or component. The following buttons are displayed on the Debug toolbar:



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 Action toolbar is not available for components. The following options are displayed on the Action toolbar:



When you have reusable or external actions in your test, the Action toolbar is always visible. If there are no reusable or external actions in your test, you can choose **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 about actions, see Chapter 17, “Working with Actions.”

Executing Commands Using Shortcut Keys

You can perform some QuickTest commands by pressing shortcut keys. The shortcut keys listed below are displayed on the corresponding menu commands.

You can perform the following File menu commands by pressing the corresponding shortcut keys:

Command	Shortcut Key	Function
New Test	CTRL+N	Creates a new test.
Open Test	CTRL+O	Opens an existing test.
Business Component > New	CTRL+SHIFT+N	Creates a new component.
Business Component > Open	CTRL+SHIFT+O	Opens an existing component.
Business Component > Edit Template	CTRL+SHIFT+E	Opens the component template of the current Quality Center project for editing.
Save	CTRL+S	Saves the active test or component.
Export to Zip File	CTRL+ALT+S	Creates a zip file of the active test or component.

Command	Shortcut Key	Function
Import from Zip File	CTRL+ALT+O	Imports a test or component from a zip file.
Print	CTRL+P	Prints the active test or component.

You can perform the following Edit menu commands by pressing the corresponding shortcut keys:

Command	Shortcut Key	Function
Cut	CTRL+X	Removes the selection from your test or component (Expert View only).
Copy	CTRL+C	Copies the selection from your test or component.
Paste	CTRL+V	Pastes the selection to your test or component.
Delete	DEL	Deletes the selection from your test or component.
Undo	CTRL+Z	Reverses the last command or deletes the last entry you typed (Expert View only).
Redo	CTRL+Y	Reverses the action of the Undo command (Expert View only).
 Rename Action	F2	Changes the name of an action.
Find	CTRL+F	Searches for a specified string (Expert View only).
Replace	CTRL+H	Searches and replaces a specified string (Expert View only).
Go To	CTRL+G	Moves the cursor to a particular line in the test or component (Expert View only).

Command	Shortcut Key	Function
Bookmarks	CTRL+B	Creates bookmarks in your script for easy navigation (Expert View only).
Complete Word	CTRL+SPACE	Completes the word when you type the beginning of a VBScript method or object (Expert View only).
Argument Info	CTRL+SHIFT+SPACE	Displays the syntax of a method (Expert View only).
Apply “With” to Script	CTRL+W	Generates With statements for the action displayed in the Expert View (Expert View only).
Remove “With” Statements	CTRL+SHIFT+W	Converts any With statements in the action displayed in the Expert View to regular (single-line) VBScript statements (Expert View only).

You can perform the following Insert menu commands by pressing the corresponding shortcut keys:

Command	Shortcut Key	Function
Checkpoint > Standard Checkpoint	F12	Creates a standard checkpoint for an object or a table.
Output Value > Standard Output Value	CTRL+F12	Creates a standard output value for an object or a table.
Step > Step Generator	F7	Opens the Step Generator.
New Step	F8	Inserts a new step in the Keyword View (Keyword View only).
New Step After Block	SHIFT+F8	Inserts a new step after a conditional or loop block in the Keyword View (Keyword View only).

You can execute the following Test or Component menu commands by pressing the corresponding shortcut keys:

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	Stops the recording or run session.
Analog Recording	CTRL+SHIFT+F4	Starts/ends analog recording mode.
Low Level Recording	CTRL+SHIFT+F3	Starts/ends low level recording mode.

You can perform the following Step menu commands by pressing the corresponding shortcut keys, depending on the selected item:

Command	Shortcut Key	Function
Object Properties	CTRL+ENTER on a test object	Opens the Object Properties dialog box of a selected object.
Value Configuration Options	CTRL+F11 on a value in the Keyword View	Opens the Value Configuration Options dialog box for a selected value (Keyword View only).
Step > Comment	CTRL+ENTER on a comment	Opens the Comment Properties dialog box for a selected comment (Keyword View only).
Step > Report	CTRL+ENTER on a report step	Opens the Report Properties dialog box for a selected report step (Keyword View only).

Part I • Starting the Testing Process

You can perform the following Debug menu commands by pressing the corresponding shortcut keys:

Command	Shortcut Key	Function
Pause	PAUSE	Stops the run session after the statement has been executed. The run session can be resumed from this point.
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 .)
Insert/Remove Breakpoint	F9	Sets or clears a breakpoint in the test or component.
Clear All Breakpoints	CTRL+SHIFT+F9	Deletes all breakpoints in the test or component.

You can perform the following Data Table menu commands by 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.

Command	Shortcut Key	Function
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.
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.

Command	Shortcut Key	Function
Insert Multi-line value	CTRL+F2 while editing a cell	Opens the Cell Text dialog box for multi-line value editing.
Activate next/previous data sheet	CTRL+PAGE UP / CTRL+PAGE DOWN	Activates the next or previous sheet (global or action) in the Data Table.

You can perform the following special options using shortcut keys only:

Option	Shortcut Key	Function
Switch between Keyword View and Expert View	CTRL+TAB	Toggles between the Keyword View and Expert View.
Open context menu	SHIFT+F10, or press the Application Key () <i>[Microsoft Natural Keyboard only]</i>	Opens the context menu for the selected step data cell in the Data Table.
Expand all branches	*	Expands all branches in the Keyword View (Keyword View only).
Expand branch	+	Expands the selected item branch and all branches below it in the Keyword View (Keyword View only).
Collapse branch	-	Collapses the selected item branch and all branches below it in the Keyword View (Keyword View only).
Open the Item or Operation list	SHIFT+F4 or ENTER, 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 (Keyword View only).

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**):

- **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 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 virtually every QuickTest feature and capability. The QuickTest 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 Help**—Opens a comprehensive help file containing the QuickTest Professional User's Guide, the corresponding user's guide for each installed add-in (if any), the QuickTest Professional Object Model Reference (including the relevant sections for installed add-ins), and the VBScript Reference.
 - **Tutorial**—Opens the QuickTest Professional tutorial, which teaches you basic QuickTest skills and shows you how to start testing your applications.
 - **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. Refer to the *QuickTest Professional Tutorial* for more information.

- **Tools**—Contains the following utilities and tools that assist you with the testing process:
 - **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. For more information, see “Inserting Encoded Passwords into Method Arguments and Data Table Cells” on page 416.
 - **Remote Agent**—Activates the QuickTest Remote Agent, which determines how QuickTest behaves when a test or component is run by a remote application such as Quality Center. For more information, see Chapter 40, “Working with Quality Center”.
 - **Test Batch Runner**—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 527.
 - **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 590.
- **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 QuickTest Professional Readme, which provides the latest news and information about QuickTest Professional.
- **Test Results Viewer**—Opens the Test Results window, which enables you to select a test or component and view information about the steps performed during the run session. For more information, see “Understanding the Test Results Window” on page 542.
- **Uninstall QuickTest Professional**—Uninstalls QuickTest Professional and all of its components, including core and external add-ins. Refer to the *QuickTest Professional Installation Guide* for more information.

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*, and explains how to view the available methods for an object and the corresponding syntax, so that you can easily add statements to your script in the Expert View.

This chapter describes:

- ▶ About Understanding the Test Object Model
- ▶ Applying the Test Object Model Concept
- ▶ Viewing Object Properties Using the Object Spy
- ▶ Viewing Object Methods and Method Syntax Using the Object Spy

About Understanding the Test Object Model

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

In the following narrative you will be introduced to the concepts related to the test object model and how QuickTest uses the information it gathers to test your application.

Understanding How QuickTest Learns Objects While Recording

QuickTest learns objects just as you would.

For example, suppose as part of an experiment, Jonny 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. Jonny 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, Jonny 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 Jonny and points out one of three children sitting on a picnic blanket. Jonny 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, Jonny 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 during the recording process.

First, it “looks” at the object on which you are recording and stores it as a *test object*, determining in which test object class it fits. Just as Jonny immediately checked whether the item was a person, animal, plant, or thing, 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, for each test object class, QuickTest has a list of *mandatory* properties that it always learns; similar to the list of characteristics that Jonny planned to learn before seeing the picture. When you record on an object, QuickTest 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 it is not, QuickTest adds *assistive* properties, one by one, to the description, until it has compiled a unique description; like when Jonny 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 Jonny 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, Jonny 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 Jonny 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 while recording. It expects to find a perfect match for both the mandatory and any assistive properties it used to create a unique description while recording. As long as the object in the application does not change significantly, the description learned during recording 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 Jonny's experiment. In this phase, the tester shows Jonny 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. Jonny 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, Jonny 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. Jonny 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.

Once 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 recorded description is no longer accurate. Even if the values of your test object properties change, QuickTest's TestGuard technology maintains your test or component's reusability by identifying the object using Smart Identification. For more information on Smart Identification, see Chapter 33, "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 or components 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 properties that can uniquely identify objects of that class and a set of relevant methods that QuickTest can record for it.

A *test object* is an object that QuickTest creates in the test or component to represent the actual object in your application. QuickTest stores information about 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 Web site or application on which methods are performed during the run session.

When you perform an operation on your application while recording, QuickTest:

- identifies the QuickTest test object class that represents the object on which you performed the operation and creates the appropriate test object
- reads the current value of the object's properties in your application and stores the list of properties and values with the test object
- chooses a unique name for the object, generally using the value of one of its prominent properties
- records the operation that you performed on the object using the appropriate QuickTest test object method

For example, suppose you click on a **Find** button with the following HTML source code:

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

QuickTest identifies the object that you clicked as a **WebButton** test object. It creates a WebButton object with the name Find, and records the following properties and values for the Find WebButton:

Property	Value
html tag	INPUT
name	Find
type	reset

It also records that you performed a **Click** method on the WebButton.

QuickTest displays your step in the Keyword View like this:

Item	Operation	Documentation
▼ Action1		Call the Action1 action.
▼ Mercury Interactive		
▼ Mercury Interactive	Click	Click the "Find" button.

QuickTest displays your step in the Expert View like this:

```
Browser("Mercury Interactive").Page("Mercury Interactive").  
WebButton("Find").Click
```

When you run a test or component, QuickTest identifies each object in your application by its test object class and its *description*—the set of test object 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 Find 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 **Find**. When it finds the object, it performs the Click method on it.

Understanding Test Object Descriptions

For each object class, QuickTest learns a set of properties when it records and it uses this description to identify the object when it runs the test or component.

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 on which you record an operation.

The screenshot shows the QuickTest interface with two main windows. The top window is the 'Object Properties' dialog box, which displays the 'test object name' (Sign-In) and 'test object class' (Image). It also shows a 'Properties' table with three rows: 'alt' (Value: Sign-In), 'html tag' (Value: INPUT), and 'image type' (Value: Image Button). The bottom window is the 'Object Repository' table, showing a hierarchy of actions and their properties. The 'Action1' action has a child 'Welcome: Mercury Tours' step, which has a child 'Welcome: Mercury Tours' step. This third step has three properties: 'user Name' (Value: mercury), 'password' (Value: 405d7bdbe1b2...), and 'Sign-In' (Value: Click).

Item	Operation	Documentation
Action1		Call the Action1 action.
Welcome: Mercury Tours		
Welcome: Mercury Tours	Set	Enter "mercury" in the "userName" edit box.
	SetSecure	Enter the encrypted string "405d7bdbe1b2..."
	Click	Click the "Sign-In" image.

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 or component 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 record descriptions of the objects in your application, and you can enable and configure the smart identification mechanism. For more information, see Chapter 33, “Configuring Object Identification.”

Understanding Test Object and Run-Time Object Properties and Methods

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

Similarly, test object methods are methods that QuickTest recognizes and records when they are performed on an object while you are recording, and that QuickTest performs when your test or component runs. Run-time object methods are the methods of the object in your application as defined by the object creator. You can access and perform run-time object methods using the **Object** property.

For information about activating run-time methods using the **Object** property, see “Retrieving and Setting Test Object Property Values” on page 898.

- Each test object method you perform while recording is recorded as a separate step in your test or component. When you run your test or component, QuickTest performs the recorded test object method on the run-time object.
- Test object properties are the properties whose values are captured from the objects in your Web site or application when you record your test or component. QuickTest uses the values of these properties to identify run-time objects in your application during a run session.

- Property values of objects in your application may change dynamically each time your application opens, or based on certain conditions. To make the test object property values match the property values of the run-time object, you can modify test object properties manually while designing your test or component or using **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 (this option is not applicable to components).

For more information on modifying object properties, see Chapter 4, “Managing Test Objects.” For more information on parameterization, see Chapter 12, “Parameterizing Values.” For more information on regular expressions, see “Understanding and Using Regular Expressions” on page 290.

- You can view or modify the test object property values that are stored with your test or component in the Object Properties or Object Repository dialog box. You can view the current test object property values of any object on your desktop using the Properties tab of the Object Spy.

For information about the Object Properties and Object Repository dialog boxes, see “Modifying Test Object Properties While Editing Your Test or Component” on page 60.

For information about viewing test object property values using the Object Spy, see “Viewing Object Properties Using the Object Spy” on page 42.

- You can view the syntax of the test object methods as well as the run-time methods of any object on your desktop using the Methods tab of the Object Spy.

For more information, see “Viewing Object Methods and Method Syntax Using the Object Spy” on page 46.

- You can retrieve or modify property values of the test object during the run session by adding **GetTObjectProperty** and **SetTObjectProperty** statements in the Keyword View or Expert View. You can retrieve property values of the run-time object during the run session by adding **GetROProperty** statements. For more information, see “Retrieving and Setting Test Object Property Values” on page 898.
- If the available test object methods or properties for an object do not provide the functionality you need, you can access the internal methods and properties of any 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 Run-Time Object Properties and Methods” on page 899.

For more information about test object methods and properties refer to the *QuickTest Professional Object Model Reference*.

Viewing Object Properties Using the Object Spy

Using the Object Spy, you can view the properties of any object in an open application. You use the Object Spy pointer to point to an object. The Object Spy displays the selected object’s hierarchy tree and its properties and values in the Properties tab of the Object Spy dialog box.

To view object properties:

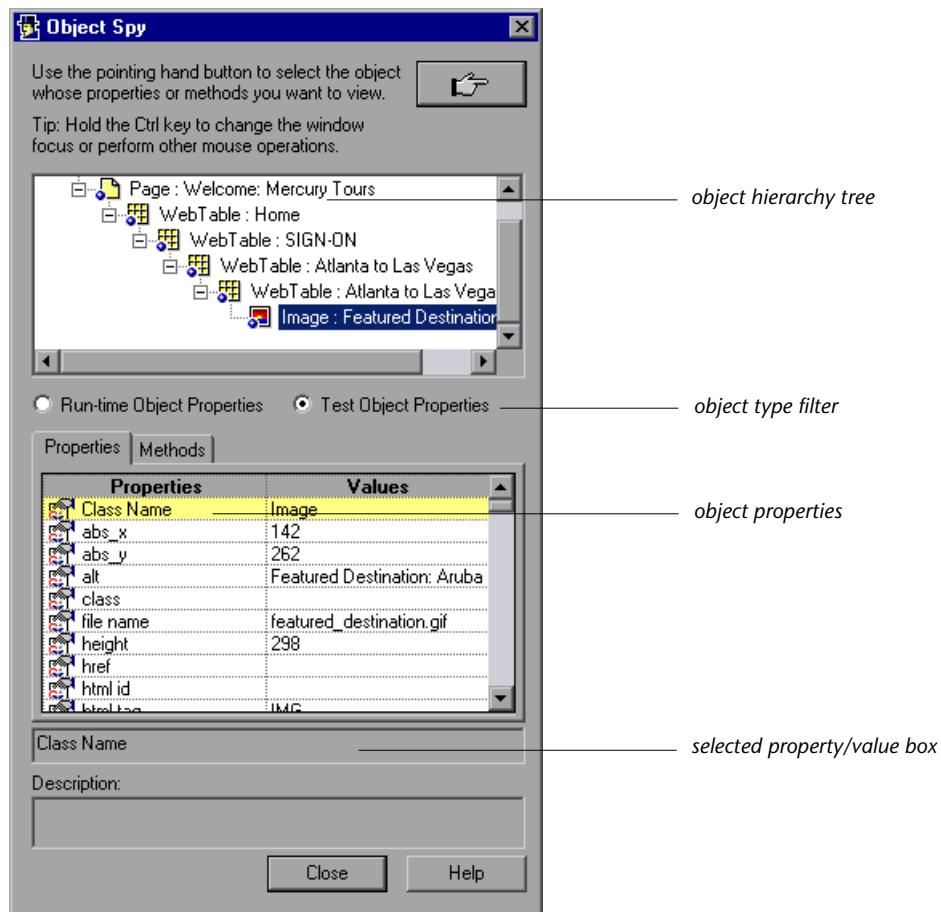
- 1 Open your browser or application to the page containing the object on which you want to spy.
- 2  Choose **Tools > Object Spy** or click the **Object Spy** toolbar button to open the Object Spy dialog box and display the **Properties** tab. Alternatively, click the **Object Spy** button from the Object Repository dialog box. For more information on the Object Repository dialog box, see “Understanding the Object Repository Dialog Box” on page 51.
- 3  In the Object Spy dialog box, click the pointing hand. Both QuickTest and the Object Spy are minimized so that you can point to and click on any object in the open application.

Note: 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. The window comes into the foreground. You can now point and click on the object you want. You can configure the length of time required to bring a window into the foreground in the General tab of the Options dialog box. For more information, see Chapter 24, “Setting Global Testing Options.” You can also hold the CTRL key to change the window focus. Note that pressing the CTRL key does not enable you to select an application from the Windows taskbar, therefore you must make sure that the window you want to access is not minimized.

- 4 If the object on which you want to spy can only be displayed by performing an event (such as a right-click or a mouse-over to display a context menu), hold the CTRL key. The pointing hand temporarily turns into a standard arrow and you can perform the event. When the object on which you want to spy is displayed, release the CTRL key. The arrow becomes a pointing hand again.
-

Note: Pressing the CTRL key does not enable you to select an application from the Windows taskbar, therefore you must make sure that the window you want to access is not minimized.

- 5 Click the object for which you want to view properties. The Object Spy returns to focus and displays the object hierarchy tree and the properties of the object that is selected within the tree.



- 6** To view the properties of the test object, click the **Test Object Properties** radio button. To view the properties of the run-time object, click the **Run-Time Object Properties** radio button.
-

Tip: You can use the **Object** property to retrieve the values of the run-time properties displayed in the Object Spy. For more information, see “Retrieving Run-Time Object Properties” on page 900.

You can use the **GetTOPProperty** and **SetTOPProperty** methods to retrieve and set the value of test object properties for test objects in your test or component. You can use the **GetROPProperty** to retrieve the current property value of the objects in your application during the run session. For more information, see “Retrieving and Setting Test Object Property Values” on page 898.

- 7** If you want to view properties for another object within the displayed tree, click the object in the tree.
- 8** If you want to copy an object property or value to the clipboard, click the property or value. The value is displayed in the selected property/value box. Highlight the text in the selected property/value box and use **CTRL + C** to copy the text to the clipboard or right-click the highlighted text and choose **Copy** from the menu.
-

Note: If the value of a property contains more than one line, the Values cell of the object properties list indicates **multi-line value**. To view the value, click the Values cell. The selected property/value box displays the value with delimiters indicating the line breaks.

Viewing Object Methods and Method Syntax Using the Object Spy

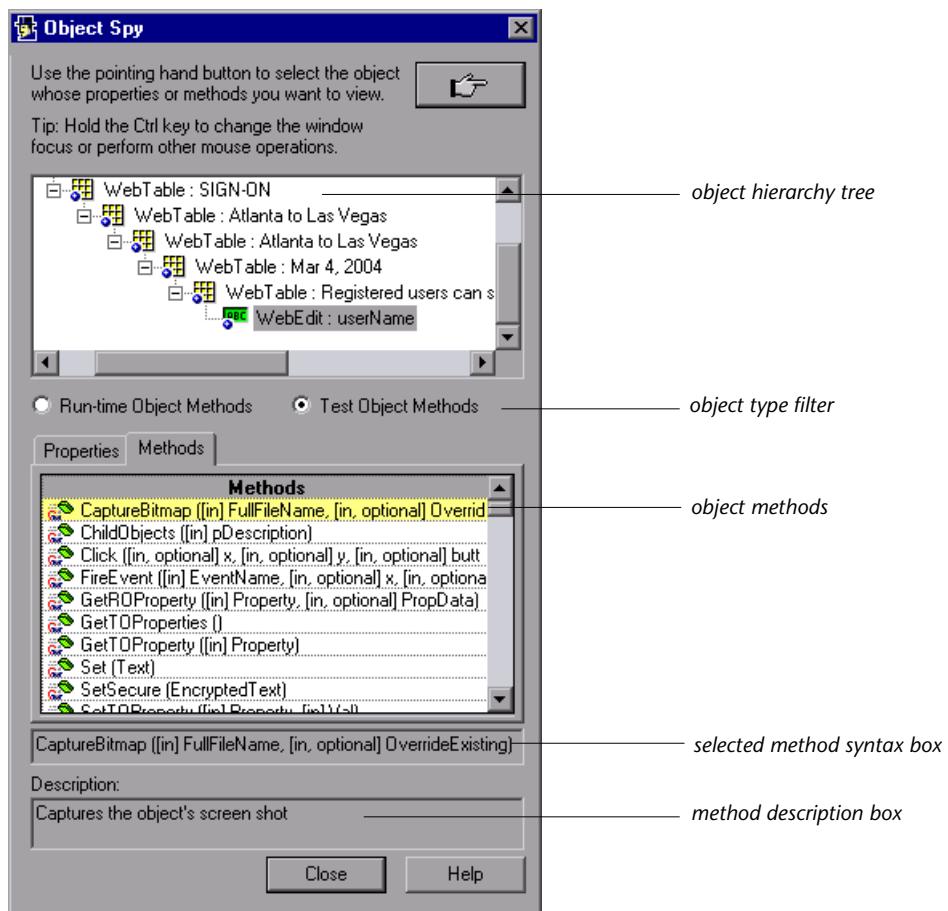
In addition to viewing object properties, the Object Spy also enables you to view both the run-time object methods and the test object methods associated with an object and to view the syntax for a selected method. You use the Object Spy pointer to point to an object. The Object Spy displays the object hierarchy tree and the run-time object methods or test object methods associated with the selected object in the Methods tab of the Object Spy dialog box.

To view object methods:

- 1 Open your browser or application to the page containing the object on which you want to spy.
- 2 Choose **Tools > Object Spy** or click the **Object Spy** toolbar button to open the Object Spy dialog box. Alternatively, click the **Object Spy** button from the Object Repository dialog box. For more information on the Object Repository dialog box, see “Understanding the Object Repository Dialog Box” on page 51.
- 3 Click the **Methods** tab.
- 4 Click the pointing hand. Both QuickTest and the Object Spy are minimized so that you can point to any object on the open application.

Note: If the object you want is partially hidden by another window, hold the pointing hand over the partially hidden window for a few seconds. The window comes into the foreground. You can now point and click on the object you want. You can configure this option in the Options dialog box. For more information, see Chapter 24, “Setting Global Testing Options.” You can also hold the CTRL key to change the window focus. Note that pressing the CTRL key does not enable you to select an application from the Windows taskbar, therefore you must make sure that the window you want to access is not minimized.

- 5 If the object on which you want to spy can only be displayed by performing an event (such as a right-click or a mouse-over to display a context menu), hold the CTRL key. The pointing hand temporarily turns into a standard arrow and you can perform the event. When the object on which you want to spy is displayed, release the CTRL key. The arrow becomes a pointing hand again.
- 6 Click the object for which you want to view the associated methods. The Object Spy returns to focus and displays the object hierarchy tree and the *run-time object* or *test object* methods associated with the object that is selected within the tree.



- 7** To view the methods of the test object, click the **Test Object Methods** radio button. To view the methods of the run-time object, click the **Run-Time Object Methods** radio button.
-

Tip: You can use the **Object** property to activate the run-time object methods displayed in the Object Spy. For more information, see “Activating Run-Time Object Methods” on page 900.

- 8** If you want to view methods for another object within the displayed tree, click the object on the tree.
- 9** If you want to copy the syntax of a method to the clipboard, click the method in the list. The syntax is displayed in the selected method syntax box. Highlight the text in the selected method syntax box and use CTRL + C to copy the text to the clipboard, or right-click the highlighted text and choose **Copy** from the menu.

4

Managing Test Objects

This chapter explains how to manage and maintain the test objects in your test or component. It describes how to modify test object properties and how to modify the way QuickTest identifies an object, which is useful when working with objects that change dynamically. It also describes how objects can be added to or deleted from your test or component.

This chapter describes:

- About Managing Test Objects
- Understanding the Object Repository Dialog Box
- Understanding the Object Properties Dialog Box
- Modifying Test Object Properties While Editing Your Test or Component
- Working with Test Objects During a Run Session
- Modifying Object Descriptions
- Adding Objects to the Object Repository
- Deleting an Object from the Object Repository

About Managing Test Objects

QuickTest identifies objects in your application based on a set of test object properties. It stores the object data it learns in the *object repository*.

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 or component may fail. Thus, when the property values of objects in your application change, you should modify the corresponding test object property values so that you can continue to use your existing tests or components.

Note: In some cases, the Smart Identification mechanism may enable QuickTest to identify an object, even when some of its property values change. However, if you know about property value changes for a specific object, you should try to correct the object definition so that QuickTest can identify the object from its basic object description. For more information on the smart identification mechanism, see Chapter 33, “Configuring Object Identification.”

There are several methods for modifying test object properties. Choose the method that best fits your needs:

- You can manually change a test object property value to match a new static property of an object in your application.
- You can use the **SetTOProperty** method to modify test object properties during a run session without changing the property values in the object repository.
- You can modify the set of properties that QuickTest uses to identify the object, so that it will be able to identify an object even when some of its properties change dynamically.

-  You can parameterize a test object property with a Data Table parameter if you expect the property value to change in a predictable way with each iteration of the test.
- You can use regular expressions to identify an object based on conditions or patterns you define.

This chapter includes information on the first three options above. You can make the most of these modifications in the Object Properties dialog box or in the Object Repository dialog box. For more information on parameterizing object properties, see Chapter 12, “Parameterizing Values.” For information on using regular expressions, see “Understanding and Using Regular Expressions” on page 290.

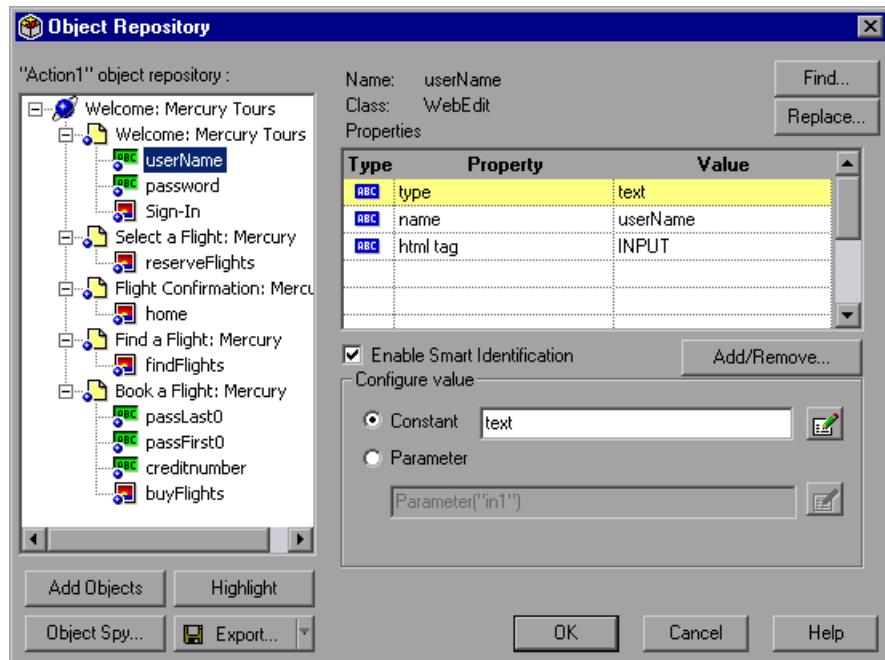
You can save your objects either in a *shared object repository* or in *action object repository*. In shared object repository mode, you can use one object repository file for multiple tests or components.  In object repository per-action mode, QuickTest automatically creates an object repository file for each action in your test. Object repository per-action mode is not available for components.

The information in this chapter on modifying test objects is relevant to both the shared object repository and the action object repository. For more information on the shared object repository and the action object repository, see Chapter 34, “Choosing the Object Repository Mode.”  For more information on organizing your test using actions, see Chapter 17, “Working with Actions.”

Understanding the Object Repository Dialog Box

The Object Repository dialog box displays a tree of all objects in the current component or in the current action or entire test (depending on the object repository mode you choose when you create your test). You can use the Object Repository dialog box to view or modify the test object description of any test object in the repository or to add new objects to your repository.

For more information on viewing and modifying test object properties, see “Modifying Test Object Properties While Editing Your Test or Component” on page 60.



Note: When working in object repository per-action mode, the object repository stores objects on a per-action basis. Thus, if the same test object is used in several steps within the same action, you need to modify the object's properties only one time. If the object is used again in another action, however, you need to update the object's properties in that action as well. For more information about actions, see Chapter 17, “Working with Actions.” For more information on choosing an object repository mode, see Chapter 34, “Choosing the Object Repository Mode.”

Even when steps containing a test object are deleted from your test or component, the objects remain in the object repository. If you delete all occurrences of an object from your action or entire test (in object repository per-action mode) or from all tests or components using the shared object repository (in shared object repository mode), you can also choose to delete the object from the object repository.

Identifying the Object

The top part of the dialog box displays information about the object:

Information	Description
Name	The name that QuickTest assigns to the test object.
Class	The class of the object.
Find	Opens the Find dialog box, where you can find a property or value that occurs several times in the object repository. For more information, see “Finding Test Object Properties” on page 63.
Replace	Opens the Replace Property Values dialog box, where you can replace a property value that occurs several times in the object repository. For more information, see “Replacing Test Object Property Values” on page 64.

Viewing the Object's Properties

The default 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
Type	<p>The  icon indicates that the value of the property is currently a constant.</p> <p>The  icon indicates that the value of the property is currently a test, action, or component parameter.</p> <p>The  icon indicates that the value of the property is currently a Data Table parameter.</p> <p>The  icon indicates that the value of the property is currently an environment variable parameter.</p> <p>The  icon indicates that the value of the property is currently a random number parameter.</p>
Property	The name of the property.
Value	The value of the property.
Enable Smart Identification	<p>Indicates whether or not QuickTest uses Smart Identification to identify this object during the run session if it is not able to identify the object using the test object description. Note that this option is available only if Smart Identification properties are defined for the object's class in the Object Identification dialog box. For more information, see "Configuring Smart Identification" on page 803.</p> <p>Note:  When you select Disable Smart Identification during the run session in the Run tab of the Test Settings dialog box, this option is disabled, although the setting is saved. When you clear the Disable Smart Identification during the run session check box, this option returns to its previous on or off setting after the run session. For more information, see "Defining Run Settings for Your Test" on page 665.</p>
Add/Remove	Opens the Add/Remove Properties dialog box which lists the properties that can be used to identify the object. For more information, see "Modifying Object Descriptions" on page 67.

Modifying the Property Values

In the **Configure value** area, you can define a property value as a **Constant** or **Parameter**. For more information on modifying values, see “Setting Values in the Configure Value Area” on page 286.

Adding or Viewing New Objects and Saving the Object Repository

From the Object Repository dialog box, you can add a new object to your repository, locate an existing object in your application, and use the Object Spy. You can also export a per-action object repository or save a shared object repository.

Option	Description
Add Objects	Adds the object selected with the pointing hand to the current object repository. For more information see “Adding Objects to the Object Repository” on page 73.
Highlight	<p>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.</p> <p>For example, to locate the User Name edit box in a Web page, you can open the page in the Web browser and then select the User Name test object in the Object Repository dialog box. When you click Highlight, the User Name edit box in your browser is framed in the Web page and flashes several times.</p> <p>Note: Both the frame and the flashing behavior are temporary.</p>
Object Spy	Opens the Object Spy dialog box, enabling you to view the properties of any object in an open application. For more information see “Viewing Object Properties Using the Object Spy” on page 42.
 Export	<p>(Available only for tests and when working in per-action repository mode.)</p> <p>Saves all the object properties and values from a per-action object repository to a separate file for use as a shared object repository in another test.</p>

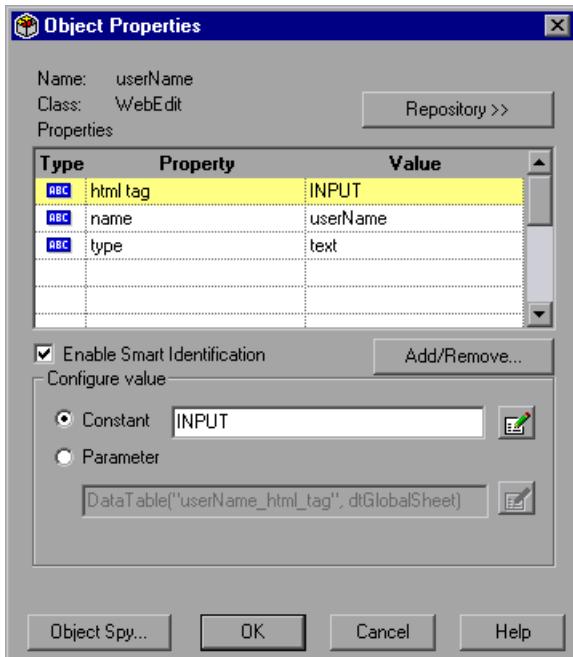
Option	Description
Save	<p>(Available only when working in shared object repository mode.)</p> <p>Saves all the object properties and values in a shared object repository file. This option allows you to save the current shared object repository without saving any changes made to the open test or component.</p> <p>To save your current shared object repository with a different name, click the arrow next to the Save button and select Save As (see below).</p>
Save As	<p>(Available only when working in shared object repository mode.)</p> <p>Saves all the object properties and values in the current shared object repository to a different shared object repository file. You can overwrite an existing shared object repository file or you can create a new one.</p> <p>When you choose to save your shared object repository with the Save As option, QuickTest gives you the option to use the new file as the shared object repository for your open test component or to continue working with the old one. For more information and guidelines, see “Choosing a Shared Object Repository File” on page 831.</p>

Note: Clicking **Save** for a shared object repository cannot be cancelled by clicking **Cancel**. Clicking **Cancel** cancels only those changes made after the last save within the same object repository session.

Clicking **OK** to close the Object Repository dialog box enables the changes you make to the repository to take effect while you are working in your open test or component. The changes you make are not saved to a per-action object repository until you save the test or to a shared object repository until you save the test or component or click **Save**.

Understanding the Object Properties Dialog Box

The Object Properties dialog box accesses test object information from the object repository and displays the properties and values of the test object in the selected step.



You can use the Object Properties dialog box to view or modify the test object description of the selected object. For more information on viewing and modifying test object properties, see “Modifying Test Object Properties While Editing Your Test or Component” on page 60.

Identifying the Object

The top part of the dialog box displays information about the object:

Information	Description
Name	The name that QuickTest assigns to the test object.
Class	The class of the object.
Repository	Opens the Object Repository dialog box and displays a tree of all objects in the current action or component, with the selected object highlighted in the tree. For more information about the Object Repository dialog box, see "Understanding the Object Repository Dialog Box" on page 51.

Viewing the Object's Properties

The default 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
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, action, or component 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 value of the property.

Pane Element	Description
Enable Smart Identification	<p>Indicates whether or not QuickTest uses Smart Identification to identify this object during the run session if it is not able to identify the object using the test object description. Note that this option is available only if Smart Identification properties are defined for the object's class in the Object Identification dialog box. For more information, see "Configuring Smart Identification" on page 803.</p> <p>Note: When you select Disable Smart Identification during the run session in the Run tab of the Test Settings dialog box, this option is disabled, although the setting is saved. When you clear the Disable Smart Identification during the run session check box, this option returns to its previous on or off setting after the run session. For more information, see "Defining Run Settings for Your Test" on page 665.</p>
Add/Remove	<p>Opens the Add/Remove Properties dialog box which lists the properties that can be used to identify the object. For more information, see "Modifying Object Descriptions" on page 67.</p>

Modifying the Property Values

In the **Configure value** area, you can define a property value as a **Constant** or **Parameter**. For more information on modifying values, see "Setting Values in the Configure Value Area" on page 286.

Using the Object Spy

You can view the properties and values of an object in any open application using the Object Spy. You can open the Object Spy from the Object Properties dialog box by clicking **Object Spy** at the bottom of the dialog box. For more information, see "Viewing Object Properties Using the Object Spy" on page 42.

Modifying Test Object Properties While Editing Your Test or Component

As Web sites and applications change, the property values of the steps in your test or component may also need to change.

Suppose an object in your application changes. If that object is part of your test or component, you should modify its values so that QuickTest can continue to identify it. For example, if the MyCompany Web site has a “Contact Us” hypertext link and then the text string in this link is changed to “Contact MyCompany,” you need to update your test or component so that QuickTest can continue to identify the link properly.

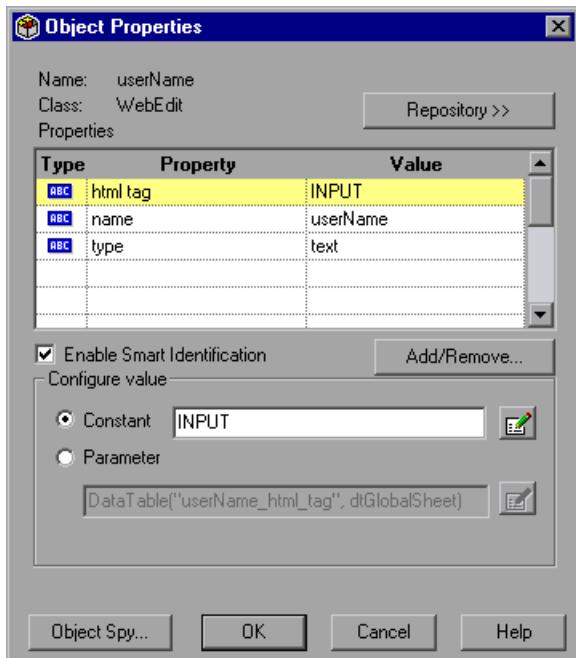
You can modify the object by modifying one or more of the object’s property values in the Object Repository or Object Properties dialog box or by changing the set of properties used to identify that object.

To modify an object property in your test or component:

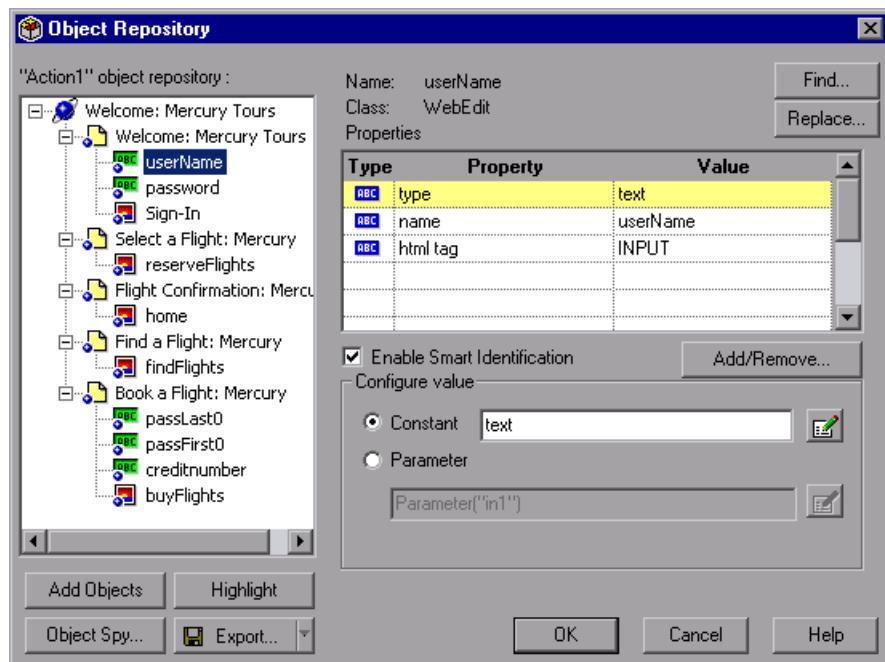
- 1 Right-click the step containing the object that changed, and choose **Object Properties** or choose **Step > Object Properties** from the menu bar.

Tip: You can also right-click an object in the Active Screen and choose **View/Add Object**. If the location you clicked is associated with more than one object, the Object Selection - Object Properties View dialog box opens. Select the object whose properties you want to modify, and click **OK**.

The Object Properties dialog box opens and displays the properties QuickTest uses to identify the object.



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



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



- 2 Highlight the property and value to modify.
- 3 Select **Constant** or **Parameter** in the **Configure value** area. For more information, see “Setting Values in the Configure Value Area” on page 286.
- 4 Click **OK** to close the dialog box.

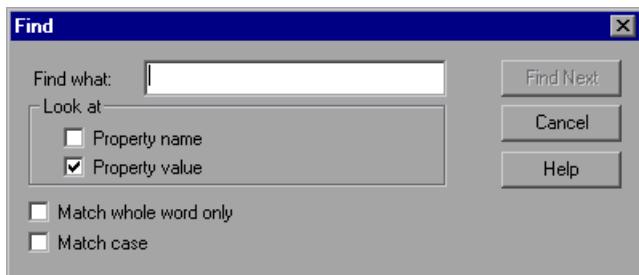
Finding Test Object Properties

You can use the **Find** button in the Object Repository dialog box to find a property or value that occurs several times in the object repository.

To find a property or value in the object repository:

- 1** Right-click an object with the property or value you want to find in the Keyword View or Active Screen and choose **Object Properties** (or **View/Add Object** in the Active Screen), and then click the **Repository** button, choose **Tools > Object Repository**, or click the **Object Repository** toolbar button.

- The Object Repository dialog box opens.
- 2** Right-click the object in the repository tree and choose **Find**, or click the **Find** button. The Find dialog box opens.



- 3** Enter the text for the property or value you want to find. Select **Property name**, **Property value**, or both.
 - If you want the search to find only complete words that exactly match the single word you entered, select **Match whole word only**.
 - If you want the search to distinguish between upper and lower case letters, select **Match case**.

Click **Find Next**. The first instance of the searched word is displayed.

- 4** To find the next instance, click **Find Next** again.

Replacing Test Object Property Values

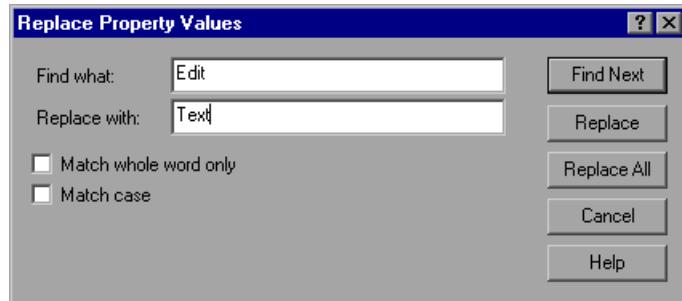
You can use the **Replace** button in the Object Repository dialog box to find and replace property values that occur several times in the object repository.

Note: You cannot replace property names. You also cannot replace values in a read-only test, action, or component.

To replace a value in the object repository:

- 1 Right-click an object in the Keyword View or Active Screen with the property or value you want to find and choose **Object Properties** (or **View/Add Object** from the Active Screen) to open the Object Properties dialog box, and then click the **Repository** button, or choose **Tools > Object Repository**, or click the **Object Repository** toolbar button. The Object Repository dialog box opens.

- 2 Right-click the object in the repository tree and choose **Replace**, or click the **Replace** button. The Replace Property Values dialog box opens.



- 3 In the **Find what** box, enter the property value you want to find, and in the **Replace with** box, enter the modified text for the found property value.
 - If you want the search to find only complete words that exactly match the single word you entered, select **Match whole word only**.
 - If you want the search to distinguish between upper and lower case letters, select **Match case**.

- 4** To individually find and replace each instance of the word(s) you are searching for one at a time, click **Find Next**. When an instance is found, click **Replace**. Then click **Find Next** again to find the next instance.

To replace all instances of the word you are searching for with the new value, click **Replace All**.

Modifying Test Object Names

When an object changes in your application, or for any other reason you are not satisfied with the current name of a test object, you can change the name that QuickTest assigns to the stored object. You modify the test object's name in the object repository. When you modify the name of an object, the name is automatically updated in both the Keyword View and the Expert View for all occurrences of the object.

If you are working with a shared object repository, your change applies to all occurrences of the object in the test or component. When you open another test or component that uses the same shared object repository and has one or more occurrences of the modified object, the names within that test or component are updated. This may take a few moments.

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

For more information on shared and per-action object repository modes, see Chapter 34, “Choosing the Object Repository Mode.”

To modify a test object's name:



- 1** Open the object repository. Choose **Tools > Object Repository**, click the **Object Repository** toolbar button, or open the Object Properties dialog box for the object you want to modify and click the **Repository** button.
- 2** Right-click the object in the object repository tree and choose **Rename**.
- 3** Modify the name of the object and click **OK** or select another object in the object repository tree.

The name you assign to the object must be unique within the object repository. Object names are not case-sensitive.

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. For recorded steps, QuickTest uses the properties in the object repository to create this temporary version of the object. For the remainder of the test or component, QuickTest refers to the temporary version of the test object rather than to the test object in the object repository.

You can also create temporary versions of test objects to represent objects from your Web site or application using programmatic descriptions and without using the object repository.

Creating Test Objects During a Run Session

Programmatic descriptions enable you to create temporary versions of test objects to 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 could use a programmatic description to add a statement in the Expert View that enters a value in the new edit box, so that QuickTest can identify the object even though you never recorded on the object or added it to the object repository. For more information on programmatic descriptions, see “Using Programmatic Descriptions” on page 878.

Modifying Test Object 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 Expert View.

Use the following syntax for the **SetTOProperty** method:

Object(description).SetTOProperty Property, Value

Note: You can also add a **SetTOProperty** statement from the Keyword View using the Step Generator. For more information, see “Inserting Steps Using the Step Generator” on page 467.

For information, refer to the *QuickTest Professional Object Model Reference*.

Modifying Object Descriptions

You can change the properties that QuickTest uses to identify an object. This is useful when you want to create and run tests or components 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, e.g. from a database.

You can also change the properties that identify an object in order to reference objects using properties that were not automatically learned while recording.

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 others 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 or component 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 so that you 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.

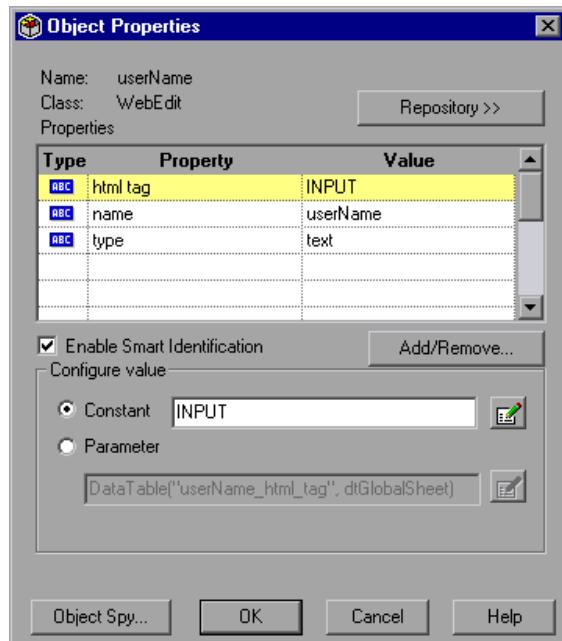
The Object Properties dialog box contains the set of object properties and values that make up the test object description of a selected object. QuickTest uses this description to identify the object during a run session. For each object class, QuickTest has a default property set that it uses for the object description for a particular object. You can use the Add/Remove Properties dialog box to change the properties that are included in the object description.

Note: You can also modify the set of properties that QuickTest learns when it records objects from a particular object class using the Object Identification dialog box. Such a change generally affects only objects that you record after you make the change. For more information, see “Configuring Object Identification” on page 789. You can also apply the changes you make in the Object Identification dialog box to the descriptions of all objects in an existing test or component using the **Update Run** option. For more information, see “Updating a Test or Component” on page 519.

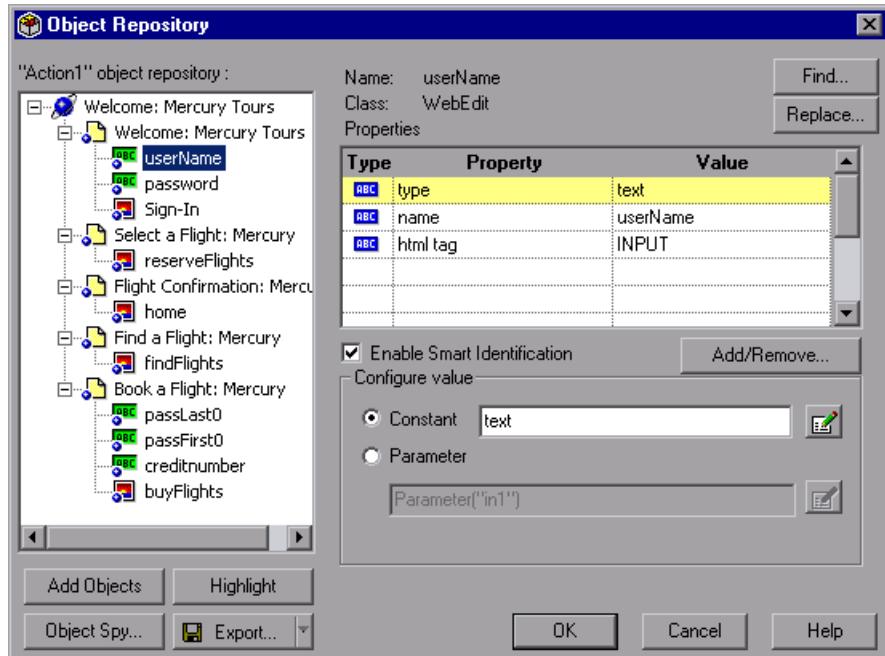
To modify an object description:

- 1 Right-click the object in the Keyword View or Active Screen containing the object that changed, and choose **Object Properties** (or **View/Add Object** from the Active Screen) or choose **Step > Object Properties**.

The Object Properties dialog box opens and displays the properties that are included in the object description.



If you want to view all objects in the action, test, or component, click the **Repository** button.  If you are working in object repository per-action mode, the Object Repository dialog box opens and displays the tree of the action. If you are working in shared object repository mode, the Object Repository dialog box displays the tree of all objects stored in the object repository file.



 **Tip:** You can also open the object repository by choosing **Tools > Object Repository** or click the **Object Repository** toolbar button.

2 Click the **Add/Remove** button.

The Add/Remove Properties dialog box opens, listing the properties that can be used to identify the object. A selected check box next to a property indicates that the property is part of the object description. The Type column indicates whether the property is a constant or a parameter. The value for each property is displayed in the **Value** column.



Notes: Values for all properties are displayed only if the application you are recording is currently open. If the application is closed, only values for properties that were part of the object description at the time of recording are shown.

You can click the **Add** button to add valid test object properties to this properties list. For more information, see “Adding Test Object Properties to the Properties List,” below.

3 Select the properties you want to include in the object description:

- To add a property to the object description, select the corresponding check box.
- To remove a property from the object description, clear the corresponding check box.

Note: You can click the **Default** button to instruct QuickTest to select only the default properties for the object class as defined in the Object Identification dialog box. For more information on the Object Identification dialog box, see Chapter 33, “Configuring Object Identification.”

4 Click **OK** to close the Add/Remove Properties dialog box.

5 Click **OK** to save your changes and close the Object Repository or Object Properties dialog box.

Tip: After you add a new property to the object description, you can modify its value in the **Configure Value** area in the Object Repository or Object Properties dialog box. For more information on modifying object properties, see “Modifying Test Object Properties While Editing Your Test or Component” on page 60.

Adding Test Object Properties to the Properties List

You can add a valid test object property to the object properties list in the Add/Remove dialog box. Suppose you want QuickTest to use a specific property to identify your object, but that property is not listed in the properties list. You can open the Add Property dialog box and add that property to the list.

Note: You can use the Properties tab of the Object Spy to view a complete list of valid test object properties for a selected object. For more information on the Object Spy, see “Viewing Object Properties Using the Object Spy” on page 42.

To add test object properties to the properties list:

- 1 Open the Add/Remove Properties dialog box. For more information, see “Modifying Object Descriptions” on page 67.
- 2 Click the **Add** button. The Add Property dialog box opens.



- 3 Enter a valid test object property:
 - **Property name**—Enter the property name.
 - **Property value**—Enter the value for the property.

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

- 4 Click **OK** to add the property to the list and close the Add Property window.
- 5 Click **OK** to include the new property in the object description and close the Add/Remove Properties dialog box.

Adding Objects to the Object Repository

When you record a test or component, QuickTest adds each object on which you perform an operation to the object repository. You can also add objects to the object repository while editing your test or component.

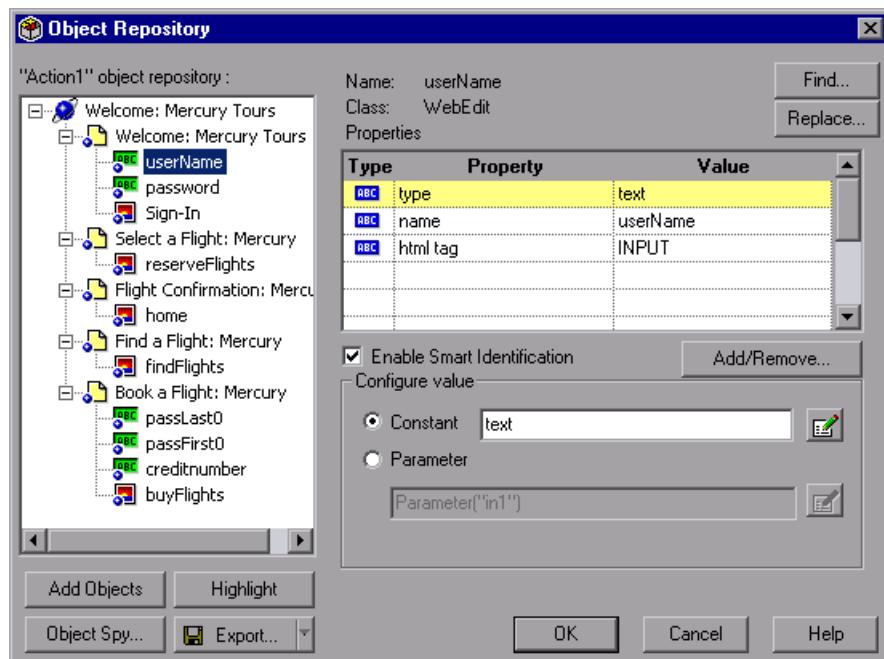
There are various ways to add an object to the object repository while editing a test or component:

- Use the **Add Objects** option in the Object Repository dialog box. You can add any object as a single object or a parent object, along with all its children.
- Choose the **View/Add Object** option from the Active Screen.
- Insert a step in your test or component for a selected object from the Active Screen.

Note: To add 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 tab of the Options dialog box. For more information, see “Setting Active Screen Options” on page 618.

To add objects to the object repository using the Add Objects option in the Object Repository dialog box:

- 1 Choose **Tools > Object Repository** or click the **Object Repository** toolbar button. The Object Repository dialog box opens.

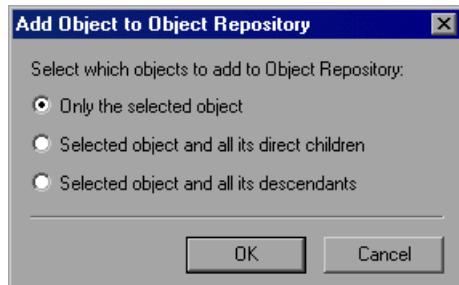


- 2** Click **Add Objects**. QuickTest and the Object Repository dialog box are minimized and the arrow becomes a pointing hand.
-

Note: If the window containing the object you want to add is partially hidden by another window, hold the pointing hand over the partially hidden window for a few seconds. The window comes into the foreground. You can now point to and click the object you want. You can configure the length of time required to bring a window into the foreground in the General tab of the Options dialog box. For more information, see Chapter 24, “Setting Global Testing Options.” You can also hold the CTRL key temporarily deactivate the pointing hand mechanism while you change the window focus. Note that pressing the CTRL key does not enable you to select an application from the Windows taskbar, therefore you must make sure that the window you want to access is not minimized.

- 3** If the object you want to add to your repository can be displayed only by performing an event (such as a right-click or a mouse-over to display a context menu), hold the CTRL key. The pointing hand temporarily turns into a standard arrow and you can perform the event. When the object you want to add is displayed, release the CTRL key. The arrow becomes a pointing hand again.
- 4** Click the object you want to add to your object repository.
- 5** 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 your repository and click **OK** to close the Object Selection dialog box.

- 6 If the object you select in the Object Selection dialog box 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 Add Object to Object Repository dialog box opens.

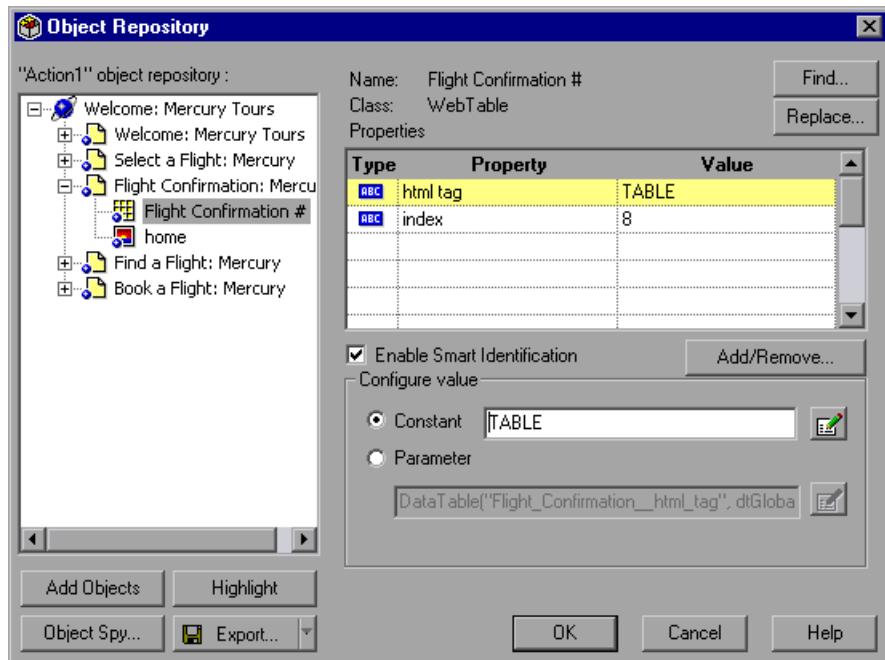


You can choose from the following options:

- **Only the selected object**—adds to the object repository the selected object's properties and values, without its child objects
- **Selected object and all its direct children**—adds to the object repository the properties and values of the selected object, along with those child objects one level below the selected parent object
- **Selected object and all its descendants**—adds to the object repository the properties and values of the selected object, along with all the child objects contained in the selected object, as well as all descendants of the object's child objects

Make your selection and click **OK** to close the Add Object to Object Repository dialog box.

- 7 The Object Repository dialog box opens, displaying the new object and its properties and values. QuickTest also adds the new object's parent objects if they do not already exist in the object repository.



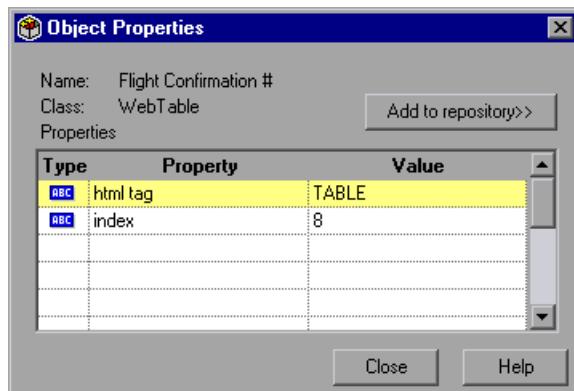
You can edit your new object's settings in the Object Repository dialog box just as you would any other object in your repository.

- 8 Click **OK**. The Object Repository dialog box closes and the new object(s) is stored in the object repository.

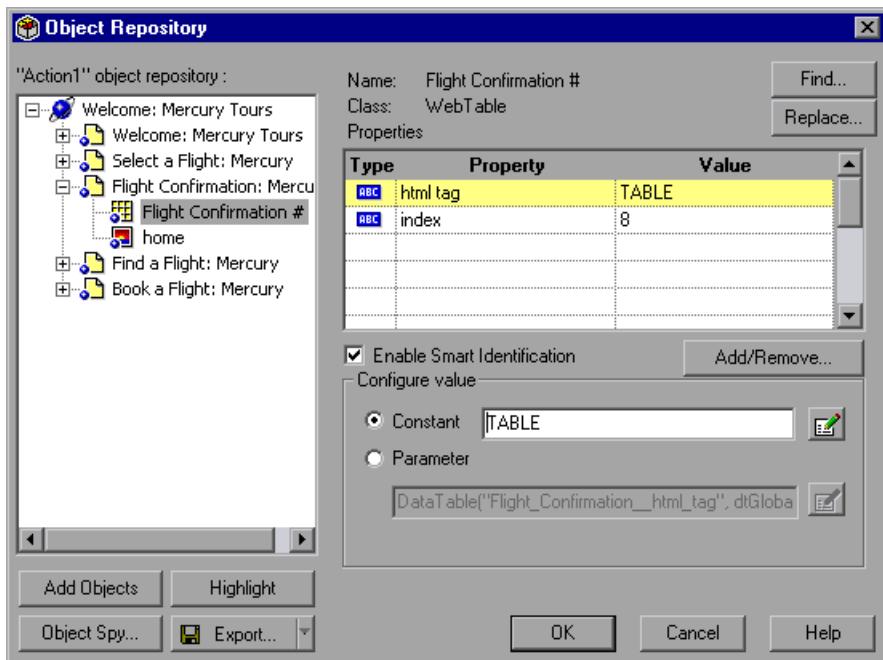
To add an object to the object repository using the View/Add Object option from the Active Screen:



- 1 If the Active Screen is not displayed, choose **View > Active Screen** or click the Active Screen toolbar button to display it.
- 2 Select a step in your test or component whose Active Screen contains the object that you want to add to the object repository.
- 3 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 for which you want to add a step, and click **OK** to close the Object Selection dialog box.
- 5 The Object Properties dialog box opens and displays the default test object properties for the object.



- 6 Click **Add to repository**. The Object Repository dialog box opens and displays the object properties for the selected object.



You can edit your new object's settings in the Object Repository dialog box just as you would any other object in your test or component.

- 7 Click **OK**. The Object Repository dialog box closes and the object is stored in the object repository.

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



- 1** If the Active Screen is not displayed, choose **View > Active Screen** or click the Active Screen toolbar button to display it.
- 2** Select a step in your test or component whose Active Screen contains the object for which you want to add a step.
- 3** 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, method, 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 or component, and the object is added to the object repository (if it was not yet included).

Deleting an Object from the Object Repository

When you remove a step from your test or component, the object remains in the object repository.

If you are working with per-action object repositories 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 test or component using the same shared object repository file before you choose to delete the object from the object repository.

Note: If your action or component contains references to an object that you have deleted from the object repository, your test or component will not run.

To delete an object from the object repository:

-  **1** Right-click an object you want to delete in the Keyword View or Active Screen and choose **Object Properties** (or **View/Add Object** in the Active Screen), and then click the **Repository** button, choose **Tools > Object Repository**, or click the **Object Repository** toolbar button. The Object Repository dialog box opens.
- 2** In the repository tree, right-click the object you want to delete and select **Delete**. A confirmation message is displayed.
- 3** Click **Yes** to confirm that you want to delete the object.
- 4** Click **OK** to delete the object from the object repository and close the Object Repository dialog box.

Tip: You can click **Cancel** in the Object Repository dialog box to cancel any deletions you performed.

Part III

Creating Tests or Components

5

Designing Tests

You can quickly create a test by recording the operations you perform on your Web site or application. Once you have created your test, you can enhance your test using checkpoints and other special testing options.

This chapter describes:

- ▶ About Designing Tests
- ▶ Planning a Test
- ▶ Recording a Test or Component
- ▶ Understanding Your Recorded Test or Component
- ▶ Enhancing Your Test
- ▶ Managing Your Test
- ▶ Choosing the Recording Mode
- ▶ Changing the Active Screen
- ▶ Creating, Opening, and Saving Tests with Locked Resources

About Designing Tests

QuickTest enables you to generate an automated test by recording the typical processes that you perform on your Web site or 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 site or application, for example, clicking a link or typing data into an edit box.

While recording or designing your test, you can insert checkpoints into your test. A *checkpoint* compares the value of an element captured in your test when you recorded your test, with the value of the same element captured during the test run. This helps you determine whether or not your application or Web site is functioning correctly.

When you test your application or site, 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 by defining environment variables and values, or you can have QuickTest generate random numbers or current user and test data. For more information, see Chapter 12, “Parameterizing Values.”

After recording, you can further enhance your test by adding and modifying steps in the Keyword View.

Notes:

Many QuickTest recording and editing operations are performed using the mouse. In accordance with the 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 operations using QuickTest’s shortcut keys. For a list of shortcut keys, see “Executing Commands Using Shortcut Keys” on page 22.

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 and character sets.

Planning a Test

Before you start recording, you should plan your test and consider the following suggestions and options:

- Determine the functionality you want to test. Short tests that check specific functions of the application or site or complete a transaction are better than long tests that perform several tasks.
- Decide which information you want to check during the test. A checkpoint can check for differences in the text strings, objects, and tables in your application or site. For more information, see Chapter 6, “Understanding Checkpoints.”
- 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 Chapter 35, “Configuring Web Event Recording.”
- 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 Chapter 12, “Parameterizing Values.”
- Change the way that QuickTest identifies objects. This is particularly helpful when your application contains objects that change frequently or are created using dynamic content, e.g. from a database. For additional information, see Chapter 33, “Configuring Object Identification.”
- Decide how you want to organize your object repository files. You can work with the individual action’s object repositories, or you can use a common (shared) object repository file for multiple tests. If you are new to testing, you may want to keep the default object repository per-action setting. Once you feel more comfortable with the basics of test design, you may want to take advantage of the shared object repository option. For additional information, see Chapter 34, “Choosing the Object Repository Mode.”

- Consider using actions to streamline the testing process. For additional information, see Chapter 17, “Working with Actions.”
- Link to WinRunner tests and call WinRunner TSL functions from a QuickTest test. For additional information, see Chapter 39, “Working with WinRunner.”

Recording a Test or Component

You create a test or component 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.

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 17, “Working with Actions.” Components do not use actions. Each component is a single self-contained unit, and cannot include calls to other components.

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 about low-level and analog recording, see “Choosing the Recording Mode” on page 101.

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 16, “Learning Virtual Objects.”

Consider the following guidelines when recording a test or component:

- 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 or Web browser 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 or browser that is already open. For more information, see Chapter 26, “Setting Record and Run Options.”
- Choose how you want QuickTest to record and run your test or component by setting global testing options in the Options dialog box and settings specific to your test or component in the Test Settings or Business Component Settings dialog boxes. For more information, see Chapter 24, “Setting Global Testing Options” and Chapter 25, “Setting Options for Individual Tests or Components.”
- If you are recording on a drop-down list in a Web page or application, you must click on the drop-down 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). This is because QuickTest only records a step if the value in the list changes.

Note: If you are creating a test on Web objects, you can record your test on one browser and run it on another browser. QuickTest supports the following browsers—Netscape Navigator, Microsoft Internet Explorer, AOL, and applications with embedded Web browser controls. For additional information, see Chapter 30, “Testing Web Objects.”

Following are instructions for recording a test. Instructions for recording a component are located in “Recording Components” on page 991.

To record a test:



1 Open QuickTest. For more information, see “Starting QuickTest” on page 12.

2 Open a test:



➤ To create a new test, click the **New** button or choose **File > New**.

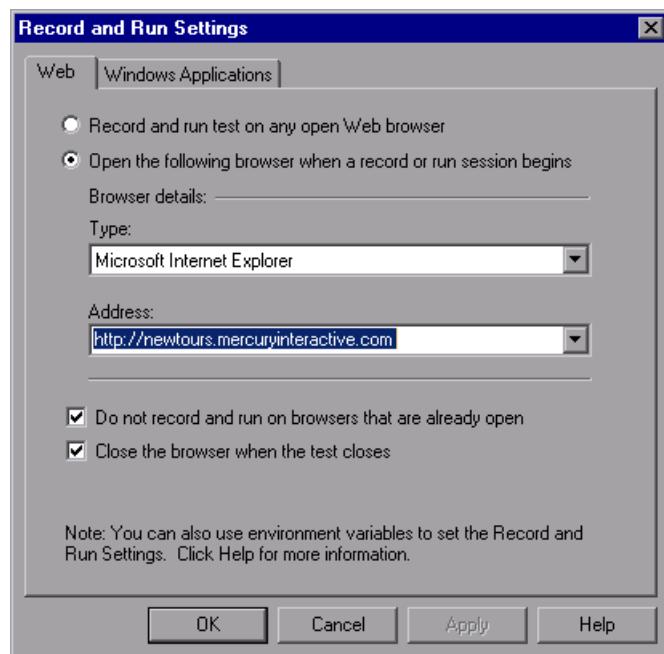


➤ To open an existing test, click the **Open** button or choose **File > Open**. In the Open QuickTest Test dialog box, select a test and click **Open**.

For more information, see “Managing Your Test” on page 95.



3 Click the **Record** button or choose **Test > 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 **Test > Record and Run Settings**), the Record and Run Settings dialog box opens.



Once you have 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 choose **Test > 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 or browser than the one you previously used.

The Record and Run Settings dialog box contains tabbed pages corresponding to the add-ins loaded. These can include both built-in and separately purchased add-ins, where applicable. The image here includes the environments listed below:

Tab Heading	Subject
Windows Applications	Options for testing standard Windows applications.
Web	Options for testing Web sites. Note: The Web tab is available only when Web support is installed and loaded.

- 4 Set an option, as described in Chapter 26, “Setting Record and Run Options.”
- 5 To apply your changes and keep the Record and Run Settings dialog box open, click **Apply**.

For more information about record and run settings, see “Setting Web Record and Run Options” on page 703.

- 6 Click **OK** to close the Record and Run Settings dialog box and begin recording your test.
- 7 Navigate through your application or Web site. QuickTest records each step you perform and displays it in the Keyword View and Expert View.
- 8 To determine whether or not your Web site or application is functioning correctly, you can insert text checkpoints, object checkpoints, and bitmap checkpoints. For more information, see Chapter 6, “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 12, “Parameterizing Values.”
-  **10** When you complete your recording session, click the **Stop** button or choose **Test > Stop**.
-  **11** To save your test, click the **Save** button or choose **File > Save**. In the Save QuickTest Test dialog box, assign a name to the test. QuickTest suggests a default folder called **Tests** under your QuickTest Professional installation folder. For more information, see “Managing Your Test” on page 95.

Understanding Your Recorded Test or Component

While recording, 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 Mercury Interactive sample Web site.

Item	Operation	Value	Documentation
Action1			Call the Action1 action.
Welcome: Mercury Tours			
Welcome: Mercury Tours			
userN ^{ame}	Set	"mercury"	Enter "mercury" in the "userName" edit box.
password	SetSecure	"405d7bdbe...	Enter the encrypted string "405d7bdbe1b2..."
Sign-In	Click	2,2	Click the "Sign-In" image.

The table below provides an explanation of each step in the Keyword View.

Step	Description
Action1	Action1 is the action name.
Welcome: Mercury Tours	The browser invokes the Welcome: Mercury Tours Web site.
Welcome: Mercury Tours	Welcome: Mercury Tours is the name of the Web page.

Step	Description
<code>userN</code> Set "mercury"	userName is the name of the edit box. Set is the method performed on the edit box. mercury is the value of the edit box.
<code>password</code> SetSecure "405d7bdbe...	password is the name of the edit box. SetSecure is an encryption method performed on the edit box. 405d7bdbe1b2c594dc9822f39212f163 is the encrypted value of the password.
<code>Sign-In</code> Click 2,2	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.

QuickTest also generates a test script—a VBScript program based on the QuickTest object model. The test script is displayed in the Expert View as follows:

```
Browser("Welcome: Mercury Tours").Page("Welcome: Mercury Tours").
    WebEdit("userName").Set "mercury"
Browser("Welcome: Mercury Tours").Page("Welcome: Mercury Tours").
    WebEdit("password").SetSecure "405d7bdbe1b2c594dc9822f39212f163"
Browser("Welcome: Mercury Tours").Page("Welcome: Mercury Tours").
    Image("Sign-In").Click 2,2
```

Enhancing Your Test

After you have recorded a test, you can use a variety of options to enhance your test.

You can add checkpoints to your test. A *checkpoint* is a step in your test that compares the values of the specified property during a test run with the values stored for the same test object property within the test. This enables you to identify whether or not your Web site or application is functioning correctly. For more information about creating checkpoints, see Chapter 6, “Understanding Checkpoints.”

You can parameterize your test to replace fixed values with values from an external source during your test run. The values can come from a Data Table, environment variables you define, or values that QuickTest generates during the test run. For more information, see Chapter 12, “Parameterizing 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 13, “Outputting Values.”

You can divide your test into actions to streamline the testing process of your Web site or application. For more information, see Chapter 17, “Working with Actions.”

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 methods 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 20, “Adding Steps Containing Programming Logic”.

You can also manually enter standard VBScript statements, as well as statements using QuickTest test objects and methods, in the Expert View. For more information, see Chapter 36, “Working with the Expert View.”

Managing Your Test

You can use the File toolbar to create, open, save, zip, unzip, and print recorded tests.

Creating a New Test



To create a new test, click the **New** button or choose **File > New**. A new test opens. You are ready to start recording your test.

Note: If your default test settings include associating new tests with a locked resource file, a message regarding locked resources opens when you create a new test. For more information, see “Creating, Opening, and Saving Tests with Locked Resources” on page 109.

Opening an Existing Test

You can open an existing test to enhance or run it.

To open an existing test:



- 1 Click the **Open** button or choose **File > Open**. The Open QuickTest Test dialog box opens.
- 2 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.

You can also open tests that are part of a Quality Center project. For more information, see Chapter 40, “Working with Quality Center”.

Tip: You can open a recently used test by selecting it from the Recent Files list in the **File** menu.

Note: If the test you are opening is associated with a locked resource file, a message opens instructing you how to open the test. For more information, see “Creating, Opening, and Saving Tests with Locked Resources” on page 109.

Saving a Test

You can save a new test or save changes to an existing test. When you save a test, any modified resource files associated with the test are also saved. These associated resources include the Data Table file and the shared object repository file.

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 Click the **Save** button or choose **File > Save** to save the test. The Save QuickTest Test dialog box opens.
- 2 Choose the folder in which you want to save the test. QuickTest suggests a default folder called **Tests** under your QuickTest Professional installation folder.
- 3 Type a name for the test in the **File** name box.
- 4 Confirm that **Save Active Screen files** is selected if you want to save the Active Screen files with your test.

Note that 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 or Component” on page 519.

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 tab of the Options dialog box. For more information, see Chapter 24, “Setting Global Testing Options.”

- 5 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.
- Choose **File > Save As** to save an existing test to a new name or a new location. If you choose **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 4 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.

You can also save tests to a Quality Center project. For more information, see Chapter 40, “Working with Quality Center”.

Note: If you are saving a test whose locked resources were not opened in read-write mode and modified while editing the test, a warning message opens regarding saving the test. For more information, see “Creating, Opening, and Saving Tests with Locked Resources” on page 109.

Zipping a Test

While you record, QuickTest creates a series of configuration, run-time, setup data, and 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** Choose **File > Export to Zip File**. The Export to Zip File dialog box opens.
- 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

To open a zipped test in QuickTest, you must use the **Import from Zip File** command to unzip it.

To unzip a zipped test:

- 1** Select **File > Import from Zip File**. The Import from Zip File dialog box opens.
- 2** Type or select 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 or Component

You can print your entire test from the Keyword View (in table format). You can also print a single action or component 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:



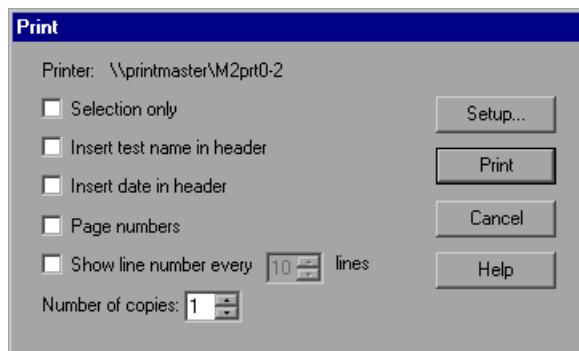
- 1** Click the **Print** button or choose **File > Print**. A standard Print dialog box opens.
- 2** Click **OK** to print the content of the Keyword View to your default Windows printer.

Tip: You can choose **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 choose **File > Print**. The Print dialog box opens.



- 2** Specify the print options that you want to use:

- **Printer**—Displays the printer to 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 test name in header**—Includes the name of the test 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.

Choosing the Recording Mode

QuickTest's 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 QuickTest's 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 tests 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 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 QuickTest's 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. Once you have recorded the necessary steps in **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 test run, 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 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 object—**Click**, **DblClick**, **Drag**, **Drop**, **Type**
 - Window test object—**Click**, **DblClick**, **Drag**, **Drop**, **Type**, **Activate**, **Minimize**, **Restore**, **Maximize**
- Each step recorded in **Low-Level Recording** mode is shown in the Keyword View and test script. (**Analog Recording** records only the one step in the Keyword View that calls the external analog data file.)

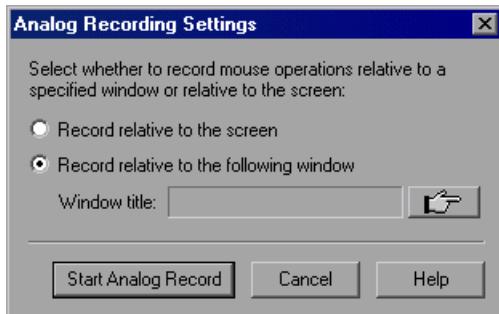
Using Analog Recording

You can switch to **Analog Recording** mode only while recording a test. The option is not available while editing a test.

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 choose **Test > 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.

Notes: When you record in analog mode relative to the screen, the test run 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 mode relative to 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 mode. The title of the window you clicked is displayed in the window title box.

Tip: You can also hold the CTRL key to change the window focus or perform operations such as a right-click or mouseover to display a context menu in order select the window you require. Note that pressing the CTRL key does not enable you to select an application from the Windows taskbar, therefore you must make sure that the window you want to access is not minimized.

- 5 Click **Start Analog Record**.

- 6 Perform the operations you want to record in analog 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 choose **Test > 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 17, “Working with Actions.”

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 test run, click **CTRL + ESC**, then click **Stop** in the Testing toolbar.

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 choose **Test > 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 choose **Test > 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 **mercury** into a user name edit box and then click the TAB key while in normal recording mode. Your test is displayed as follows in the Keyword View and Expert View:

3		Welcome: Mercury Tours		
		Welcome: Mercury Tours		
		userName	Set	"mercury"

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

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	"mercury"	Type "mercury" 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 "mercury"  
Window("Microsoft Internet Explorer").WinObject("Internet Explorer_Server").  
    Type micTab
```

Changing the Active Screen

As the content of your application or Web site changes, you can continue to use tests you developed previously. You simply 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 the Mercury Tours site now includes a new object and you want to add a checkpoint that checks for this object. You can use the **Change Active Screen** command to replace the page in your Active Screen tab and then proceed to create a checkpoint for this object.

To change the Active Screen:

- 1 Make sure that your application or Web browser is displaying the window or page that you want to use to replace what is currently displayed in the Active Screen tab.
- 2 In the Keyword View, click a step that you want to change and the window or page is displayed in the Active Screen tab.
- 3 Choose **Tools > Change Active Screen**. The QuickTest window is minimized and the mouse pointer becomes a pointing hand.

- 4 Click the window or page displayed in your application or browser. A message prompts you to change your current Active Screen display.
-

Tip: You can also hold the CTRL key to change the window focus or perform operations such as a right-click or mouseover to display a context menu to choose the new Active Screen display. Note that pressing the CTRL key does not enable you to select an application from the Windows taskbar, therefore you must make sure that the window you want to access is not minimized.

- 5 Click Yes.

Creating, Opening, and Saving Tests with Locked Resources

QuickTest resource files can be locked by QuickTest to protect the information in the file from being overwritten. QuickTest resource files are locked if the file:

- is being used by another QuickTest user
- has been checked into a Quality Center version control database or some other version control software
- has been marked as read-only

External Resource Files

QuickTest resource files are external files that are associated with your test and can be edited within QuickTest. These can include:

- Shared object repository files. (For more information, see Chapter 34, “Choosing the Object Repository Mode.”)
 - Data Table files. (For more information, see Chapter 18, “Working with Data Tables.”)
-

Note: External files that are associated with your test, but cannot be edited within QuickTest (such as library files and environment variable files) are not affected by this locking mechanism.

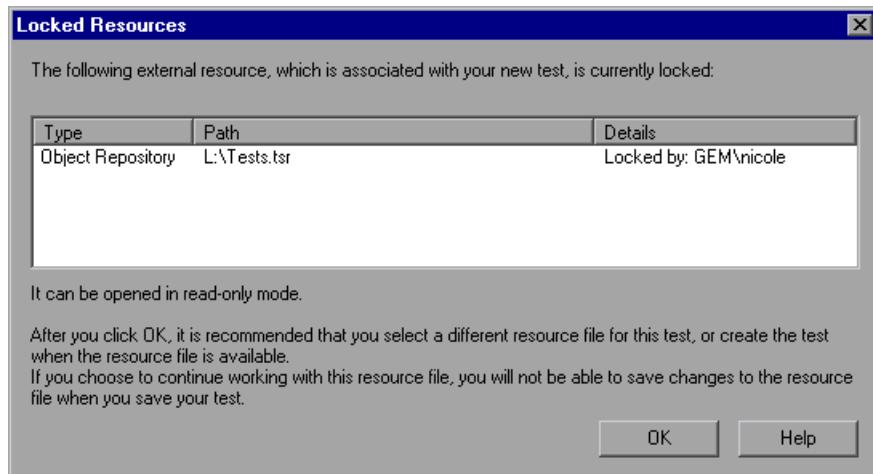
QuickTest notifies you if a QuickTest resource file that is associated with your test is locked when:

- creating a new test whose default settings associate the test with a QuickTest resource that is locked. For more information, see “Creating Tests with Locked Resources” on page 111.
- opening a test whose QuickTest resource file is locked. For more information, see “Opening Tests with Locked Resources” on page 112.
- saving a test whose external resource files were not opened in read-write mode (because they were locked when you opened the test) and modified while editing the test. For more information, see “Saving Tests with Locked Resources” on page 114.

Creating Tests with Locked Resources

If your QuickTest settings have been configured to use a default, shared object repository file, any test you create is automatically associated with that external file for all test object information. When you create a new test, the shared object repository file may be locked.

When you open QuickTest and when you create a new test, the following message opens if the shared object repository file is locked:



Click **OK**.

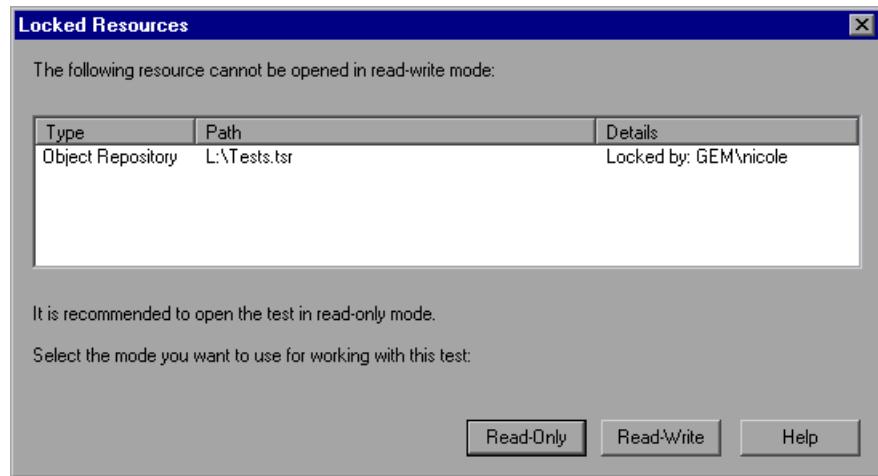
It is recommended that you select a different shared object repository or select **Per-action** as the **Object repository type** in the Resources tab of the Test Settings dialog box before beginning to record or edit your test. For more information, see “Setting the Object Repository Mode” on page 828. Alternatively, create the test when the associated shared object repository is available.

Note: You are able to change the object repository of the test only before you begin recording or editing the test.

If you choose to create the test with a locked shared object repository, then the shared object repository opens in read-only mode. You can run the test while it is open and you can choose to save the test, but any changes to the shared object repository are not saved. This means that if any of the descriptions or values of any of the objects change as a result of recording or editing your new test, the object repository file will not be saved with the modified data.

Opening Tests with Locked Resources

When you choose to open a test that is associated with a locked, QuickTest resource file, the following message opens:



The resources listed are opened in read-only mode. You have the option to open the test in:

- **Read-Only Mode**
- **Read-Write Mode**

Read-Only Mode

If you open the test in read-only mode, you can run the test but not make any changes to it. All editing options are disabled and you cannot edit the test in the Keyword View or the Expert View. While the test itself cannot be saved, if you run the test and choose to save the results, the test results file is saved.

You can choose the **Save As** option from the **File** menu, save the test under a new name, and then edit the test. The same is true for a shared object repository file opened in read-only mode.

Read-Write Mode

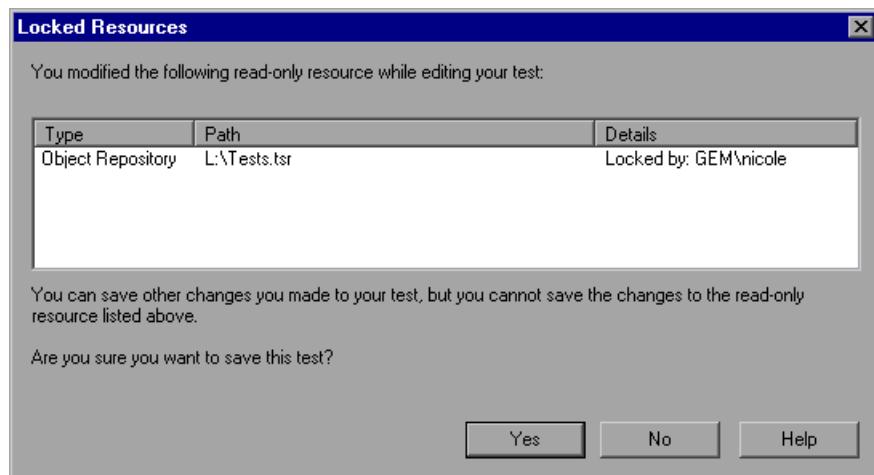
If you open a test in read-write mode, you can run and edit the test and save the changes you make to the test. However, any changes to locked, QuickTest resource files cannot be saved. Locked resource files are opened in read-only mode even if the test has been opened in read-write mode.

For example, suppose you choose to open a test in read-write mode that has a locked Data Table file associated with it. When you edit the test, any changes to the Data Table will not be saved. If any of the Data Table values have changed as a result of editing the test and you save the changes to the test, the test may fail the next time it calls the unmodified Data Table values.

It is therefore recommended to open the test in read-only mode or to close the test without saving.

Saving Tests with Locked Resources

When you choose to save a test with locked resources that you opened in read-write mode, the following message opens if anything in a locked resource file has changed:



It is recommended not to save a test whose locked resource files have been modified. If you do save the test, the next time you run the test, the test may fail because the test may be expecting different values from the resource files.

6

Understanding Checkpoints

You can check objects in your application or Web site to ensure that they function properly.

This chapter describes:

- About Understanding Checkpoints
- Adding Checkpoints to a Test or Component
- Understanding Types of Checkpoints

About Understanding Checkpoints

QuickTest enables you to add checks to your test or 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 Web site or application is functioning correctly.

When you add a checkpoint, QuickTest adds a checkpoint to the current row in the Keyword View and adds a **Check CheckPoint** statement in the Expert View. When you run the test or component, 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.

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 873.

Adding Checkpoints to a Test or Component

You can add checkpoints during a recording session or while editing your test or component. It is generally more convenient to define checks once the initial test or component has been recorded.

There are several ways to add checkpoints.

To add checkpoints while recording or editing:

- Use the commands on the **Insert** menu, or click the arrow beside the **Insert Checkpoint** button on the Testing toolbar. This displays a menu of checkpoint options that are relevant to the selected step in the Keyword View.


To add a checkpoint while editing only:

- Right-click the step in the Keyword View where you want to add the checkpoint and choose **Insert Standard Checkpoint**.
 - Right-click any object in the Active Screen and choose **Insert Standard Checkpoint**. This option 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 method, ensure that Active Screen contains sufficient data for the object you want to check. For more information, see “Setting Active Screen Options” on page 618.

Throughout this User’s 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.

Understanding Types of Checkpoints

You can insert the following checkpoint types to check various objects in a Web site or application.

- **Standard Checkpoint** checks the property value of an object in your application or Web page. The standard checkpoint checks a variety of objects such as buttons, radio buttons, combo boxes, lists, etc. For example, you can check that a radio button is activated after it is selected or you can check the value of an edit field.

Standard checkpoints are supported for all add-in environments (see “Supported Checkpoints” on page 121).

For more information on standard checkpoints, see Chapter 7, “Checking Object Property Values.”

- **Image Checkpoint** checks the value of an image in your application or Web page. 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 environment (see “Supported Checkpoints” on page 121).

For more information on image checkpoints, see Chapter 7, “Checking Object Property Values.”

- **Bitmap Checkpoint** checks an area of your Web page or 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.

Bitmap checkpoints are supported for all add-in environments (see “Supported Checkpoints” on page 121).

For more information on bitmap checkpoints, see Chapter 10, “Checking Bitmaps.”

- **Table Checkpoint** checks information within a table. For example, suppose your application or Web site 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 and ActiveX environments (see “Supported Checkpoints” on page 121). Table checkpoints are also supported for many external add-in environments.

For more information on table checkpoints, see Chapter 8, “Checking Tables and Databases.”

- **Text Checkpoint** checks that a text string is displayed in the appropriate place in your application or on a Web page. For example, suppose your application or 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 all add-in environments (see “Supported Checkpoints,” below).

For more information on text checkpoints, see Chapter 9, “Checking Text.”

- **Text Area Checkpoint** checks that a text string is displayed within a defined area in a Windows 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 Standard Windows, Visual Basic, and ActiveX add-in environments (see “Supported Checkpoints” on page 121).

Text area checkpoints are also supported for some external add-in environments.

For more information on text area checkpoints, see Chapter 9, “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 environment (see “Supported Checkpoints” on page 121).

For more information on accessibility checkpoints, see Chapter 30, “Testing Web Objects.”

- **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 environment (see “Supported Checkpoints” on page 121).

For more information on page checkpoints, see Chapter 30, “Testing Web Objects.”

- **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 by all environments (see “Supported Checkpoints,” below).

For more information on database checkpoints, see Chapter 8, “Checking Tables and 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 11, “Checking XML.”

XML checkpoints (Web page/frame) are supported for the Web environment; XML checkpoints (file) are supported by all environments (see “Supported Checkpoints,” below).

Supported Checkpoints

The following table shows the types of checkpoints that are supported for each add-in environment supplied with the core installation of QuickTest Professional.

	Web	Std	VB	ActiveX	Other Object
Standard	S	S	S	S	NA
Image	S	NS	NS	NS	NA
Table	S	NS	NS	S	NA
Text	S	S	S	S	NA
Text Area	NS	S	S	S	NA
Bitmap	S	S	S	S	NA
Accessibility	S	NS	NS	NS	NA
XML	S	NS	NS	NS	S (File)
Page	S	NA	NA	NA	NA
Database	NA	NA	NA	NA	S (DbTable)

S—Supported

NS—Not Supported

NA—Not Applicable

Note: Checkpoints are also supported for various external add-in environments. Refer to your QuickTest Professional add-in documentation for details.

Checking Object Property Values

By adding standard checkpoints to your tests and components, you can compare object property values in different versions of your application or Web site.

This chapter describes:

- ▶ About Checking Object Properties
- ▶ Creating Standard Checkpoints
- ▶ Understanding the Checkpoint Properties Dialog Box
- ▶ Understanding the Image Checkpoint Properties Dialog Box
- ▶ Modifying Checkpoints

About Checking Object Properties

You can check the object property values in your Web site or application using standard checkpoints. Standard checkpoints compare the expected values of object properties captured during recording to the object's current values during a run session.

You use standard checkpoints to perform checks on images, tables, Web page properties, and other objects within your application or Web site.

Note: You can create standard checkpoints for all supported testing environments (as long as the appropriate add-ins are loaded).

Creating Standard Checkpoints

You can check that a specified object in your application or on your Web page has the property values you expect, by adding a standard checkpoint to your test or component. To set the options for a standard checkpoint, you use the Checkpoint Properties dialog box.

Creating a Standard Checkpoint

You can add a standard checkpoint while recording or editing your test or component.

To add a standard checkpoint while recording:



- 1 Click the **Insert Checkpoint** toolbar button or choose **Insert > Checkpoint > Standard Checkpoint**.

The QuickTest window is minimized and the mouse pointer turns into a pointing hand.

Note: If the object you want to check can only be displayed by performing an event (such as a right-click or a mouse over to display a context menu), hold the CTRL key. You can also use the CTRL key to change the window focus. The pointing hand temporarily turns into a standard arrow and you can perform the event. When the object you want to check is displayed, release the CTRL key. The arrow becomes a pointing hand again.

- 2 Click the object you want to check. The Select an Object 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, for example:

Icon	Object	Class
	Windows button	WinButton
	Windows object	WinObject
	Windows edit box	WinEdit

Icon	Object	Class
	Windows dialog box	Dialog
	Web check box	WebCheckBox
	Web edit box	WebEdit
	Web radio button	WebRadioGroup
	Web list box	WebList
	Web element	WebElement
	Visual Basic combo box	VbComboBox
	Visual Basic radio button	VbRadioButton
	Visual Basic window	VbWindow

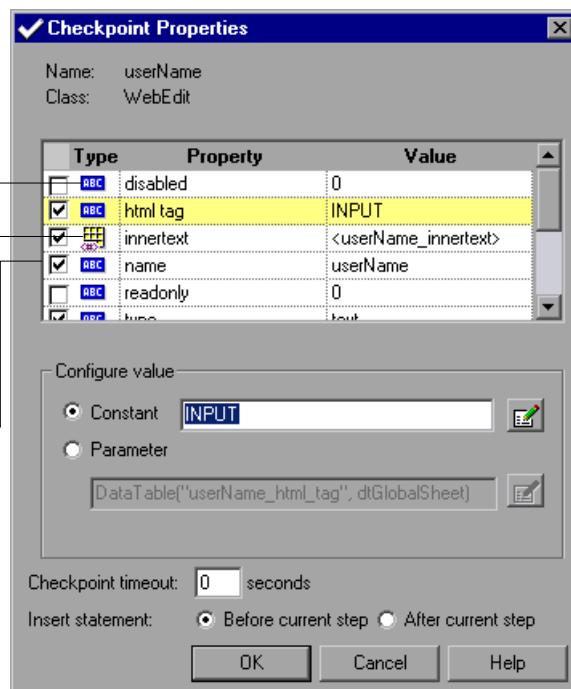
- 4 Click **OK**. The Checkpoint Properties dialog box opens.
- 5 Specify the settings for the checkpoint. For more information, see “Understanding the Checkpoint Properties Dialog Box” on page 126.
- 6 Click **OK** to close the dialog box. A checkpoint statement is added for the selected object in the Keyword View and Expert View.

To add a standard checkpoint while editing:

- 1 Right-click the step on which you want to perform a checkpoint and choose **Insert Standard Checkpoint**. The Checkpoint Properties dialog box opens.
- 2 Specify the settings for the checkpoint. For more information, see “Understanding the Checkpoint Properties Dialog Box,” below.
- 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 about the Dialog Box for:	See:
Image checkpoint properties	"Understanding the Image Checkpoint Properties Dialog Box" on page 130
Page checkpoint properties	"Understanding the Page Checkpoint Properties Dialog Box" on page 750
Table checkpoint properties	"Understanding the Table/Database Checkpoint Properties Dialog Box" on page 140

Identifying the Object

The top part of the dialog box displays information about the object to check:

Information	Description
Name	The name that QuickTest assigns to the test object.
Class	The type of object. In this example, the WebEdit class indicates that the object is an edit field.

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	<p>For each object class, QuickTest recommends default property checks. You can accept the default checks or modify them accordingly.</p> <p>To check a property, select the corresponding check box.</p> <p>To exclude a property check, clear the corresponding check box.</p>
Type	<p>The  icon indicates that the value of the property is currently a constant.</p> <p>The  icon indicates that the value of the property is currently a test, action, or component parameter.</p> <p>The  icon indicates that the value of the property is currently a Data Table parameter.</p> <p>The  icon indicates that the value of the property is currently an environment variable parameter.</p> <p>The  icon indicates that the value of the property is currently a random number parameter.</p>
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 286.

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 286.

Setting General Standard Checkpoint Options

The bottom part of the 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 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 make sure that the object has sufficient time to achieve that state and therefore enables the checkpoint to pass before the maximum timeout is reached.

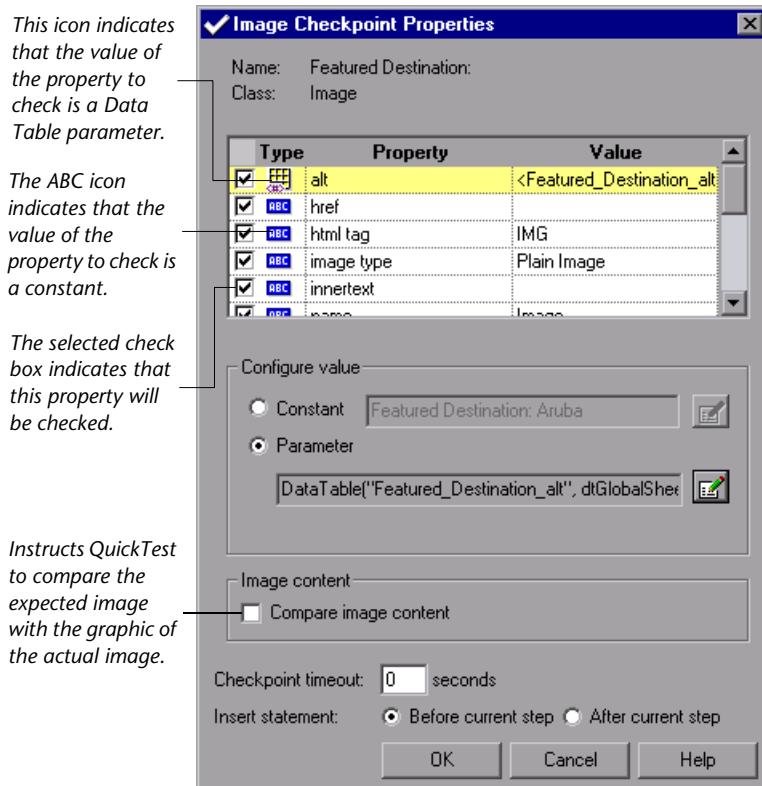
You can see information about the checkpoint timeout, including the time interval used by QuickTest to perform the checkpoint, in the Test Results window.

- **Insert statement**—Specifies when to perform the checkpoint in the test or component. Choose **Before current step** if you want to check the value of the object property before the highlighted step is performed. Choose **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 when adding a new checkpoint to an existing test or component 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 Image

The top part of the dialog box displays information about the image to check:

Information	Description
Name	The name that QuickTest assigns to the test object.
Class	The type of object. This is always Image .

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 128.

Editing the Expected Value of an Image Property

The middle part of the Image Checkpoint Properties dialog box contains the following:

- **Configure value** area—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 286.
- **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 General Standard Checkpoint Options” on page 129.

Modifying Checkpoints

You can modify the settings of existing checkpoints. For example, you can choose to use parameters, or you can use filters to specify which image sources and links to check.

To modify a checkpoint:

- 1** Right-click a row with a checkpoint in the Keyword View and choose **Checkpoint Properties**, or select a row with a checkpoint and choose **Step > Checkpoint Properties**. The relevant checkpoint dialog box opens.
- 2** Modify the properties and click **OK**.

8

Checking Tables and Databases

By adding table checkpoints, you can check the content of tables displayed in your application. By adding database checkpoints, you can check the contents of databases accessed by your Web site or application. The process for creating and modifying table or database checkpoints is quite similar.

This chapter describes:

- About Checking Tables and Databases
- Creating a Table Checkpoint
- Creating a Check on a Database
- Understanding the Table/Database Checkpoint Properties Dialog Box
- Modifying a Table Checkpoint
- Modifying a Database Checkpoint

About Checking Tables and Databases

You can check that a specified value is displayed in a cell in a table by adding a table checkpoint to your test or component. For ActiveX tables, you can also check the properties of the table object. To add a table checkpoint, you use the Checkpoint Properties dialog box.

Table checkpoints are supported for Web and ActiveX applications, as well as for a variety of external add-in environments.

You can use database checkpoints in your test or component to check databases accessed by your Web site or application and to detect defects. You define a query on your database, and then you create a database checkpoint that checks the results of the query.

Database checkpoints are supported for all core environments, as well as for a variety of external add-in environments.

There are two ways to define a database query:

- Use Microsoft Query. You can install Microsoft Query from the *custom installation* of Microsoft Office.
- Manually define an SQL statement.

Creating a Table Checkpoint

You can add a table checkpoint while recording or editing your test or component.

To add a table checkpoint while recording:



- 1 Choose **Insert > Checkpoint > Standard Checkpoint** or click the **Insert Checkpoint** button. The QuickTest window is minimized, and the mouse pointer turns into a pointing hand.

Tip: You can hold the CTRL key to change the pointing hand into a standard arrow, and then change the window focus or perform operations such as a right-click or mouseover to display a context menu. Note that pressing the CTRL key does not enable you to select an application from the Windows taskbar, therefore you must make sure that the window you want to access is not minimized.

- 2 Click a table you want to check. The Object Selection - Checkpoint Properties dialog box opens.
- 3 Select a table item from the displayed object tree.
- 4 Click **OK**. The Table Checkpoint Properties dialog box opens.
- 5 Specify the settings for the checkpoint. For more information, see “Understanding the Table/Database Checkpoint Properties Dialog Box” on page 140.

- 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 table checkpoint while editing:

- 1 Depending on whether the table object is already in a step in the Keyword View, perform one of the following:

- If you have already recorded a step on the table object you want to check, right-click the table step in the Keyword View and choose **Insert Standard Checkpoint**.
- If you have not recorded a step on the table object you want to check, make sure the **Active Screen** button is selected. Click a step in your test or component 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 choose **Insert Standard Checkpoint**. The Object Selection - Checkpoint Properties dialog box opens. Select a table item from the displayed object tree and click **OK**.



The Table Checkpoint Properties dialog box opens.

- 2 Specify the settings for the checkpoint. For more information, see “Understanding the Table/Database Checkpoint Properties Dialog Box” on page 140.
- 3 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 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 in order to check the contents of the entire result set, or a part of it. QuickTest captures the current data from the database and saves this information as *expected data*. A database checkpoint is inserted into the test or component. This checkpoint is displayed in the Expert View as a **DbTable.Check CheckPoint** statement and as a step in the Keyword View, as follows:

```
DbTable | Check | CheckPoint("DbTable") | Check whether specified content in a database query matches
```

When you run the test or component, 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. The results of the checkpoint can be viewed in the Test Results window. For more information, see Chapter 23, “Analyzing Test Results.”

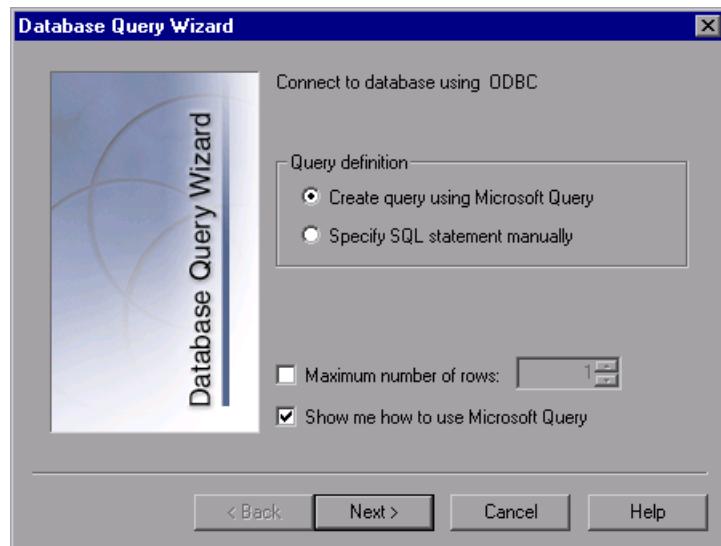
You can modify the expected data of a database checkpoint before you run your test or component. You can also make changes to the query in an existing database checkpoint. This can be useful if you move the database to a new location on the network.

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 Choose **Insert > Checkpoint > Database Checkpoint**. 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. Once 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. For additional information, see step 3.
 - **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 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 about creating a query, see “Creating a Query in Microsoft Query” on page 139.

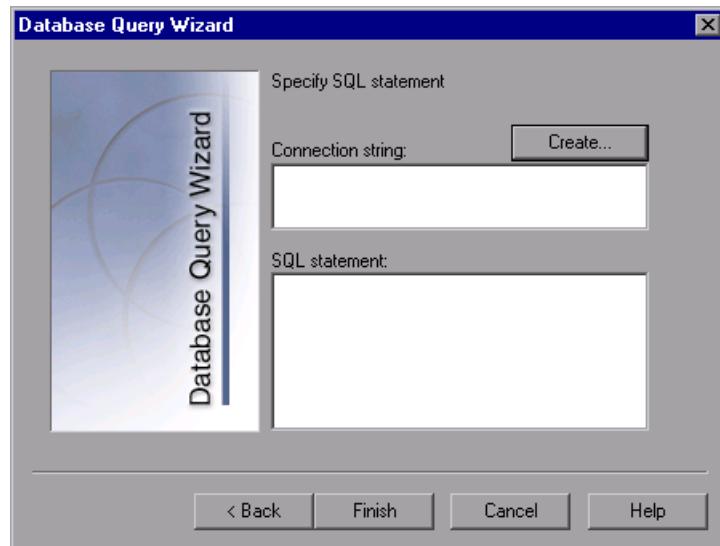
If you chose **Specify SQL statement** in the previous step, the **Specify SQL statement** screen opens. Specify the connection string and the SQL statement, and click **Finish**. For more information about specifying SQL statements, see “Specifying SQL Statements” on page 138.
- 4 The Checkpoint Properties dialog box opens. Select the checks to perform on the result set as described in “Understanding the Table/Database Checkpoint Properties Dialog Box” on page 140. 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.

Specifying SQL Statements

You can manually specify the database connection string and the SQL statement.

To specify SQL statements:

- 1 Choose **Specify SQL statement** in the Database Query Wizard screen. 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 Return to step 4 in the previous procedure to continue creating your database checkpoint in QuickTest.

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, choose **File > Exit and return to QuickTest Professional** to close Microsoft Query and return to QuickTest.
- 4** Return to step 4 on page 137 to continue creating your database checkpoint in QuickTest.

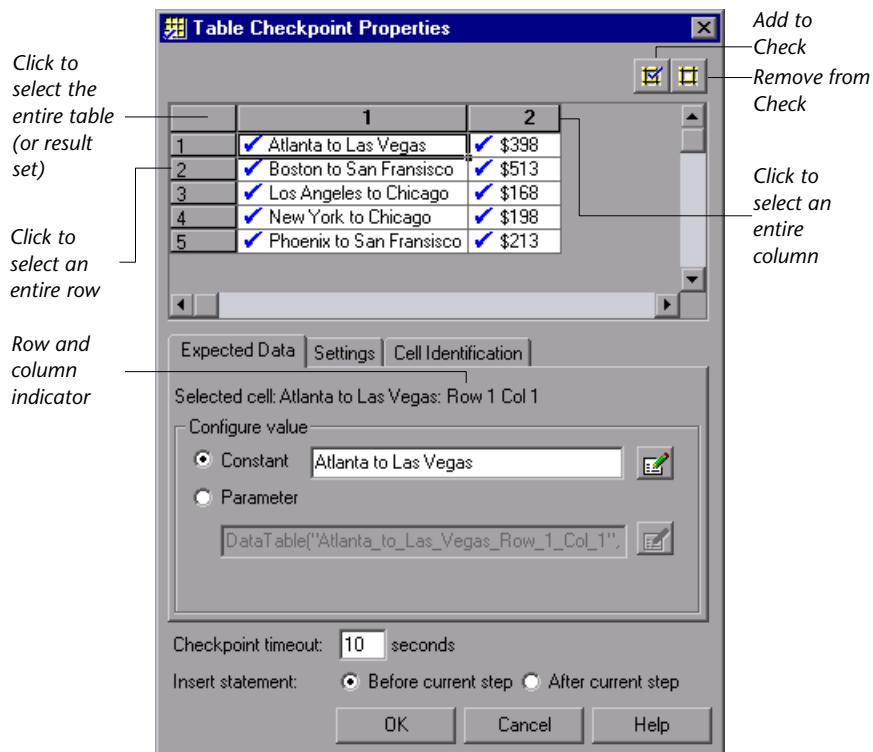
For additional information on working with Microsoft Query, refer to your Microsoft Query documentation.

Understanding the Table/Database Checkpoint Properties Dialog Box

The Table/Database Checkpoint Properties dialog box enables you to specify which cell contents of your table or 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.

Note: The Table Checkpoint Properties dialog box for an ActiveX table looks slightly different from the dialog box shown below. For an ActiveX table, you can also choose whether to check the properties of the table object (using the Properties tab), in addition to or instead of checking the table content (using the Table Content tab). Select the **Select Table Content** check box and/or the **Select Properties** check box, at the top of the relevant tabs in the Table Checkpoint Properties dialog box, depending on the type of information you want to check.

For information on the options in the Table Content tab, see the sections below. For information on the options in the Properties tab, see “Understanding the Checkpoint Properties Dialog Box” on page 126.



The top part of the Table/Database Checkpoint Properties dialog box displays the data that was captured for the checkpoint. You use this area to specify which cells you want to check. For more information, see “Specifying Which Cells to Check,” on page 144.

Below the expected data, the Table/Database Checkpoint dialog box contains the following three tabs:

- **Expected Data**—Enables you to set each checked cell as a constant or parameterized value. For example, you can instruct QuickTest to use a value from the Data Table as the expected value for a particular cell. For more information, see “Specifying the Expected Data” on page 145.

- **Settings**—Enables you to set the criteria for a successful match between the expected and actual 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. For more information, see “Specifying the Value Type Criteria” on page 146.
- **Cell Identification**—Enables you to instruct QuickTest how to locate 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 Table/Database Checkpoint Properties dialog box. However, you know that each time you run your test or component, 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*. For more information, see “Specifying the Cell Identification Settings” on page 147.

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 table or database, create separate checkpoints and specify the relevant cells for each.

The bottom part of the Table/Database 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 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 data to load in a table. Increasing the checkpoint timeout value in this case can ensure that the data has sufficient time to load and therefore enables the checkpoint to pass before the end of the timeout period is reached.

You can see information about the checkpoint timeout, including the time interval used by QuickTest to perform the checkpoint, in the Test Results window.

Note: The **Checkpoint timeout** option is available only when creating a table checkpoint. It is not available when creating a database checkpoint.

- **Insert statement**—Specifies when to perform the checkpoint in the test or component. Choose **Before current step** if you want to check the table or database content before the highlighted step is performed. Choose **After current step** if you want to check the table or database content 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 checkpoint. It is available when adding a new checkpoint to an existing test or component.

Specifying Which Cells to Check

The top part of the Table/Database Checkpoint Properties dialog box displays a grid representing the cells in the table or 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 table/database checkpoint, all cells contain a blue check mark, indicating they are selected for verification. You can select to check the entire table or results set, specific rows, specific columns, or specific cells.

To add to check or to remove from check:	Double-click:
a single cell	the cell
an entire row	the row header
an entire column	the column header
the entire result set	the top-left corner of the grid



Alternatively, you can select a range of cells and click the **Add to Check** button to check the selected cells or click the **Remove from Check** button to remove the selected cells from the check.

QuickTest checks only cells containing a check mark.

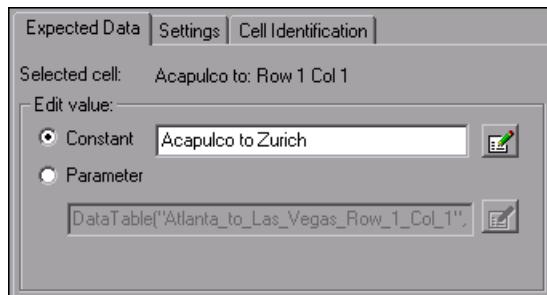
Notes:

When more than one cell is selected, the options on the Expected Data tab are disabled.

Double-clicking on the grid toggles the settings of each cell in the selection.

Specifying the Expected Data

The Expected Data tab displays options for setting the expected value of the selected cell in the table or 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 during the run session. If the expected value and the actual value do not match, the checkpoint fails.

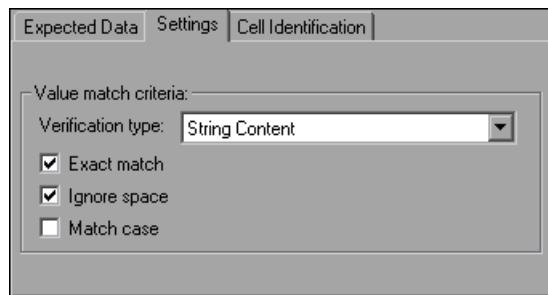
To modify or parameterize several cells in the table, select a cell and then set your preferences for that cell in the Expected Data tab. Repeat the process for each cell you want to modify.

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 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 286.

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.

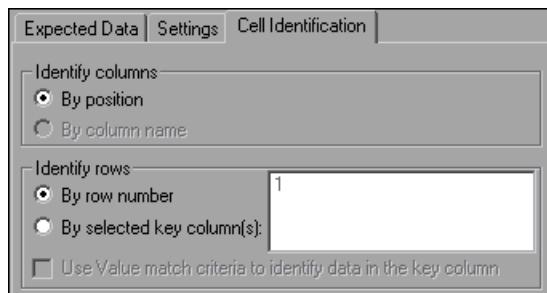
The Settings tab includes the following options:

Option	Description
Verification type	<p>Specifies how cell contents are compared:</p> <ul style="list-style-type: none">• 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. The settings in this tab apply to all selected cells.



The Cell Identification tab includes the following options:

Identify columns	<p>Specifies the column location of the cell(s) in your actual database to which you want to compare the expected data.</p> <ul style="list-style-type: none"> • By position—Locates cells according to the column position. A shift in the position of the columns within the database results in a mismatch. Always selected for table checkpoints. • By column name—Default for database checkpoints. Not available for table checkpoints. 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	<p>Specifies by row the location of the cell(s) in your actual database to which you want to compare the expected data.</p> <ul style="list-style-type: none"> • 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 <i>key column</i>. 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. <p>Note: A key symbol  is displayed in the header of selected key columns.</p>
Use value match criteria to identify data in the key column	<p>Instructs QuickTest to use the verification type settings from the Settings tab as the criteria for identifying data in the key column.</p> <p>Enabled only when you select to identify rows By selected key column(s).</p>

Modifying a Table Checkpoint

You can change the expected data, settings and cell identification options for an existing table checkpoint.

To modify the table checkpoint:

- 1 In the Keyword View or Expert View, right-click the table checkpoint that you want to modify.
- 2 Select **Checkpoint Properties**. The Checkpoint Properties dialog box opens.

Modify the settings as described in “Understanding the Table/Database Checkpoint Properties Dialog Box” on page 140.

Modifying a Database Checkpoint

You can make the following changes to an existing database checkpoint:

- Modify the SQL query definition.
- Modify the expected data, verification type, or method.

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.
 - 2 Select **Checkpoint Properties**. The Checkpoint Properties dialog box opens.
- Modify the settings as described “Understanding the Table/Database Checkpoint Properties Dialog Box” on page 140.

Checking Text

QuickTest can check that a text string is displayed in the appropriate place in an application or on a Web page.

This chapter describes:

- About Checking Text
- Creating a Text Checkpoint
- Creating a Standard Checkpoint for Checking Text
- Creating a Text Area Checkpoint
- Understanding the Text/Text Area Checkpoint Properties Dialog Box
- Modifying a Text or Text Area Checkpoint

About Checking Text

You can check that a specified text string is displayed by adding one of the following checkpoints to your test or component.

- **Text Checkpoint**—Enables you to check that the text is displayed in a screen, window, or Web page, according to specified criteria. It is supported for all environments.
- **Standard Checkpoint**—Enables you to check the **text** property of an object. This is the preferred way of checking text in many Windows applications. For more information, see “Creating Standard Checkpoints” on page 124.

- **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 Standard Windows, Visual Basic, and ActiveX environments.
-

Note: Text and text area checkpoints are also supported for various external QuickTest Professional add-ins (purchased separately). Refer to your add-in documentation for details.

Considerations for Using a Text Checkpoint for Windows-Based Applications

The text-recognition mechanism used when you create a text or text area checkpoint may occasionally retrieve unwanted text information (such as hidden text and shadowed text, which appears as multiple copies of the same string).

Additionally, text (or text area) checkpoints may behave differently in different run sessions depending on the operating system version you are using, service packs you have installed, other installed toolkits, the APIs used in your application, and so on.

Therefore, when possible, 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 that the above issues do not apply when working with Web-based applications.

Creating a Text Checkpoint

You can add a text checkpoint while recording a test or component in all environments. You can also add a text checkpoint on a Web page or frame from the Active Screen while editing your test or component.

Note: Text checkpoints are also supported for various external QuickTest Professional add-ins (purchased separately). Refer to your add-in documentation for details.

To add a text checkpoint while recording:

- 1 Display the page, window, or screen containing the text you want to check.
- 2 Choose **Insert > Checkpoint > Text Checkpoint**, or click the arrow next to the **Insert Checkpoint** button and choose **Text Checkpoint**.

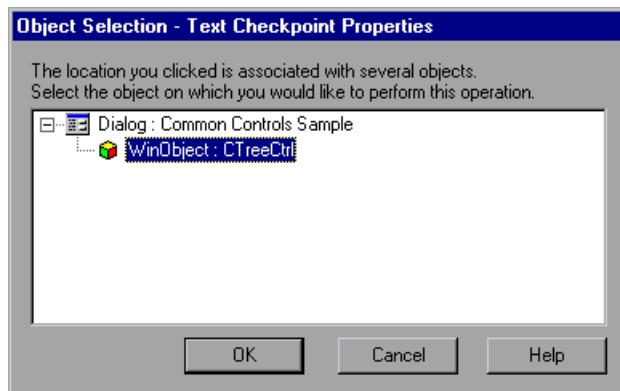


The QuickTest window is minimized and the mouse pointer turns into a pointing hand.

Tip: Hold the CTRL key to change the window focus or perform operations such as a right-click or mouseover to display a context menu. Note that pressing the CTRL key does not enable you to select an application from the Windows taskbar, therefore you must make sure that the window you want to access is not minimized.

- 3 Click the text string for which you want to create the checkpoint.

If the area you defined is associated with more than one object, the Object Selection–Text Checkpoint Properties dialog box opens.



Select the object for which you are creating the checkpoint.

- 4 The Text Checkpoint Properties dialog box opens.
- 5 Specify the checkpoint settings. For more information, see “Understanding the Text/Text Area Checkpoint Properties Dialog Box” on page 159.
- 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 text checkpoint while editing:

Note: You can insert text checkpoints using the Active Screen only for Web applications. This option is not supported for other environments.



- 1 Make sure the **Active Screen** 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 choose **Insert Text Checkpoint**. The Text Checkpoint Properties dialog box opens.
- 5 Specify the settings for the checkpoint. For more information, see “Understanding the Text/Text Area Checkpoint Properties Dialog Box” on page 159.
- 6 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 Standard Checkpoint for Checking Text

You can check the text in Windows-based applications by using a standard checkpoint to check the text property of an object. This is the preferred way of checking text in many Windows applications. Note, however, that the standard checkpoint does not allow you to use the features available in the Text Checkpoint Properties dialog box (see “Considerations for Using a Text Checkpoint for Windows-Based Applications” on page 152).

To add a standard checkpoint for checking text, while recording:



- 1 Click the **Insert Checkpoint** button or choose **Insert > Checkpoint > Standard Checkpoint**.

The QuickTest window is minimized, and the mouse pointer turns into a pointing hand.

Tip: Hold the CTRL key to change the window focus or perform operations such as a right-click or mouseover to display a context menu. Note that pressing the CTRL key does not enable you to select an application from the Windows taskbar, therefore you must make sure that the window you want to access is not minimized.

- 2 Click the object whose text you want to check. If the area you clicked is associated with more than one object, 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 Select the **text** property.
- 6 If necessary, edit the **text** value you want QuickTest to check. Note that you can parameterize this value.
- 7 If you only want to check text, clear the other check boxes in the dialog box.
- 8 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 for checking text while editing:

- 1 In the Keyword View, right-click the step for the object whose text you want to check, and choose **Insert Standard Checkpoint**. The Checkpoint Properties dialog box opens.
- 2 Select the **text** property.
- 3 If necessary, edit the text value you want QuickTest to check. Note that you can parameterize this value.
- 4 If you want to check only text, clear the other check boxes in the dialog box.
- 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 creating standard checkpoints, see Chapter 7, “Checking Object Property Values.”

Creating a Text Area Checkpoint

You can add a text area checkpoint only while recording a test or component on Windows-based applications—Standard Windows, Visual Basic, and ActiveX.

Note: Text area checkpoints are also supported for various external QuickTest Professional add-ins (purchased separately). Refer to your add-in documentation for details.

To add a text area checkpoint:

- 1** Choose **Insert > Checkpoint > Text Area Checkpoint**, or click the arrow next to the **Insert Checkpoint** button and choose **Text Area Checkpoint**.

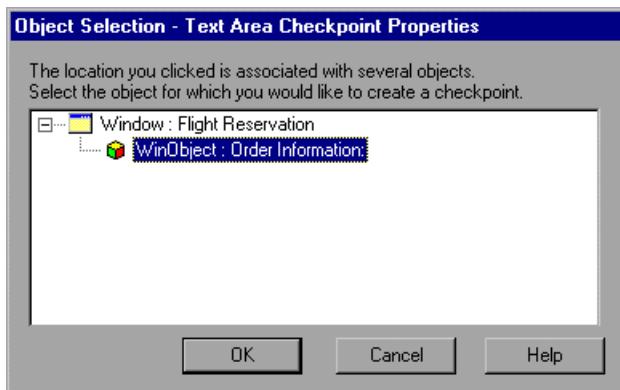
The QuickTest window is minimized, and the mouse pointer turns into a crosshairs pointer.

- 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,” below.)

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 “Understanding the Text/Text Area Checkpoint Properties Dialog Box” on page 159.
- 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 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 or component, it checks for the selected text within the defined area, according to the settings you configured.

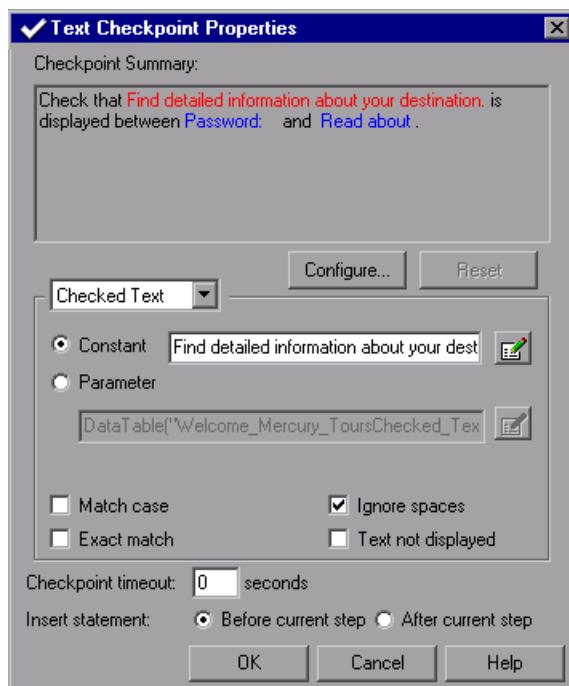
Consider the following when defining the area for a text area checkpoint:

- 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.

Understanding the Text/Text Area Checkpoint Properties Dialog Box

In the Text/Text Area Checkpoint Properties dialog box, you can specify the text to be checked as well as 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 or defined text area. 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.



The Checkpoint Summary pane at the top of the dialog box summarizes the selected text for the checkpoint. 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 pane 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.

QuickTest automatically displays the Checked Text in red and the text before and after the Checked Text in blue. For text area checkpoints, only the text string captured from the defined area is displayed (Text Before and Text After are not displayed).

To designate parts of the captured string as Checked Text and other parts as Text Before and Text After, click the **Configure** button. The Configure Text Selection dialog box opens.

For more information about configuring the text selection, see “Configuring the Text Selection” on page 161.

To set parameterization and other preferences for each of the string elements in your checkpoint, select the string element type (**Checked Text / Text Before / Text After**) from the list box and select your preferences. For more information, see “Setting Options for the Checked Text” on page 163, “Setting Options for Text Displayed Before the Checked Text” on page 164, and “Setting Options for Text Displayed After the Checked Text” on page 166.

The bottom part of the dialog box enables you to specify the time interval during which QuickTest attempts to perform the checkpoint successfully. You can also specify when to perform the checkpoint. For more information, see “Setting General Checkpoint Options” on page 167.

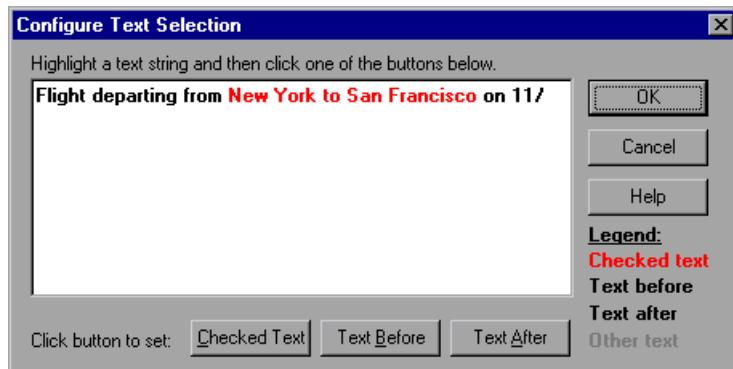
Configuring the Text Selection

You can configure the text selection displayed in the Checkpoint Summary pane using 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).
Checkpoint summary	Displays the text configuration you make using the Configure Text Selection dialog box.
Reset	Resets the text selection to the previous configuration.

The Configure Text Selection dialog box displays the text you captured when creating the text checkpoint. For text checkpoints, it also displays some 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).

For text area checkpoints, you can designate parts of the captured text as checked text, and other parts as text before and text after.



Note: If you are modifying an existing text checkpoint, the Configure Text Selection dialog box displays the existing text configuration.

Tip: If you want to configure more text than is displayed, cancel the text checkpoint and make a larger text selection in your Web page or application window.

You can specify the parts of the displayed text for the checkpoint by highlighting them and clicking one of the following buttons:

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: QuickTest displays in gray any text that is not configured as Checked Text, Text Before, or Text After. The gray text is not displayed the next time the Configure Text Selection dialog box is opened.

When you close the Configure Text Selection dialog box, the Checkpoint Summary pane displays the new text selection configuration.

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 the Checked Text

Choose **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. Choose from the following options for the checked text:

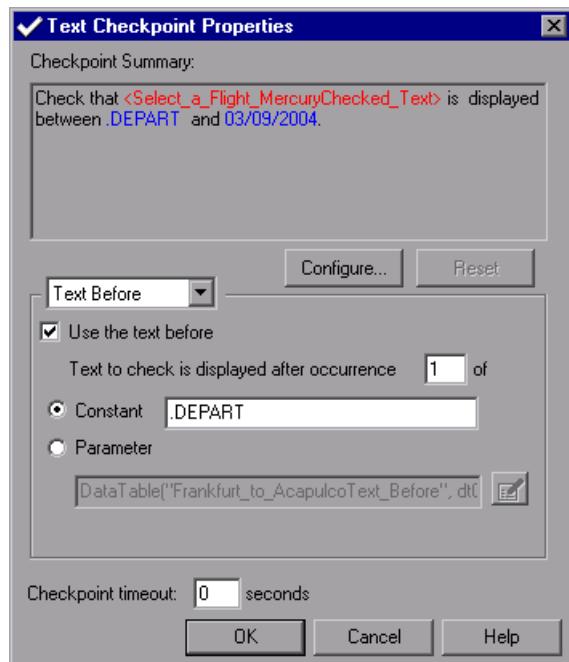
- **Constant** (default)—Sets the expected value of the checked text as a constant. For information on modifying values, see “Setting Values in the Configure Value Area” on page 286.
-

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 286.
- **Match case**—Conducts a case-sensitive check.
- **Exact match**—Checks for the exact expected text. For example, 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, 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

Choose **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 for 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 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 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 286.

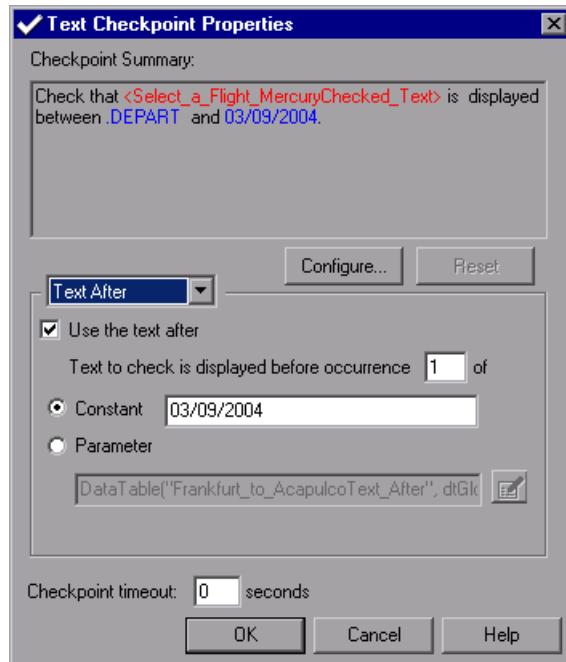
If you modify the text, use a string that is unique within the object whenever possible, so that the occurrence number is 1.

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 286.

Setting Options for Text Displayed After the Checked Text

Choose **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 for setting which text is 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. QuickTest starts counting occurrences of the **is displayed before** text you specify, from the end of the **is displayed after** string. In other words, it starts looking for the specified text from the text you selected to check.

If you accept the default text that QuickTest recommends, the number in the dialog box will be correct. If you modify the recommended text string and the string you specify is displayed in your highlighted text as well as in the **is displayed before** text, you need to modify the occurrence number accordingly.

For example, if you want to check that the words my hat is the best are displayed before the word hat, enter 2 in the **Text to check is displayed before occurrence** box, to show that you want your text to be displayed before the second occurrence of the word hat.

- **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 286.

If you modify the text, use a string that is unique within the object whenever possible, 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 286.

Setting General Checkpoint Options

The bottom part of the Text/Text Area 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 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 time for an object to achieve an expected state. Increasing the checkpoint timeout value in this case can ensure that the object has sufficient time to achieve that state and therefore enables the checkpoint to pass before the end of the timeout period is reached.

You can see information about the checkpoint timeout, including the time interval used by QuickTest to perform the checkpoint, in the Test Results window.

Note: The **Checkpoint timeout** option is only available when creating a text checkpoint. It is not available when creating a text area checkpoint.

- **Insert statement**—Specifies when to perform the checkpoint. Choose **Before current step** if you want to check the value of the text before the highlighted step is performed. Choose **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 or component 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, right-click the checkpoint that you want to modify.
- 2 Select **Checkpoint Properties**. The Text/Text Area Checkpoint Properties dialog box opens.
- 3 Modify the settings. For more information, see “Understanding the Text/Text Area Checkpoint Properties Dialog Box” on page 159.

10

Checking Bitmaps

QuickTest enables you to compare objects in a Web page or application by matching captured bitmaps.

This chapter describes:

- About Checking Bitmaps
- Checking a Bitmap
- Modifying a Bitmap Checkpoint

About Checking Bitmaps

You can check an area of a Web page or application as a bitmap. While creating a test or component, you specify the area you want to check by selecting an object. You can check an entire object or any area within an object. QuickTest captures the specified object as a bitmap, and inserts a checkpoint in the test or component. You can also choose to save only the selected area of the object with your test or component in order to save disk space.

When you run the test or component, QuickTest compares the object or selected area of the object currently displayed on the Web page or application with the bitmap stored when the test or component was recorded. If there are differences, QuickTest captures a bitmap of the actual object and displays it with the expected bitmap in the details portion of the Test Results window. By comparing the two bitmaps (expected and actual), you can identify the nature of the discrepancy. For more information on test results of a checkpoint, see “Viewing Checkpoint Results” on page 557.

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).

Note: The results of bitmap checkpoints may be affected by factors such as operating system, screen resolution, and color settings.

Checking a Bitmap

You can add a bitmap checkpoint while recording or editing your test or component.

To create a bitmap checkpoint while recording:



- 1 Choose **Insert > Checkpoint > Bitmap Checkpoint** or click the arrow beside the **Insert Checkpoint** button and choose **Bitmap Checkpoint**.

The QuickTest window is minimized and the mouse pointer turns into a pointing hand.

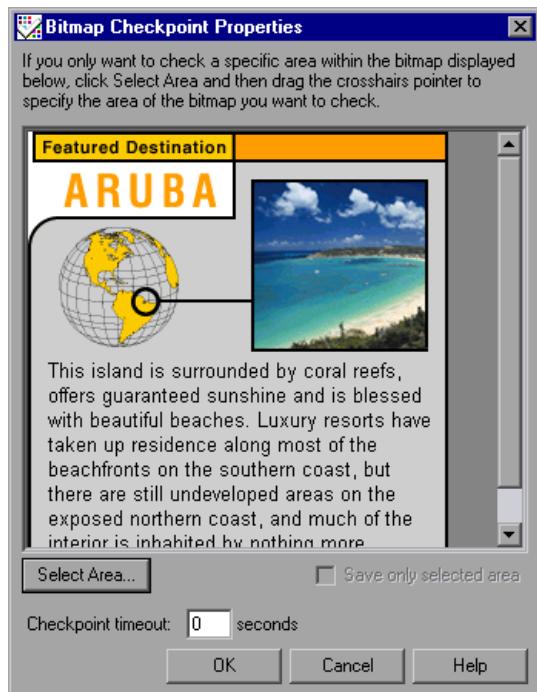
Note: If the object you want to check can only be displayed by performing an event (such as a right-click or a mouse over to display a context menu), hold the CTRL key. The pointing hand temporarily turns into a standard arrow and you can perform the event. When the object you want to check is displayed, release the CTRL key. The arrow becomes a pointing hand again. You can also use the CTRL key to change the window focus. Note that pressing the CTRL key does not enable you to select an application from the Windows taskbar, therefore you must make sure that the window you want to access is not minimized.

- 2 Click an object to check in your Web page or application. If the location you click is associated with more than one object, the Object Selection - Bitmap Checkpoint Properties dialog box opens.



Tip: Select an object from the tree on which to create a bitmap checkpoint. If you want to create a bitmap checkpoint of multiple objects, you should select the highest level object that includes all the objects to include in the bitmap checkpoint.

- 3 Click **OK**. The Bitmap Checkpoint Properties dialog box opens.



A bitmap of the object you selected in the previous step is displayed in the dialog box.

- 4 If you want to check a specific area of the object, click the **Select Area** button. Use the crosshairs pointer to specify the area you want to select. This instructs QuickTest to check only the selected area and to ignore the remainder of the bitmap. The Test Results window displays the bitmap with this area highlighted.

- 5 If you want to save only the selected area of the object with your test or component (to save disk space), select the **Save only selected area** check box. The Test Results window displays only the selected area of the bitmap.
-

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** option (**Test > Update Run**) 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.

- 6 Specify the **Checkpoint Timeout** if you want 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 ensure that the object has sufficient time to achieve that state and therefore enables the checkpoint to pass before the end of the timeout period is reached.

You can see information about the checkpoint timeout, including the time interval used by QuickTest to perform the checkpoint, in the Test Results window.

- 7 Click **OK** to add the bitmap checkpoint to your test or component. A checkpoint statement is added for the selected object in the Keyword View and Expert View.

To create a bitmap checkpoint while editing:



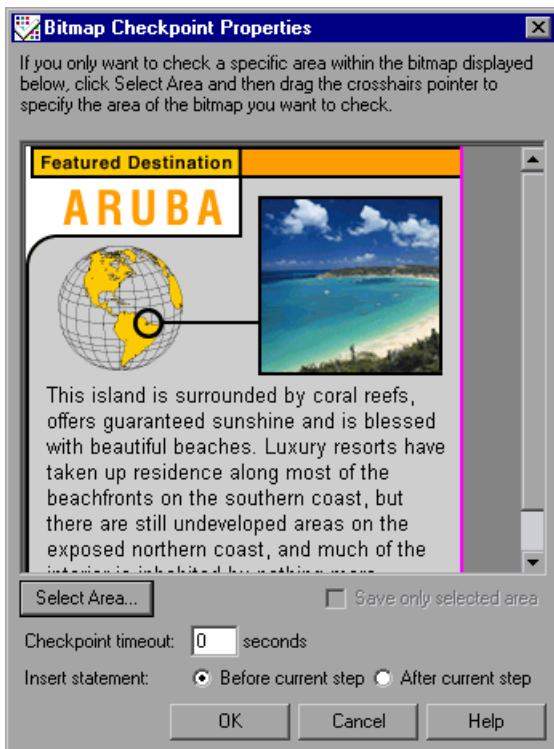
- 1** Make sure the **Active Screen** button is selected.
- 2** Click a step in the Keyword View to add a checkpoint. The Active Screen displays the Web page or application corresponding to the highlighted step.
- 3** Right-click an object in the Active Screen and choose **Insert Bitmap Checkpoint**. If the location you click is associated with more than one object, the Object Selection - Bitmap Checkpoint Properties dialog box opens.



Note: Select an object from the tree on which to create a bitmap checkpoint. Ensure that the object you select is completely visible. Otherwise, if another application is overlapping the object, it will also be captured.

Tip: If you want to create a bitmap checkpoint of multiple objects, you should select a parent object that includes all the objects to include in the bitmap checkpoint.

- 4 Click **OK**. The Bitmap Checkpoint Properties dialog box opens.



A bitmap of the object you selected in the previous step is displayed in the dialog box.

- 5 If you want to check a specific area of the object, click the **Select Area** button. Use the crosshairs pointer to specify the area you want to select. This instructs QuickTest to check only the selected area and to ignore the remainder of the bitmap. The Test Results window displays the bitmap with this area highlighted.

- 6 If you want to save only the selected area of the object with your test or component (to save disk space), select the **Save only selected area** check box. The Test Results window displays only the selected area of the bitmap.
-

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** option (**Test > Update Run**) 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.

- 7 Specify the **Checkpoint Timeout** if you want 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 ensure that the object has sufficient time to achieve that state and therefore enables the checkpoint to pass before the end of the timeout period is reached. You can see information about the checkpoint timeout, including the time interval used by QuickTest to perform the checkpoint, in the Test Results window.

-
- 8** Choose to insert the bitmap checkpoint before or after the highlighted step.

Notes:

Choose **Before current step** if you want to check the bitmap before the highlighted step is performed. Choose **After current step** if you want to check the bitmap after the highlighted step is performed.

The **Insert statement** option is not available when adding a new bitmap checkpoint during recording or when modifying an existing bitmap checkpoint. It is available only when adding a new bitmap checkpoint to an existing test or component.

- 9** Click **OK** to add the bitmap checkpoint. A checkpoint statement is added for the selected object in the Keyword View and Expert View.

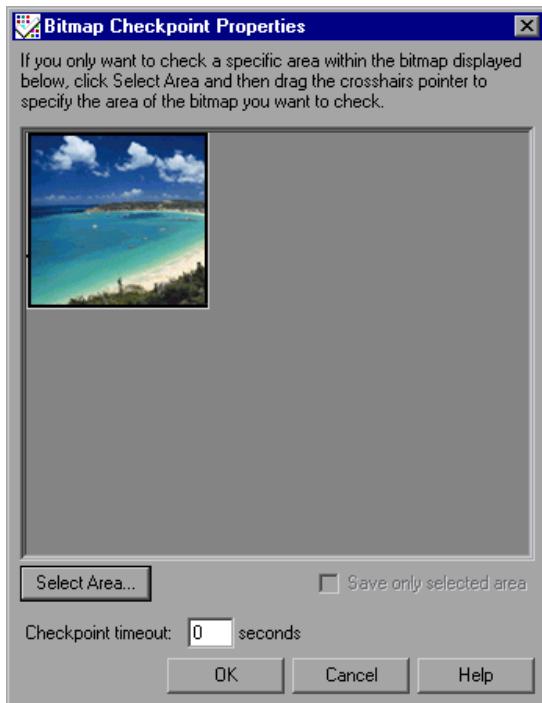
Modifying a Bitmap Checkpoint

You can modify an existing bitmap checkpoint.

Note: If you selected the **Save only selected area** check box when you created or previously modified the checkpoint, then you can only modify the checkpoint by selecting a smaller area within the bitmap; you cannot return the bitmap to its former size. The **Update Run** option (**Test > Update Run**) 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.

To modify a bitmap checkpoint:

- 1 In the Test pane, right-click the bitmap checkpoint that you want to modify and select **Checkpoint Properties**. The Bitmap Checkpoint Properties dialog box opens and displays the object or area you saved with the checkpoint.



- 2 Click the **Select Area** button. Use the crosshairs pointer to specify the area you want to select. This instructs QuickTest to check only the selected area and to ignore the remainder of the bitmap. The Test Results window displays the bitmap with this area highlighted.
- 3 If you want to save only the newly selected area of the object with your test or component (to save disk space), select the **Save only selected area** check box. The Test Results window displays only the selected area of the bitmap.

- 4 Specify the **Checkpoint Timeout** if you want 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 ensure that the object has sufficient time to achieve that state and therefore enables the checkpoint to pass before the end of the timeout period is reached.

You can see information about the checkpoint timeout, including the time interval used by QuickTest to perform the checkpoint, in the Test Results window.

- 5 Click **OK** to modify the checkpoint.

11

Checking XML

By adding XML checkpoints to your test or component, you can check the contents of individual XML data files or documents that are part of your Web application.

This chapter describes:

- ▶ About Checking XML
- ▶ Creating XML Checkpoints
- ▶ Modifying XML Checkpoints
- ▶ Reviewing XML Checkpoint Results
- ▶ Using XML Objects and Methods to Enhance Your Test or Component

About Checking XML

XML (Extensible Markup Language) is a meta-markup language for text documents that is endorsed as a standard by the 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 by inserting XML checkpoints. A few common uses of XML checkpoints are described below:

- An XML file can be a static data file that is accessed in order 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 structure (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.
- 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 in order to check an XML document or file whose data changes in a predictable way.
- XML documents and files often need a well-defined structure in order 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.

Creating XML Checkpoints

You can create two types of XML checkpoints:

- **XML Web Page/Frame Checkpoint**—Checks an XML document within a Web page or frame.
- **File Checkpoint**—Checks a specified XML file.

Creating XML Web Page/Frame Checkpoints

You can create an XML Web page/frame checkpoint for any XML document contained in a Web page or frame. Note that you can create an XML Web page/frame checkpoint only while recording.

To create an XML Web page/frame checkpoint:

- 1 Begin recording your test or component.
- 2 Choose **Insert > Checkpoint > XML Checkpoint (Web Page/Frame)**, or click the **Insert Checkpoint** arrow and select **XML Checkpoint (Web Page/Frame)**.



Note: The **XML Checkpoint (Web Page/Frame)** option is available only when the Web Add-in is installed and loaded. For more information on loading add-ins, see Chapter 29, “Working with QuickTest Add-Ins.”

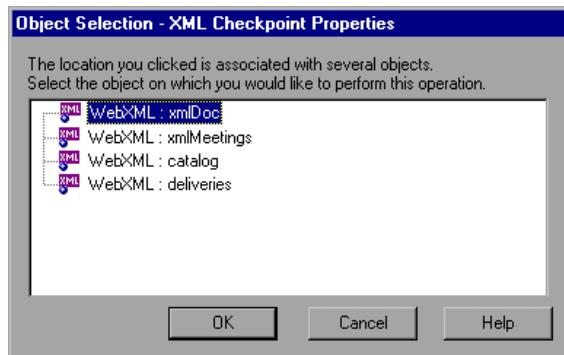
The QuickTest window is minimized and the mouse pointer becomes a pointing hand.

Tip: Hold the CTRL key to change the window focus or perform operations such as a right-click or mouseover to display a context menu. Note that pressing the CTRL key does not enable you to select an application from the Windows taskbar, therefore you must make sure that the window you want to access is not minimized.

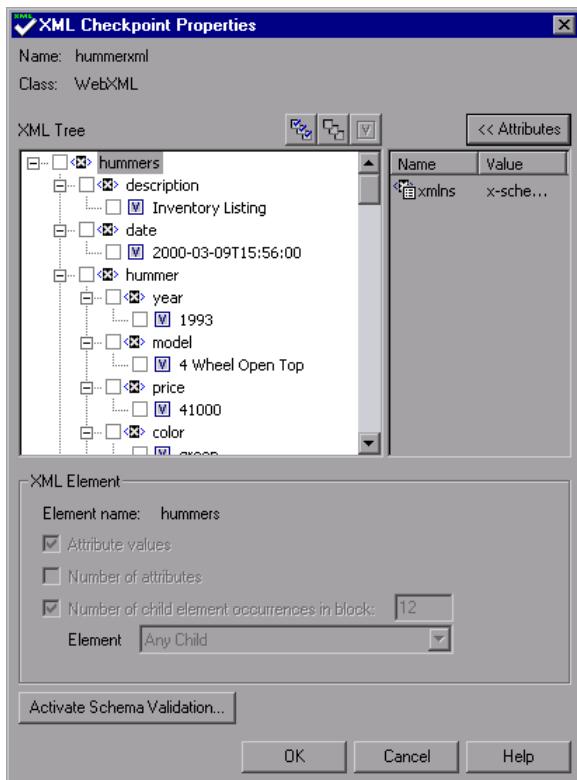
- 3** Click a Web page or frame to check the XML documents associated with the page.

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.



The XML Checkpoint Properties dialog box displays the element hierarchy and values (character data) of the selected XML document.

Note: QuickTest loads large XML files in the background. While the file is loading, an indication of this is shown in the XML Checkpoint Properties dialog box, and you can only select XML nodes that are already loaded.

- 5 Select the element(s), attribute(s), and/or value(s) 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.
For more information on these options, see “Understanding the XML Checkpoint Properties Dialog Box” on page 190.
- 6 If you want to check that the XML structure adheres to a specific XML schema, click **Activate Schema Validation** and set the required options. For more information on these options, see “Understanding the Schema Validation Dialog Box” on page 197.
- 7 Click **OK** to add the XML checkpoint. A checkpoint similar to the following is added to the Keyword View.

Item	Operation	Value
Action1		
Simple XML Example		
Simple XML Example		
contents		
XML AccessoriesXML	Check	CheckPoint("AccessoriesXML")

QuickTest records this step in the Expert View as:

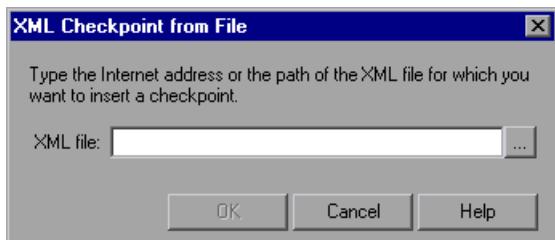
```
Browser("Simple XML Example").Page("Simple XML Example").
Frame("contents").WebXML("AccessoriesXML").
Check CheckPoint("AccessoriesXML")
```

Creating XML File Checkpoints

You create XML file checkpoints in order to directly access and verify specified XML files in your system. Note that you can create an XML file checkpoint while you are recording or editing your test or component.

To create an XML file checkpoint:

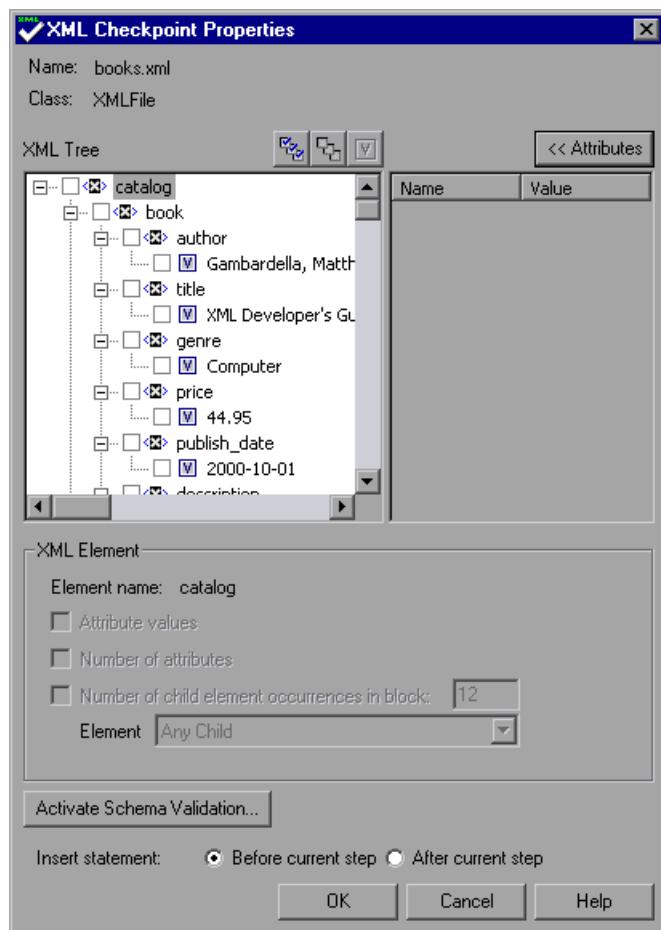
- 1 Choose **Insert > Checkpoint > XML Checkpoint (File)**, or click the **Insert Checkpoint** arrow and select **XML Checkpoint (File)**. The XML Checkpoint from File dialog box opens.
 A small icon consisting of a blue square with a white checkmark and a downward-pointing arrow, positioned to the left of the first list item.



- 2 In the **XML file** box, enter the file path or Internet address of the XML file. Alternatively, click the browse button to 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.

Note: You can enter a relative path and QuickTest will search for the XML file in the folders listed in the Folders tab of the Options dialog box. Once QuickTest locates the file, it saves it as an absolute path and uses the absolute path during the run session. For more information, see Chapter 24, "Setting Folder Testing Options."

- 3 Click **OK**. The XML Checkpoint Properties dialog box opens.



The XML Checkpoint Properties dialog box displays the element hierarchy and values (character data) of the selected XML file.

Note: QuickTest loads large XML files in the background. While the file is loading, an indication of this is shown in the XML Checkpoint Properties dialog box, and you can only select XML nodes that are already loaded.

- 4** Select the element(s), attribute(s), and/or value(s) 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. For more information on these options, see “Understanding the XML Checkpoint Properties Dialog Box” on page 190.
- 5** If you want to check that the XML structure adheres to a specific XML schema, click **Activate Schema Validation** and set the required options. For more information on these options, see “Understanding the Schema Validation Dialog Box” on page 197.
- 6** If you are inserting the checkpoint while editing, choose whether you want to insert the XML checkpoint before or after the highlighted step. Choose **Before current step** if you want to check the XML file before the highlighted step is performed. Choose **After current step** if you want to check the XML file 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. It is available only if you are adding a new XML checkpoint to an existing test or component.

- 7** Click **OK** to add the XML checkpoint. A checkpoint similar to the following is added to the Keyword View.

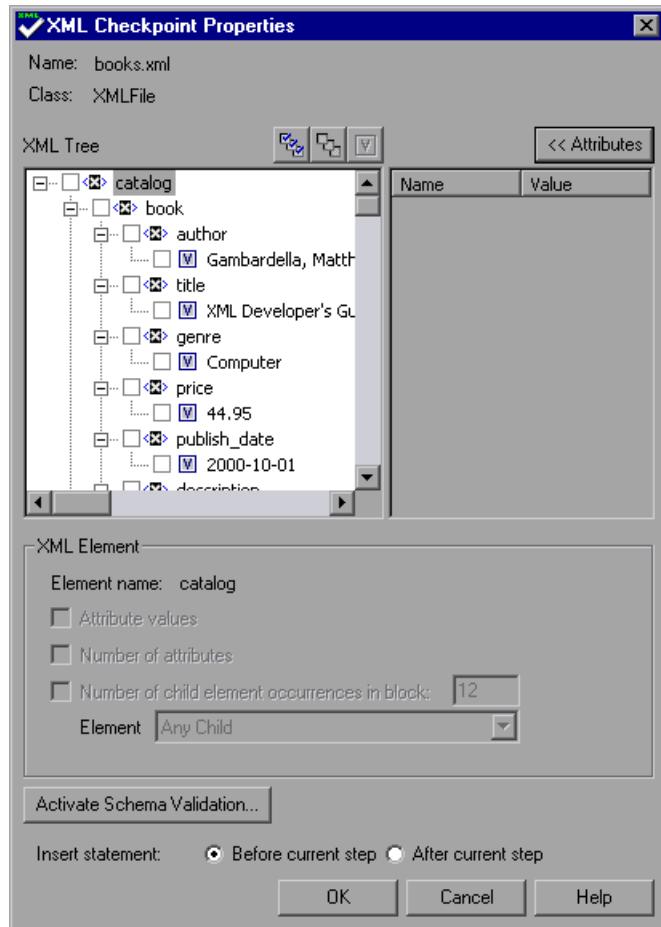
Item	Operation	Value	Documentation
▼ Action1			Call the Action1 action.
└ books.xml	Check	CheckPoint("books.xml")	Check whether the content of the specified xml file match

QuickTest inserts this step as follows in the Expert View:

```
XMLFile("books.xml").Check CheckPoint("books.xml")
```

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.



Note: QuickTest loads large XML files in the background. While the file is loading, an indication of this is shown in the XML Checkpoint Properties dialog box. You can only select XML nodes that are already fully loaded.

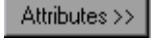
Identifying the Object

The top part of the dialog box displays test object information about the selected XML document or file:

Option	Description
Name	The name assigned to the checkpoint and step.
Class	The test object class. This is either XMLFile (for files) or WebXML (for Web pages or frames).

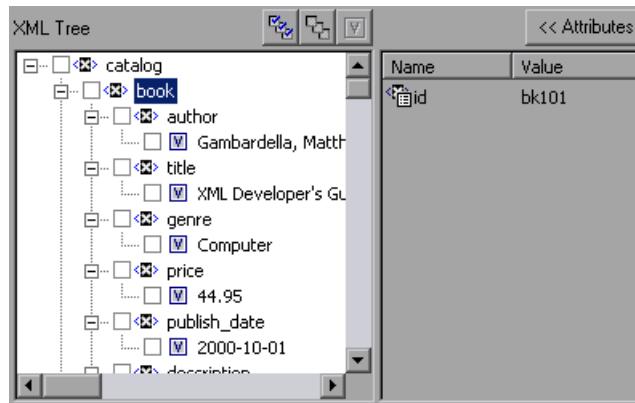
XML Checkpoint Control Buttons

In the right-hand corner, above the XML Tree, there are four control buttons:

Option	Description
 Select All	Selects all elements and values in the XML Tree.
 Clear All	Clears all elements and values in the XML Tree.
 Open Value Dialog	Opens the Element Value dialog box, enabling you to view the complete value of each element. This button is enabled only when a value item is selected. Note that you can also open the Element Value dialog box by double-clicking a value tree item. For more information, see “Understanding the Element Value Dialog Box” on page 196.
 Attributes	Expands the window in order to display the attributes of the selected XML element. The attributes are displayed to the right of the XML Tree. Note that you can click this button again to hide the attributes.

XML Tree Pane

The XML Tree pane displays the hierarchy of the XML file, enabling you to select the elements, attributes and/or values you want to check.



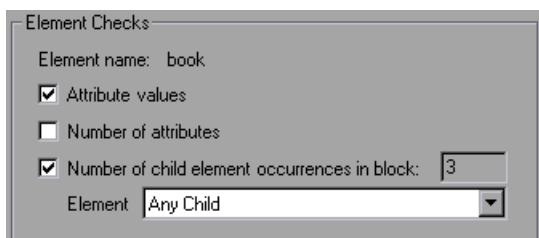
Option	Description
XML Tree	The XML Tree displays the hierachal relationship between each element and value in the XML document or file. Each element is displayed with a icon. Each value is displayed with a icon.
Attributes Pane	When you click the Attributes >> button, the attributes associated with the selected element are displayed in the Attributes pane to the right of the XML Tree. The attributes pane lists the name and value of each attribute. Note that you can click this button again to hide the Attributes pane.

Checkpoint Options Pane

The checkpoint options pane enables you to select the types of checks you want to perform on selected elements and/or the parameterization options you want to set for selected values and attributes.

► Element Checks

When you select an element in the XML Tree, the checkpoint options pane displays the Element Checks area, which includes the name of the selected element and the available element checks.



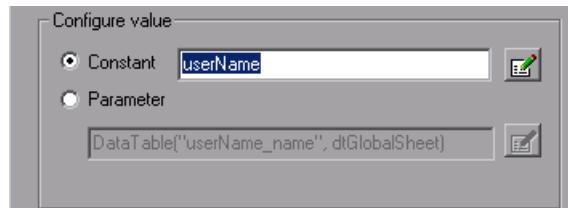
The following element checks are available:

Option	Description
Attribute values	Checks the values of the attributes that are attached to the element (selected by default).
Number of attributes	Checks the number of attributes that are attached to the element.

Option	Description
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 document or file (with the specified name, if applicable) corresponds to the number that appears in the read-only Number of child element occurrences in block field. (selected by default)
Element	Specifies 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 appears in the read-only 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.

► Configure Value Checks

When you select an element value or attribute in the XML Tree or attributes pane, the checkpoint options pane displays the **Configure Value** area, enabling you to define the value as a constant or parameterize the value or attribute.



For more information about modifying values, see “Setting Values in the Configure Value Area” on page 286.

The following **Configure Value** options are available:

Option	Description
Constant	Sets the expected value of the selected element value (character data) or attribute as a constant.
Parameter	Sets the expected value of the selected element value (character data) or attribute as a parameter.

For more information on parameterizing values and attributes, see Chapter 12, “Parameterizing Values.”

► **Activate Schema Validation**

You can use the **Activate Schema Validation** button to confirm that the XML in your application or file adheres to the XML structure defined in a specific XML schema or schemas. You can validate the XML structure using one or more external schema files or using schema(s) embedded within your XML document. For more information, see “Understanding the Schema Validation Dialog Box” on page 197.

► **Insert statement options**

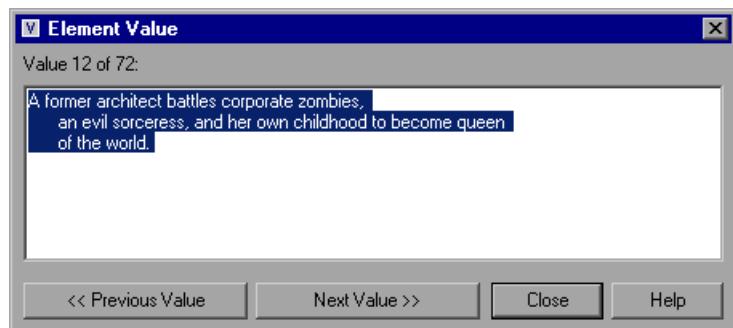
If you are inserting a checkpoint while editing your test or component, the bottom part of the XML Checkpoint Properties dialog box displays the **Insert statement** options, enabling you to choose whether you want to insert the XML checkpoint before or after the step that you highlighted. Choose **Before current step** if you want to check the value of the text before the highlighted step is performed. Choose **After current step** 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 or component.

Understanding the Element Value Dialog Box

The XML Tree in the XML Checkpoint Properties dialog box and XML Output Value dialog box displays values in a single line. There is limited space for a value to be displayed in both the XML Tree and tooltip (which is displayed when you hold your mouse over the value). The Element Value dialog box enables you to view multi-line values and/or values that are very long.

The following is an example of an Element Value dialog box for a multi-line value with several line breaks.

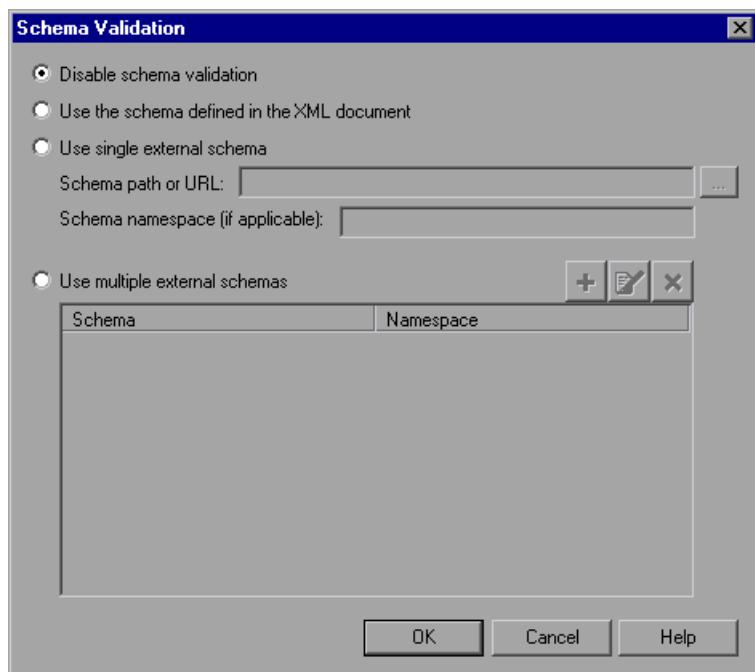


The Element Value dialog box contains the following options:

Option	Description
<input type="button" value="Next Value >>"/>	Enables you to navigate forward through the element values in the XML Tree. Clicking this button displays the next value in the XML Tree.
<input type="button" value="<< Previous Value"/>	Enables you to navigate backward through the element values in the XML Tree. Clicking this button displays the previous value in the XML Tree.

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 structure of the XML in your application or file.



The Schema Validation dialog box contains the following options:

- **Disable schema validation**—Specifies that you do not want to validate the XML in your application or file against an XML schema. This is the default option.
- **Use the schema defined in the XML document**—Instructs QuickTest to use the schema or schemas defined within your XML document to validate the structure of the XML in your Web page/frame or file.
- **Use single external schema**—Instructs QuickTest to use an external XML schema file to validate the structure of your XML. If you select this option, 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 or file. You can specify a schema file either from your file system or from Quality Center.
- **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 than the one specified in the schema file, the validation will fail.
- **Use multiple external schemas**—Instructs QuickTest to use multiple external XML schema files to validate the structure of your XML. You can specify a schema file either from your file system or from Quality Center. For each external file you want to use, you must specify its path or URL and namespace.

If you select this option, the following toolbar buttons are enabled:

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 200.
	Enables you to modify the details of an external schema file in the list. For more information, see “Understanding the Edit Schema Dialog Box” on page 200.
	Enables you to remove an external schema file from the list.

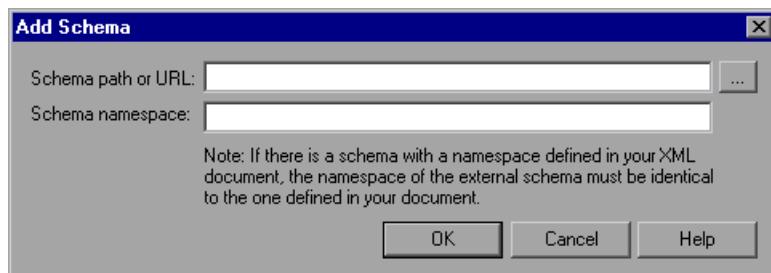
Guidelines for Schema Validation

Following are specific guidelines that you should adhere to when specifying a schema file to validate your XML.

- If there is more than one schema embedded in the XML, QuickTest recognizes each individual schema and compares the relevant section of the XML document against the schema for the corresponding section.
- When specifying multiple external schemas either from your file system or from Quality Center, the paths cannot include spaces. When specifying multiple external schemas from Quality Center, QuickTest ignores the "[Quality Center]" part of the file path, but the remainder of the path must not contain spaces.
- 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 tab of the Options dialog box. For more information, see Chapter 24, "Setting Folder Testing Options."
- 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.
- 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 defined in the document, 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 will cause the schema validation to fail.

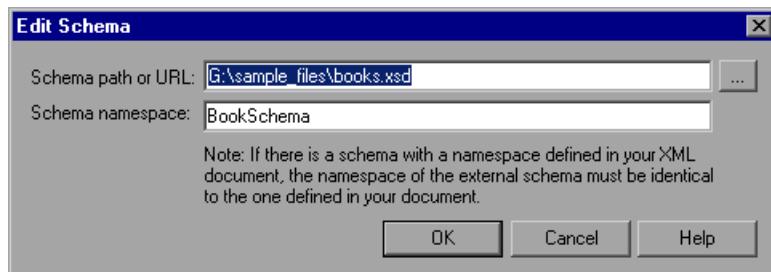
Understanding the Add Schema Dialog Box

The Add Schema dialog box enables you to specify the path or URL of an external schema file. For each external schema file, you must also specify its namespace.



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. You can also modify 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.
- 2 Select **Checkpoint Properties**. The XML Checkpoint Properties dialog box opens.
- 3 Modify the settings as described in the previous sections.

Reviewing XML Checkpoint Results

By adding XML checkpoints to your tests and components, you can verify that the data and structure in your XML documents or files has not changed unexpectedly. When you run your test or component, 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 564.

Using XML Objects and Methods to Enhance Your Test or Component

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 can also use these methods to create and manipulate new XML objects. You do this by entering XML statements in the Expert View. For more information about programming in the Expert View, see Chapter 36, “Working with the Expert View.”

For additional information on XML objects and methods, refer to the Supplemental section of the *QuickTest Professional Object Model Reference*.

12

Parameterizing Values

QuickTest enables you to expand the scope of a basic test or component by replacing fixed values with parameters. This process, known as *parameterization*, greatly increases the power and flexibility of your test or component.

This chapter describes:

- ▶ About Parameterizing Values
- ▶ Parameterizing Values in Steps and Checkpoints
- ▶ Using Test, Action, and Component Input Parameters
- ▶ Using Data Table Parameters
- ▶ Using Environment Variable Parameters
- ▶ Using Random Number Parameters
- ▶ Example of a Parameterized Test
- ▶ Using the Data Driver to Parameterize Your Test

About Parameterizing Values

You can use the parameter feature in QuickTest to enhance your test or component 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 values in steps and checkpoints in your test or component. You can also parameterize the values of action parameters.

If you wish to parameterize the same value in several steps in your test or component, you may want to consider using the Data Driver rather than adding parameters manually.

There are four types of parameters:

- **Test, action or component parameters** enable you to use values passed from your test or component, or values from other actions in your test.
 - In order 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 or component.

For example, suppose that you want to parameterize a step in Action3 using a value that is passed into your test from the external application that runs (calls) your test. You can pass the value from the test level to Action1 (a top-level action) to Action3 (a child action of Action1), and then parameterize the required step using this Action input parameter value (that was passed through from the external application).

- **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 or Web site 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 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 machine running the test or component.

- **Random number parameters** enable you to insert random numbers as values in your test or component. 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 field.

Parameterizing Values in Steps and Checkpoints

You can parameterize values in steps and checkpoints while recording or editing your test or component.

You can parameterize the values of object properties for a selected step. You can also parameterize the values of the operation (method or function arguments) defined for the step.

For example, your application or Web site may include a form with an edit field into which the user types the user name. You may want to test whether your application or Web site reads this information and displays it correctly in a dialog. 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, you are modifying the test object description in the object repository. Therefore, all occurrences of the specified object within the test, action, or component are parameterized. For more information on the object repository, see Chapter 4, "Managing Test Objects."

Parameterizing the value of a checkpoint property enables you to check how an application or Web site 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 or component with a separate checkpoint for each destination, you can add a Data Table parameter for the destination information. For each iteration of the test or component, QuickTest checks the flight information for a different destination.

For more information on using checkpoints, see Chapter 6, “Understanding Checkpoints.”

When you define a value as a parameter, you first select the value and then specify the parameter type and its settings.

For more information on using specific parameter types, see:

- “Using Test, Action, and Component Input Parameters” on page 211
- “Using Data Table Parameters” on page 215
- “Using Environment Variable Parameters” on page 222
- “Using Random Number Parameters” on page 233

You can parameterize values in existing steps and checkpoints for object properties, checkpoint properties, and operations (method and function arguments). For more information, see:

- “Parameterizing Property Values for Objects and Checkpoints” on page 207
- “Parameterizing Values for Operations” on page 208

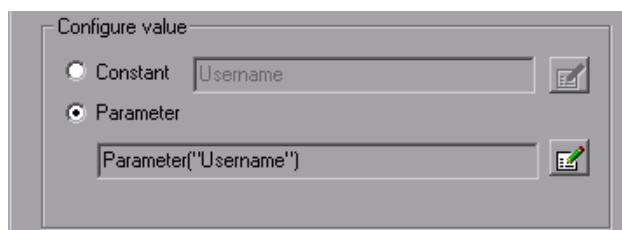
Tip: When you use the Step Generator to add new steps, you can parameterize the values for the operation you select. For more information, see “Inserting Steps Using the Step Generator” on page 467.

Parameterizing Property Values for Objects and Checkpoints

You can parameterize the values for one or more properties of an object in the Object Properties or Object Repository dialog box. You can parameterize the values for one or more properties of a checkpoint in the Checkpoint Properties dialog box.

To parameterize object or checkpoint property values:

- 1 Open the dialog box for the object properties or checkpoint properties in one of the following ways:
 - Choose **Step > Object Properties**, or right-click a step and choose **Object Properties**. The Object Properties dialog box opens.
 - Choose **Tools > Object Repository**, click the **Object Repository** toolbar button, or right-click the action or component containing the object and choose **Object Repository**. The Object Repository dialog box opens.
 - Choose **Step > Checkpoint Properties**, or right-click the checkpoint and choose **Checkpoint Properties**.
- 2 Select the property that you want to parameterize.
- 3 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 210.

4 Accept or change the displayed parameter definition:

- To accept the displayed parameter statement and close the dialog box, click **OK**.
- To accept the displayed parameter statement and parameterize another of the displayed values, select another property.
- 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.



For more information on defining value settings for specific parameter types, see:

- “Setting Test, Action, or Component Parameter Options” on page 213
- “Setting Data Table Parameter Options” on page 218
- “Choosing Global or Action Data Table Parameters” on page 220
- “Using Random Number Parameters” on page 233

Parameterizing Values for Operations

If the method 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 or component runs.

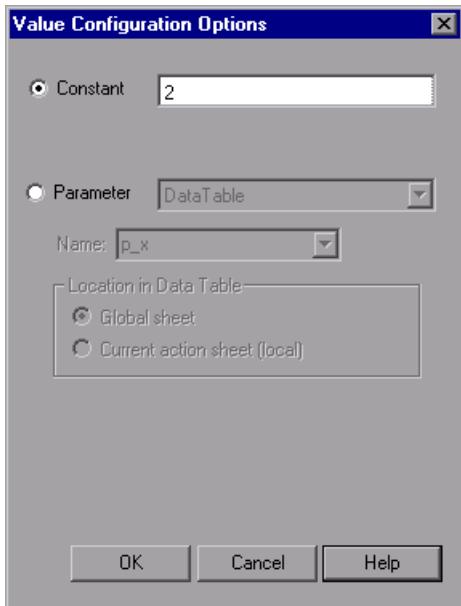
"Book a Flight: Mer...			
Loc "passFirst0"	Set "John"		Enter "John" in the "passFirst0" edit box.
Loc "passLast0"	Set "Brown"		Enter "Brown" in the "passLast0" edit box.
Loc "creditnumber"	Set	<RandomNumber>	Enter <the value of a generated random number> in the "creditnumber"

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.



- 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 210.
- 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, Action, or Component Parameter” on page 214
- “Defining the Settings for a Data Table Parameter” on page 219
- “Defining the Settings for an Environment Variable Parameter” on page 230
- “Defining Settings for a Random Number Parameter” on page 234

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 tab of the Test Settings dialog box

When parameterizing	Condition	Default parameter type	Default parameter name
 A value for a step or checkpoint in a component	At least one input parameter is defined for the component	Component parameter	The first input parameter displayed in the Parameters tab of the Business Component 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 for components are created in the Data Table sheet for each component.

 Data Table parameters for tests are created in the Global sheet.

For more information on Data Table sheets, see Chapter 18, “Working with Data Tables.”

Using Test, Action, and Component Input Parameters

You can parameterize a step using a test, action, or component input parameter, in order to use values that have been passed from the application that ran (called) your test or component. For example, you can use an input test parameter as the value for a method argument.

You can only parameterize a value using a test, action, or component parameter if the parameter has been defined for the calling test, action or component. For more information on defining parameters, see “Setting Action Parameters” on page 370 and “Setting Action Call Parameter Values” on page 379.

You can parameterize steps by selecting input component parameters in the Parameter Options or Value Configuration Options dialog box.

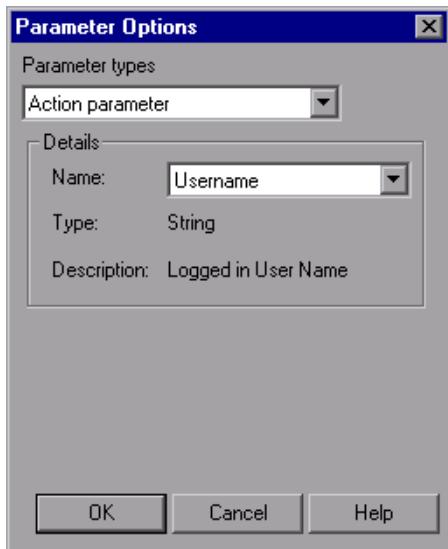
The parameter options that are available in these dialog boxes depend on where you are currently located in your test, and whether test, action, or component parameters are defined. In addition, the option to parameterize a value using a specific test, action, or component parameter is available only if the value types match. For more information, see “Using Action Parameters” on page 364 and “Defining Parameters for Your Test or Component” on page 682.

Alternatively, you can parameterize steps using the **Parameter** utility object, in the format: `Parameter("ParameterName")`. For more information, see “Using Action, or Component Parameters in Steps in the Expert View” on page 214.

Tip: You can also create test, action, or component 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 or component. For more information, see “Outputting a Value to an Action or Component Parameter” on page 262.

Setting Test, Action, or Component Parameter Options

When **Test parameter**, **Action parameter**, or **Component parameter** is selected as the parameter type, you can select the required parameter in the Parameter Options dialog box. The **Parameter** section of the Value Configuration Options dialog box is very similar to the Parameter Options dialog box.



Tip: When you open the Parameter Options dialog box, the default parameter type may be set to **Test parameter**, **Action parameter**, or **Component parameter**. For more information about default parameter type settings, see “Understanding Default Parameter Values” on page 210.

Defining the Settings for a Test, Action, or Component Parameter

The following options are available for configuring test, action, or component parameters:

Name—Specifies the name of the parameter.

Type—Displays the type defined for the parameter.

Description—Displays the description defined for the parameter.

You can also use test, action, or component parameter variables using parameterization objects and methods in the Expert View. For more information, refer to the *QuickTest Professional Object Model Reference*.

Using Action, or Component 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 in order 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

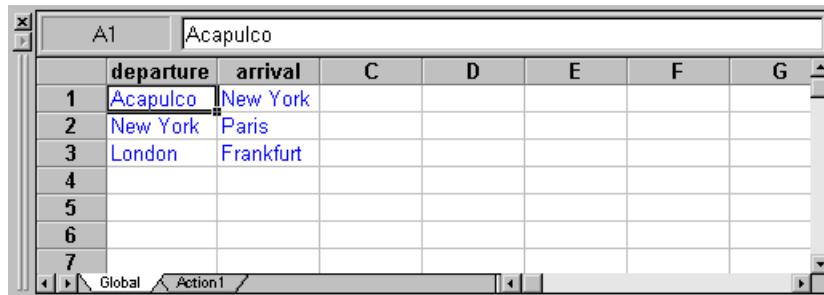
You can supply the list of possible values for a parameter by creating a Data Table parameter. Data Table parameters enable you to create a data-driven test, component, or action that runs several times using the data you supply. In each repetition, or *iteration*, QuickTest uses a different value from the Data Table.

For example, 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 recording the submission of numerous queries. This is a slow, laborious, and inefficient solution. By using Data Table parameters, you can run the test or component for multiple queries in succession.

When you parameterize your test or component, you first record steps that accesses the Web site and checks for the available flights for one requested itinerary.

You then substitute the recorded itinerary with a Data Table parameter and add your own sets of data in the global sheet of the Data Table, one for each itinerary.



A screenshot of a Data Table in the Global sheet of QuickTest. The table has columns labeled 'departure' and 'arrival'. Row 1 contains 'Acapulco' in the 'departure' column and 'New York' in the 'arrival' column. Row 2 contains 'New York' in the 'departure' column and 'Paris' in the 'arrival' column. Row 3 contains 'London' in the 'departure' column and 'Frankfurt' in the 'arrival' column. Rows 4 through 7 are empty. The table is titled 'A1' and has a header row with columns C, D, E, F, and G.

	departure	arrival	C	D	E	F	G
1	Acapulco	New York					
2	New York	Paris					
3	London	Frankfurt					
4							
5							
6							
7							

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.

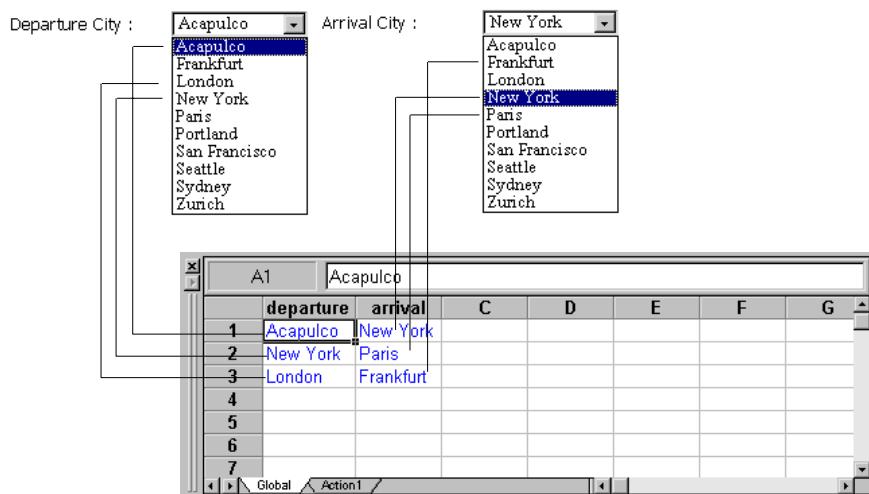
Each column in the table represents the list of values for a single Data Table parameter. The column header is the parameter name.

Each row in the table represents a set of values that QuickTest submits for all the parameters during a single iteration of the test or component. When you run your test or component, QuickTest runs one iteration of the test or component 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 18, “Working with Data Tables.”

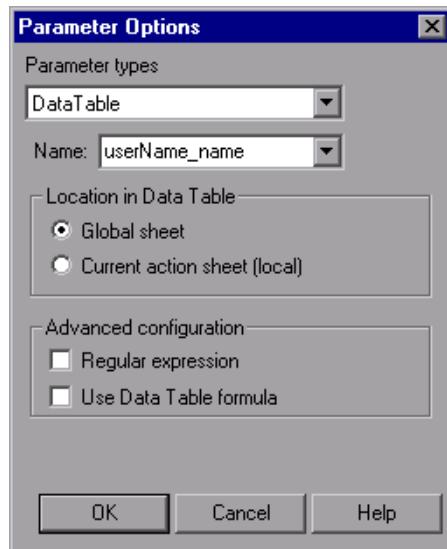
Tip: You can also create Data Table output values, which retrieve values during the run session and insert them into a column in the Data Table. You can then use these columns as Data Table parameters in your test or component. For more information, see Chapter 13, “Outputting Values.”

In the previous example, QuickTest submits a separate query for each itinerary when you run the test or component.



Setting Data Table Parameter Options

When **Data Table** is selected as the parameter type, the Parameter Options dialog box enables you to configure your parameter to use values from the Data Table. The **Parameter** section of the Value Configuration Options dialog box is very similar to the Parameter Options dialog box.



Tip: When you open the Parameter Options dialog box, **Data Table** may be set as the default parameter type. For more information about default parameter type settings, see “Understanding Default Parameter Values” on page 210.

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 about global and action parameters, see “Choosing Global or Action Data Table Parameters” on page 220. For more information about actions, see Chapter 17, “Working with Actions.”

If you are working in shared object repository mode, and you want to parameterize an object property value, the **Current action sheet (local)** option is disabled and you can store your parameter only in the global Data Table. For more information, see “Parameterizing Test Objects in Shared Object Repository Mode” on page 824.

Advanced configuration (if applicable):

- **Regular expression** (if applicable)—Sets the value of the parameter as a regular expression. For more information, see “Understanding and Using Regular Expressions” on page 290. 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 412.

Note: You can also define Data Table variables using parameterization objects and methods in the Expert View. For more information, refer to the *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* or an *action Data Table parameter*.

Note:  When working with components, there is only one type of Data Table sheet and one type of Data Table parameter.

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. 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 easily pass values from one action to another. For more information, see Chapter 13, “Outputting Values.”

For more information about setting global iteration preferences, see “Defining Run Settings for Your Test” on page 665.

Action 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 parameters in each iteration of the action. By default, actions run only one iteration.

You can also set 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 about setting action iteration preferences, see “Inserting Calls to Existing Actions” on page 357.

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 581.

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 18, “Working with Data Tables.”

There are three 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 tab 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 or Component” on page 685.

For information on creating or modifying environment variables in the Parameter Options dialog box, see “Setting Environment Variable Parameter Options” on page 229.

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 13, “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 224.
 - **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 components, and are designated as read-only. For more information, see “Using Built-in Environment Variables” on page 227.
-

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 “Using Environment Variables to Specify the Application Details for Your Test” on page 709.

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 tab of the Test Settings or Component 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 25, “Setting Options for Individual Tests or Components.”

Notes:

You can also store environment variable files in Quality Center. For more information, see “Using Environment Variable Files with Quality Center” on page 226.

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.

You can continue to use existing external environment variable files that were created for QuickTest 6.5 (in **.ini** format) in this version of QuickTest.

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** Choose **Test > Settings** to open the Test Settings dialog box. For more information about the Test Settings dialog box, see Chapter 25, “Setting Options for Individual Tests or Components.”
- 2** Click the **Environment** tab.
- 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 229.

Using Environment Variable Files with Quality Center

When working with Quality Center and environment variable files, you must save the environment variable file as an attachment in your Quality Center project *before* you specify the file in the Environment tab 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 the file system. Thus, once you save the file to the project, changes made to the Quality Center repository 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 224.
- 2** In Quality Center, add the file to the project as an attachment. For more information, refer to your Quality Center documentation.
- 3** In QuickTest, connect to the Quality Center project. For more information, see “Connecting to and Disconnecting from Quality Center” on page 946.
- 4** In the Test Settings dialog box, click the **Environment** tab.
- 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 40, “Working with Quality Center” and refer to your Quality Center documentation.

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 229.

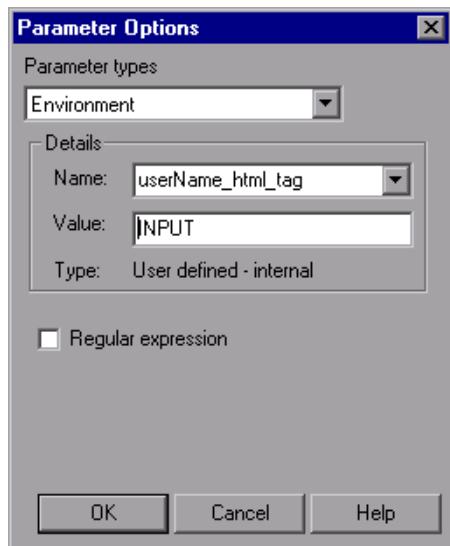
The following built-in environment variables are available:

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.
ResultDir	The path of the folder in which the current test results are located.
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 or Component" on page 519.

Name	Description
UpdatingCheckpoints	Indicates whether checkpoints are being updated during the update run process. For more information, see “Updating a Test or Component” on page 519.
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 or Component” on page 519.
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 select **Environment** as the parameter type, the Parameter Options dialog box enables you to configure your parameter to use values from the Environment variable list. The **Parameter** section of the Value Configuration Options dialog box is very similar to the Parameter Options dialog box.



Defining the Settings for an Environment Variable Parameter

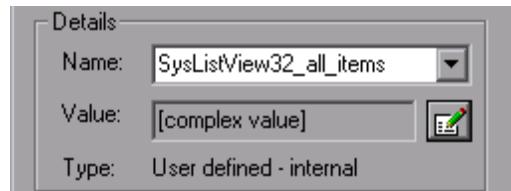
The following options are available for configuring environment variable parameters:

- **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.

Note: 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.

- **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 **Edit Complex Value** button. For more information, see “Viewing and Editing Complex Parameter Values” on page 232.



- **Type**—Specifies the type of environment variable parameter:
 - 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.

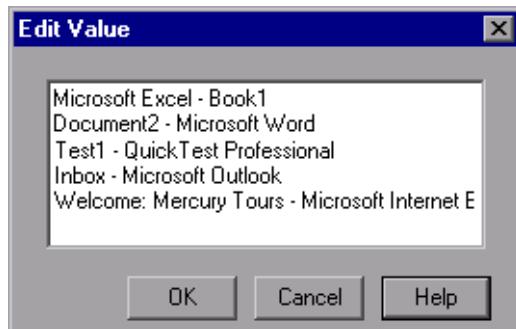
Regular expression—Sets the value of the parameter as a regular expression. This option is available only when parameterizing a checkpoint or object property text string value, and the selected environment variable parameter type is **internal user-defined**. For more information on regular expressions, see “Understanding and Using Regular Expressions” on page 290.

Note: You can also define environment variables using parameterization objects and methods in the Expert View. For more information, refer to the *QuickTest Professional Object Model Reference*.

Viewing and Editing Complex Parameter Values



When you click the **Edit Complex Value** button for a parameter with a value that cannot be displayed entirely in the **Value** box, the Edit Value dialog box displays the full contents of the value.

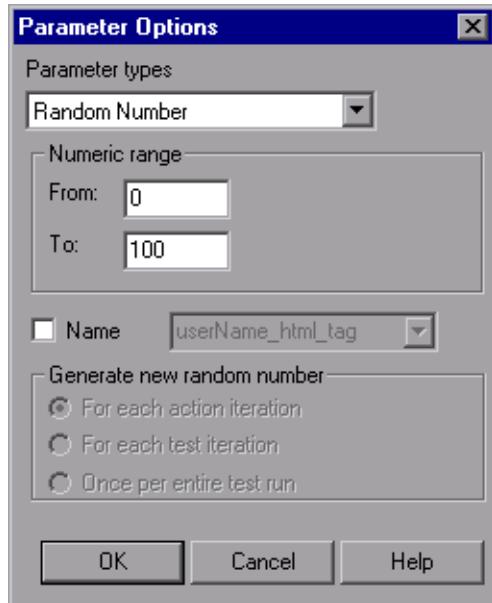


You can edit the value for an internal user-defined environment variable parameter.

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 select **Random Number** as the parameter type, the Parameter Options dialog box enables you to configure your parameter to use random numbers. The **Parameter** section of the Value Configuration Options dialog box is very similar to the Parameter Options dialog box.



Defining Settings for a Random Number Parameter

Note: Random number parameters are not appropriate for non-numeric values, such as text or hypertext links.

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:

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, refer to the *QuickTest Professional Object Model Reference*.

Example of a Parameterized Test

The following example shows how to parameterize a step object, step method, and a checkpoint using Data Table parameters.

When you test your application or Web site, 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.

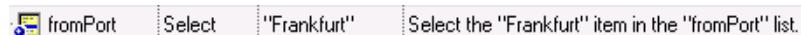
Suppose that you want to check that the flights are booked correctly for a variety of different locations. Rather than create a separate test with a separate checkpoint for each location, you can parameterize the location information. For each iteration of the test, QuickTest then checks the flight information 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.

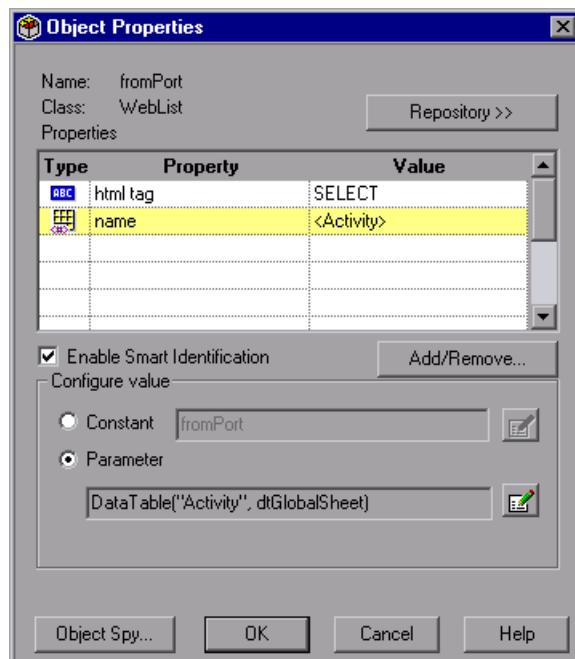
Item	Operation	Value	Documentation
Action1			
Welcome: Mercury Tours			
Welcome: Mercury Tours			
userNmae	Set	"mercury"	Enter "mercury" in the "userName" edit box.
password	SetSecure	"404ddfac4dc9b8230742e447b1b7...	Enter the encrypted string "404ddfac4dc9b8230742e447b1b7..."
Sign-In	Click	2,2	Click the "Sign-In" image.
Find a Flight: Mercury			
fromPort	Select	"Frankfurt"	Select the "Frankfurt" item in the "fromPort" list.
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	Sync		Wait for the Web page to synchronize before continuing the run.
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.

Parameterize a Step

Parameterize the object and the method of the following step:

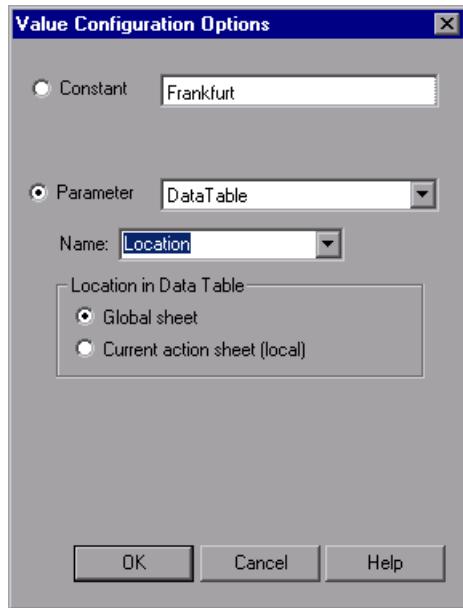


In the Object Properties dialog box, select the **name** property. Select the **Parameter** radio button and click the **Parameter Options** button. In the Parameter Options dialog box, rename fromPort to Activity. Click **OK** to close the Parameter Options dialog box.



Click **OK** to close the Object Properties dialog box. The Activity column is added to the Data Table.

Now parameterize the method of the **fromPort** step. In the Keyword View, click in the **Value** column 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 205.

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:

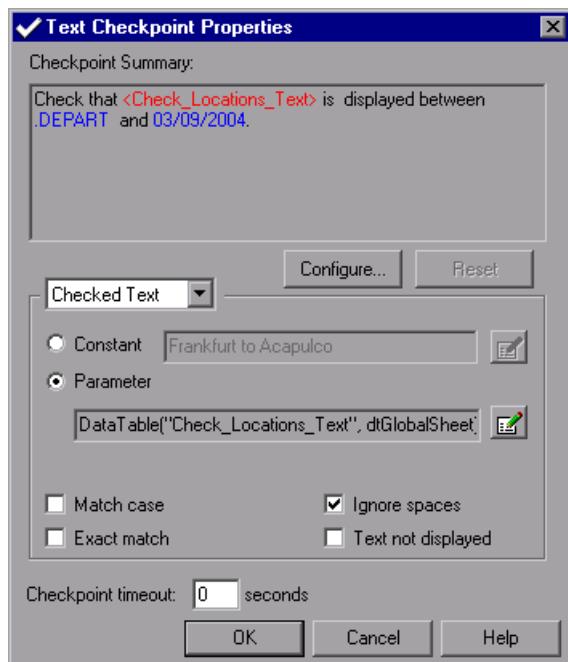
The screenshot shows a flight booking interface. At the top left is a yellow bar with the text "SELECT FLIGHT". To its right is a blue icon of an airplane. Below this is a section titled "DEPART" with the subtitle "Frankfurt to Acapulco" and the date "03/09/2004". A table lists five flight options:

SELECT	FLIGHT	DEPART	STOPS
<input checked="" type="radio"/>	Blue Skies Airlines 210 Price: \$672 (based on round trip)	5:03	non-stop
<input type="radio"/>	Blue Skies Airlines 211 Price: \$685 (based on round trip)	7:09	non-stop
<input type="radio"/>	Pangea Airlines 212 Price: \$712 (based on round trip)	9:15	non-stop
<input type="radio"/>	Unified Airlines 213 Price: \$737 (based on round trip)	11:21	non-stop

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 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 205.

Enter Data in the Data Table

Complete the Data Table. The Data Table may be displayed as follows:

	Activity	Location	Check_Locations_Text	D	E
1	fromPort	Frankfurt	Frankfurt to Acapulco		
2	fromPort	Acapulco	Acapulco to Acapulco		
3	toPort	Acapulco	Acapulco to Acapulco		
4	toPort	Frankfurt	Acapulco to Frankfurt		
5					
6					
7					
8					
9					
10					
11					

For more information on Data Tables, see Chapter 18, “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...	"404ee5e998b25bcd6f116b031452..."	Enter the encrypted string "404ee5e998b25bcd6f116b031452..."
└─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 column>
└─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" window matches the parameterized text.
└─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.)

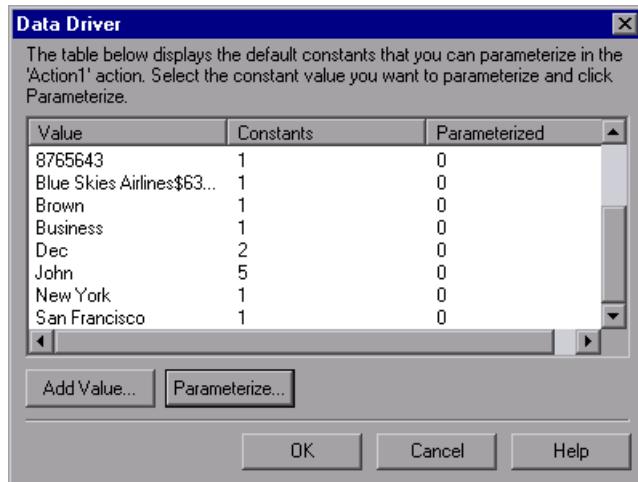
You cannot use the Data Driver to parameterize the values of arguments for user-defined methods or VBScript functions.

To parameterize a value using the Data Driver:

- 1** Display the action you want to parameterize.
- 2** Choose **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.



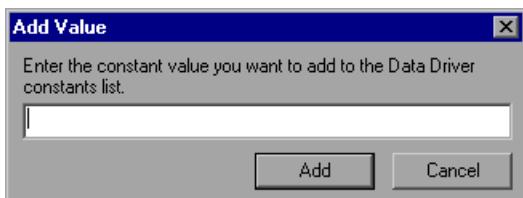
The Data Driver displays the Constants list for the action. For each constant value, it displays the number of times the constant value appears in the action.

By default, the list displays the constants for one or more of the arguments of the following methods: **Activate**, **Collapse**, **Deselect**, **Expand**, **ExtendSelect**, **GetTextLocation**, **Press**, **Select**, **SelectColumn**, **SelectRange**, **SelectRow**, **Set**, **SetCellData**, **SetSecure**, **SetText**, **Type**, and **WaitProperty**.

For more information on how to work with testing methods, see Chapter 36, "Working with the Expert View." For syntax and method information, refer to the *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 now add the constant values that you want to parameterize, as described in the following step.

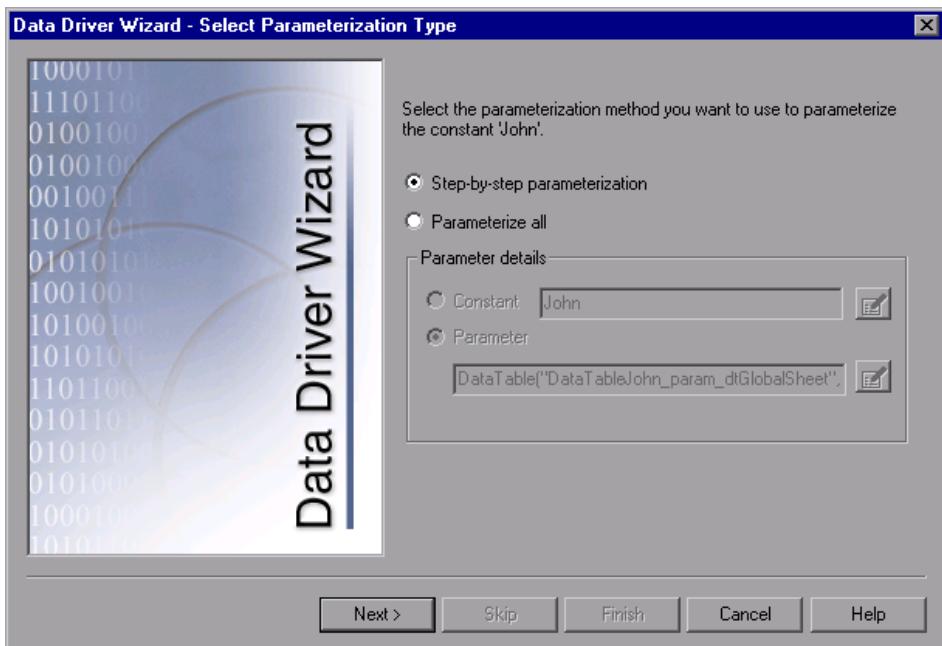
- 3 If you want to parameterize a value that is not currently displayed in the list (such as an object property value), click **Add Value**. The Add Value dialog box opens.



Enter a constant value in the dialog box and click **OK**. The constant is added to the list.

Note: You can add only constant values that currently exist in the test.

- 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:

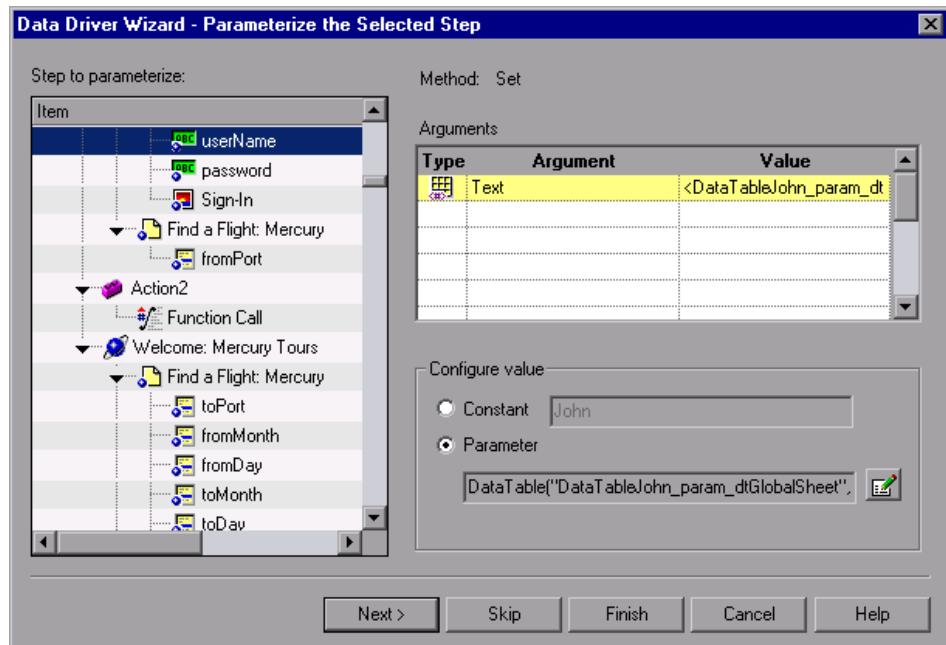
- **Step-by-step parameterization**—Enables you to view the current values of each step containing the selected value. For each step, you can choose whether or not to parameterize the value and if so, which parameterization options you want to use.
- **Parameterize all**—Enables you to parameterize all 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 **Parameter details** 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 205.

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 210.

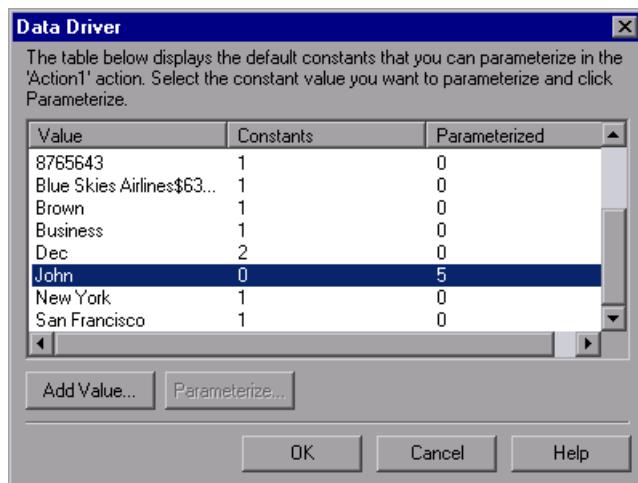
 Accept the default parameterization settings or click the **Parameter Options** button to set the parameterization options you want to apply to this step. For more information, see “Parameterizing Values in Steps and Checkpoints” on page 205.

- 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 - 9.
- 11** When you are finished parameterizing constants, click **OK**. The parameterization options you selected are applied to your action.

13

Outputting Values

QuickTest enables you to retrieve values in your test or component and to store them as output values. You can subsequently retrieve these values and use them as input at a different stage in the run session.

This chapter describes:

- ▶ About Outputting Values
- ▶ Creating Output Values
- ▶ Outputting Property Values
- ▶ Specifying the Output Type and Settings
- ▶ Outputting Text Values
- ▶ Outputting Database Values
- ▶ Outputting XML Values

About Outputting Values

An *output value* is a step in which one or more values are captured at a specific point in your test or 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.

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 578.

Creating Output Values

When you add an output value step to your test or component, you first select the category of values to output, for example, property values, text values, or XML element values. You can then determine which values to output and the storage location for each value.

For more information on storage location options, see “Storing Output Values” on page 252.

You can create the following categories of output values:

- Standard output values
- Text and text area 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.

You can also use standard output values to output the contents of table cells.

Tip: You can use standard output values to output text strings by specifying the **text** property of the object as an output value. This is the preferred way of outputting the text displayed in many Windows applications.

For more information on standard output values, see “Outputting Property Values” on page 255.

Text and Text Area Output Values

You can use text output values to output text strings displayed in a screen or Web page. 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 application.

For example, suppose that you want to store the text of any error message that appears after a specific step in the 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 268. For more information on text area output values, see “Creating Text Area Output Values” on page 270.

Considerations for Using a Text Output Value for Windows-Based Applications

The text-recognition mechanism used when you create a text or text area output value on a Windows-based application may occasionally retrieve unwanted text information (such as hidden text and shadowed text, which appears as multiple copies of the same string).

Additionally, text (and text area) output values may behave differently in different run sessions depending on the operating system version you are using, service packs you have installed, other installed toolkits, the APIs used in your application, and so on.

Therefore, when possible, it is highly recommended to retrieve text from your application window by using a standard output value step to output the value of the object's **text** (or similar) property.

Note that the above issues do not apply when working with Web-based applications.

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 on database output values, see “Outputting Database Values” on page 277.

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 further information, see Chapter 23, “Analyzing Test Results.”

For example, assume 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 280.

Output Value Categories and Environments

The following table shows the categories of output values that are supported by each environment.

Output Value Category	Web	Standard Windows	VB	ActiveX	Other Environment
Standard	S	S	S	S	NA
Page (Standard)	S	NA	NA	NA	NA
Table (Standard)	S	NA	NA	S	NA
Text	S	S	S	S	NA
Text area	NS	S	S	S	NA
Database	NS	NA	NA	NA	S (DbTable)
XML	S	NA	NA	NA	XML file

S—Supported

NS—Not Supported

NA—Not Applicable

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, action or component parameter
 - the run-time Data Table
 - an environment variable
-

Note: Output values are stored only for the duration of the test or component, and are not saved with the test or component. 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, Action or Component Parameters

You can output a value to an action or component 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 or component.

For example, suppose you are testing a shopping application that calculates your purchases and automatically debits your account with the amount that you have purchased. You want to test that the application correctly debits the purchase amount from the account each time that the action or component is run with a different list of items to purchase. You could output the total amount spent to an action or component 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 364.

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.

Note:  Although you can output values to the run-time Data Table for components, you can use only the first row of the Data Table, because component iterations are defined by the business process test in Quality Center and not by Data Table rows. For more information, refer to the *Business Process Testing User's Guide*.

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 or **component**, 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 in the Keyword View or Expert View 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 258
- “Defining Text and Text Area Output Values” on page 272
- “Defining Database and Table Cell Output Values” on page 278
- “Defining XML Output Values” on page 282

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 or component.

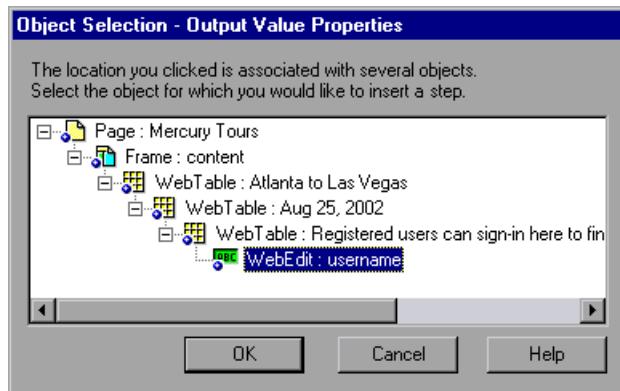
To create standard output values while recording:



- 1 In the Keyword View or Expert View, choose **Insert > Output Value**. Alternatively, you can click the arrow beside the **Insert Checkpoint** button on the toolbar.
- 2 Select **Standard Output Value**. The mouse pointer turns into a pointing hand.

Tip: Hold the CTRL key to change the window focus or perform operations such as a right-click or mouseover to display a context menu. Note that pressing the CTRL key does not enable you to select an application from the Windows taskbar, therefore you must make sure that the window you want to access is not minimized.

- 3 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.



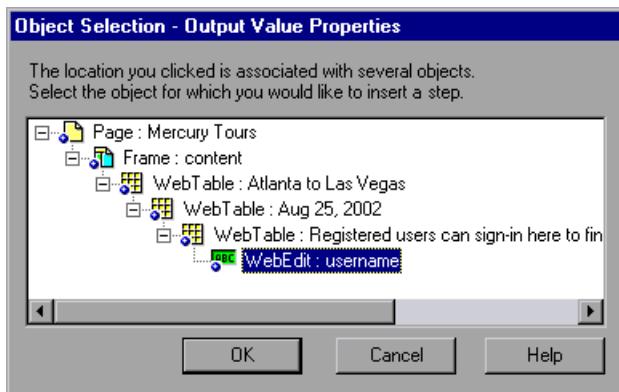
- 4 In the Object Selection dialog box, select the object for which you want to specify an output value, and click **OK**. The Output Value Properties dialog box opens for the selected object. If you select a **Table** item, the Table Output Value Properties dialog box opens.
 5 Specify the property values to output and their settings. For more information, see “Defining Standard Output Values” on page 258. If you selected a **Table** item, see “Defining Database and Table Cell Output Values” on page 278.
- 6 When you have finished defining the output value details, click **OK**. QuickTest inserts an output value step in your test or component.

To create standard output values while editing your test or component:

- 1 Make sure the **Active Screen** button is selected.
- 2 In the Keyword View or Expert View, 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 618.

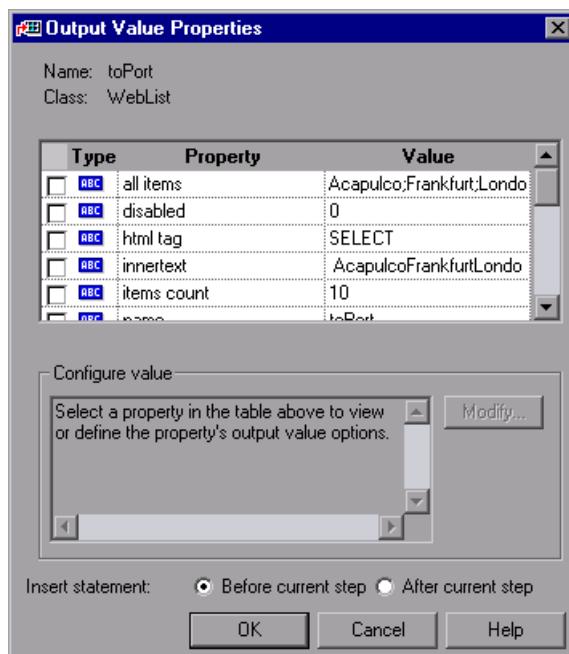
- 3 In the Active Screen, right-click the object for which you want to specify an output value and choose **Insert Output Value**. Alternatively, you can right-click the step in your test and choose **Insert Output Value**.
- 4 If the location you clicked is associated with more than one object, the 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 258. If you selected a **Table** item, see “Defining Database and Table Cell Output Values” on page 278.
- 7 When you have finished defining the output value details, click **OK**. QuickTest inserts an output value step in your test or component.

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 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 the specified property values.

Identifying the Object

The top part of the dialog box displays information about the test object for which you are creating an output value:

Item	Description
Name	The name of the test object.
Class	The type of object. In this example, the WebList class indicates it is a list object in a Web application.

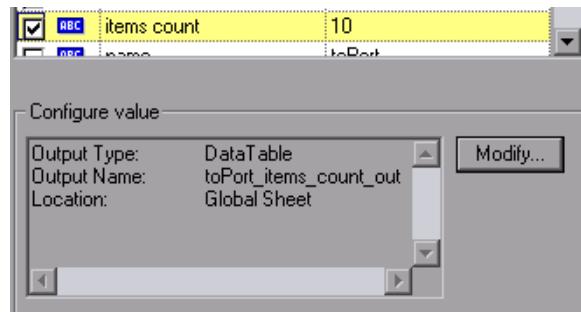
Selecting the Property Values to Output

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, action, or component 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,” below.

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 261.

When you select a property value to output, you can:

- accept the displayed output definition by selecting another property value 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 261.

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 or component. For more information, see “Selecting the Location for the Output Value Step” on page 268.

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,” below.

You can change the current output definition for the selected value by selecting a different output type and/or changing the output settings. For more information, see:

- “Outputting a Value to an Action or Component Parameter” on page 262
- “Outputting a Value to the Data Table” on page 264
- “Outputting a Value to an Environment Variable” on page 266

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 an action (for a test):

- if at least one output parameter is defined in the action, the default output type is **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 “Using Action Parameters” on page 364.

For more information on Data Table sheets, see Chapter 18, “Working with Data Tables.”



When you output a value for a step in a component:

- if at least one output parameter is defined in the component, the default output type is **Component parameter** and the default output name is the first output parameter displayed in the Component Properties dialog box.
- if no output parameters are defined in the component, 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 Data Table sheet for the component.

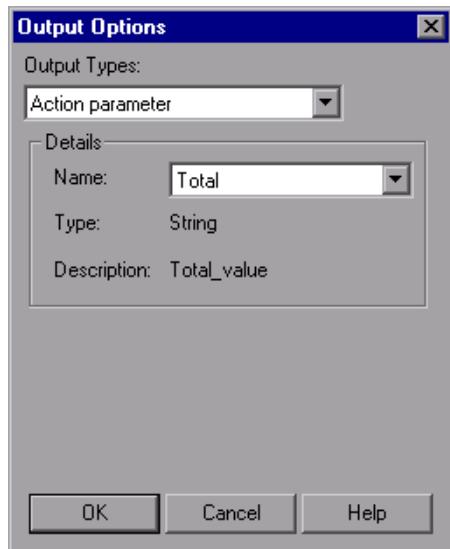
For more information on defining output parameters for components, see “Defining Parameters for Your Test or Component” on page 682.

For more information on Data Table sheets, see Chapter 18, “Working with Data Tables.”

Outputting a Value to an Action or Component Parameter

You can output a value to an action or component 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 or component. You can only output a value to an action or component parameter if the parameter has been defined as an output parameter for the calling action or component. In addition, the option to output a value to an action or component is available only if the output value type and the parameter value type match.

When **Test parameter**, **Action parameter**, or **Component 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.

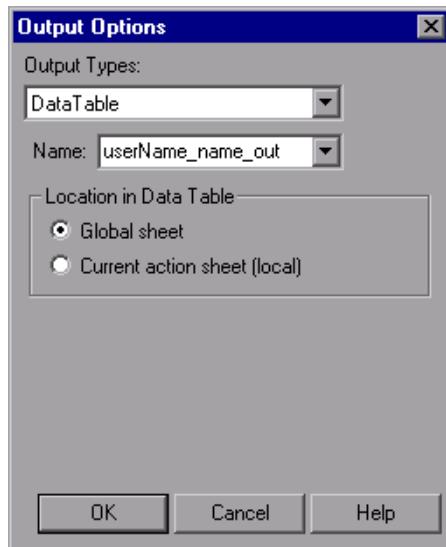


Tip: When you open the Output Options dialog box, QuickTest may display **Action parameter** or **Component parameter** as the default output type. For more information, see “Understanding Default Output Definitions” on page 261.

- **Name**—Specifies the name of the parameter in which to store the output value. The list contains the names of the currently defined output parameters for the action or component.
- **Type**—Displays the type of parameter, for example, String.
- **Description**—Displays the description defined for the parameter.

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.



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 261.

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 345. For more information on actions, see Chapter 17, “Working with Actions.”

Note:  This option is not available when outputting values for a component.

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.



- **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 222.

Selecting the Location for the Output Value Step

When you create output values while editing a test or component, 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 a screen. 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 also output text values from defined text areas. For more information, see “Creating Text Area Output Values” on page 270.

Note: When you create a text or text area output value, the text-recognition mechanism may not always retrieve the expected text. For this reason, when possible, you should retrieve text from your application window by using a standard output value for the object containing the text you require, using its **text** (or similar) property. For more information on standard output values, see “Outputting Property Values” on page 255. When an object does not have a text-type property, a text area output value must be used.

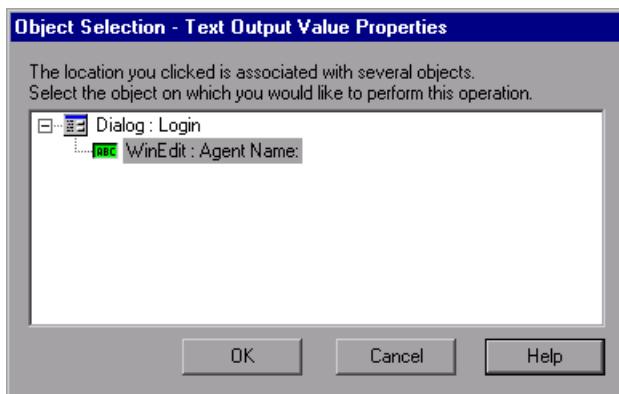
You can create a text output value while recording or editing your test or component.

To create a text output value while recording:

- 1** Highlight or display the text string you want to use for an output value.
- 2** Choose **Insert > Output Value > Text Output Value**. The mouse pointer turns into a pointing hand.

Tip: Hold the CTRL key to change the window focus or perform operations such as a right-click or mouseover to display a context menu. Note that pressing the CTRL key does not enable you to select an application from the Windows taskbar, therefore you must make sure that the window you want to access is not minimized.

- 3** In your application, click the text string for which you want to specify a text output value. If the location you clicked is associated with more than one object, the Object Selection – Text Output Value Properties dialog box opens.



- 4** In the Object Selection dialog box, select the object for which you want to specify a text output value, and click **OK**.
- 5** The Text Output Value Properties dialog box opens.
- 6** Specify the settings for the output value. For more information, see "Defining Text and Text Area Output Values" on page 272.
- 7** When you have finished defining the text output value details, click **OK**. QuickTest inserts an output value step in your test or component.

To create a text output value when editing your test or component:



- 1** Make sure the **Active Screen** button is selected.
- 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 choose **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 272.
- 6** When you have finished defining the output value details, click **OK**. QuickTest inserts an output value step in your test or component.

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 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—Standard Windows, Visual Basic and ActiveX.

Tip: 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.

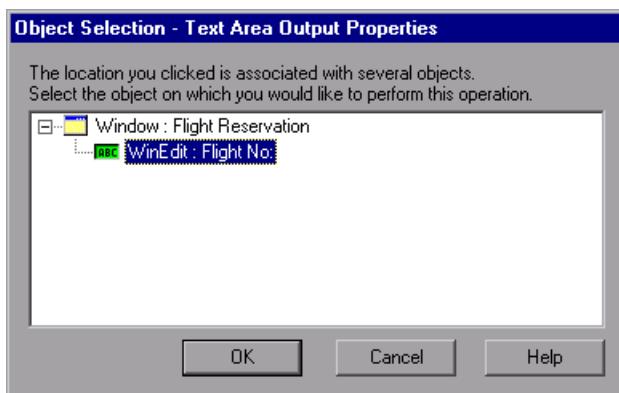
Because text may change its position during test runs, you must make sure that the area defined is large enough so that the output text is always within its boundaries.

To create a text area output value:

-  **1** While recording, choose **Insert > Output Value > Text Area Output Value**. The QuickTest window is minimized 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.

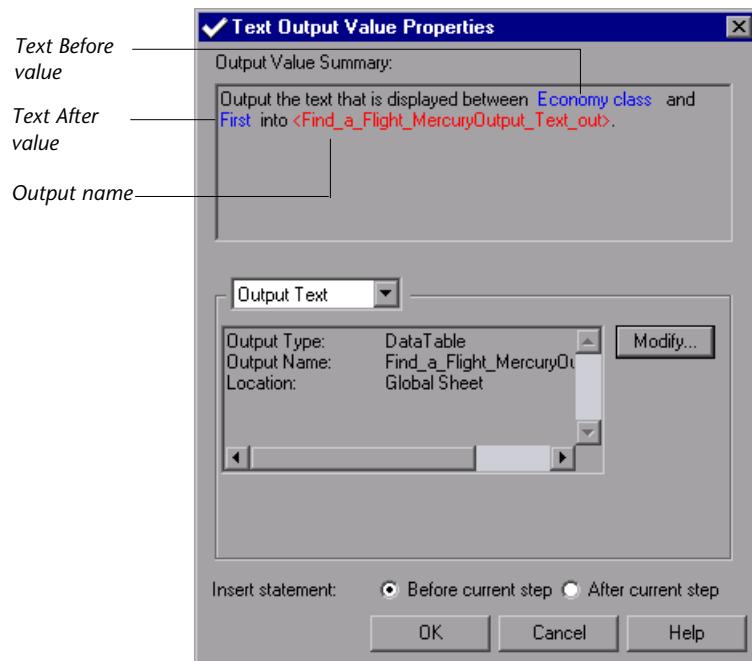


- 3** Select the object for which you are creating the output value. The Text Area Output Value Properties dialog box opens.
- 4** Specify the settings for the output value. For more information, see "Defining Text and Text Area Output Values" on page 272.
- 5** When you have finished defining the output value details, click **OK**. QuickTest inserts an output value step in your test or component.

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 test runs.

The Text Output Value Properties and Text Area Output Value Properties dialog boxes enable you to define the output value settings for the selected text string, and also to define the options for the text displayed before and after the output value.



Note: The Text Output Value Properties and Text Area Output Value Properties dialog boxes are identical. However, when you create a text area output value, the **Text Before** and **Text After** values are not captured.

In the example shown above, the output value is the text displayed between **Economy class** (the **Text Before** value) and **First** (the **Text After** value).

When you create a text or text area output value, you can specify the **Text Before** and **Text After** values, and 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.

Identifying the Text for the Output Value

The **Output Value Summary** pane at the top of the dialog box describes the text string for the output value. For a text output value, this is the string displayed between the **Text Before** value and the **Text After** value. For a text area output value, the output value string contains all the text in the selected area.

This pane also shows the output name assigned to the text string.

Specifying the Captured Text as an Output Value

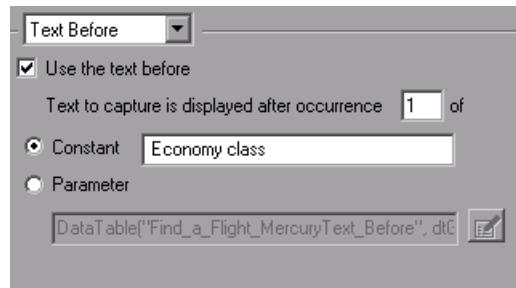
By default, **Output Text** is selected in the list box in the center of the dialog box and the area below the list box displays the current output value settings for the selected text.

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 261.

You can accept the displayed output definition or click **Modify** to specify the output settings for the selected text. For more information, see “Specifying the Output Type and Settings” on page 261.

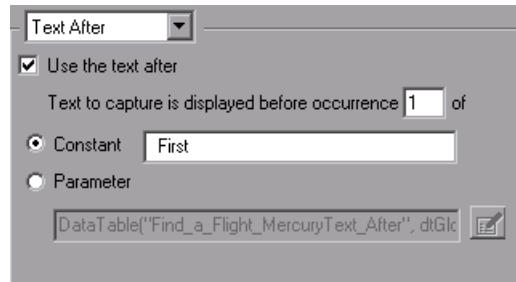
Specifying Options for the Text Before/Text After Values

When you choose **Text Before** from the list box, you can define the options for the text displayed before the output value string.



Note: If you clear the **Use the text before** check box, the options below it are not available. During the run session, QuickTest retrieves the value of the first occurrence of the defined output string, regardless of the text displayed before it.

When you choose **Text After** from the list box, you can define the options for the text displayed after the output value string.



Note: If you clear the **Use the text after** check box, the options below it are not available. During the run session, QuickTest retrieves the value of the first occurrence of the defined output string, regardless of the text displayed after it.

When the **Use the text before** check box is selected, the current **Text Before** value is displayed in the **Constant** box. When the **Use the text after** check box is selected, the current **Text After** value is displayed in the **Constant** box.

You can use the following options to define these values:

- **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 it 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. In the example above, the selected output string is displayed before the first occurrence of the string First. 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, 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.

- **Constant**—Sets the **Text Before** or **Text After** value as a constant. A constant is a value that is defined directly within the test or component and remains set for the duration of the test or component.

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 insure 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 287.

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 or component. For more information, see “Selecting the Location for the Output Value Step” on page 268.

Outputting Database Values

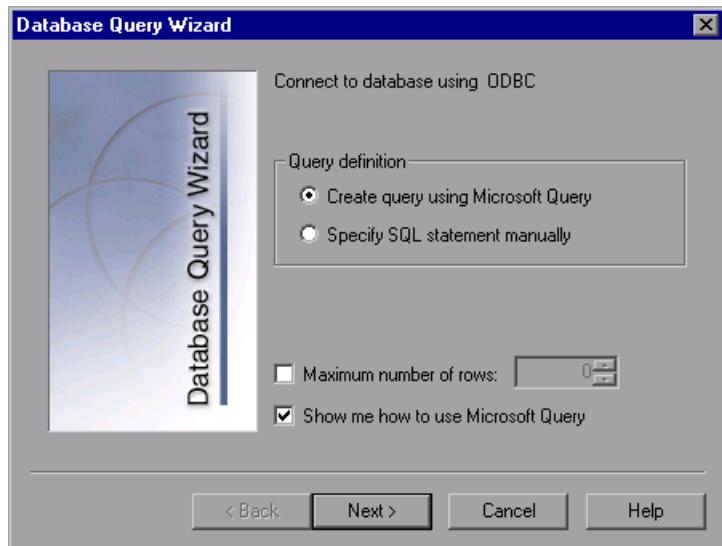
You can create database output values by defining a query to retrieve data from the database and selecting the values you want to output from the query result set. You can then specify the output settings for the selected values. During the run session, QuickTest captures the current data from the database and outputs the values according to the specified settings.

You can create database output values while recording or editing your test or component.

To create database output values:



- 1 Choose **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 135.

When you have finished defining your query, the Database Output Value Properties dialog box opens.

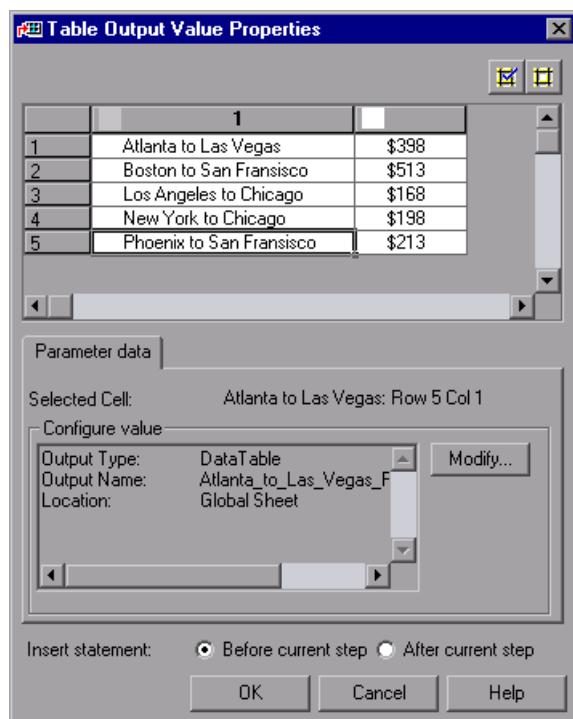
- 3 Specify the values to output and their settings. For more information, see "Defining Database and Table Cell Output Values" on page 278.

- 4 When you have finished defining the output value details, click **OK**. QuickTest inserts an output value step in your test or component.

Defining Database and Table Cell Output Values

The Database Output Value Properties dialog box enables you to select the database cells for the values you want to output. The Table Output Value Properties dialog box enables you to select the table cells for the values you want to output. You can define the output settings for each value that you select.

The options in the two dialog boxes are identical.



Choosing Cells for Output Values

The top part of the dialog box displays a grid representing the cells in the captured table or the 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 an value to output, double-click it 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 deselect a cell for an output value, double-click it 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 or 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 261.

When you select a value in a table or 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 261.

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 or component. For more information, see “Selecting the Location for the Output Value Step” on page 268.

Outputting XML Values

You can create XML output values from any XML document contained in an XML Web page or frame, or directly from an XML file. You can output element and/or attribute values in an XML output value step.

You can insert XML Web page or frame output values while you are recording. You can create XML output values directly from an XML file while recording or editing your test or component.

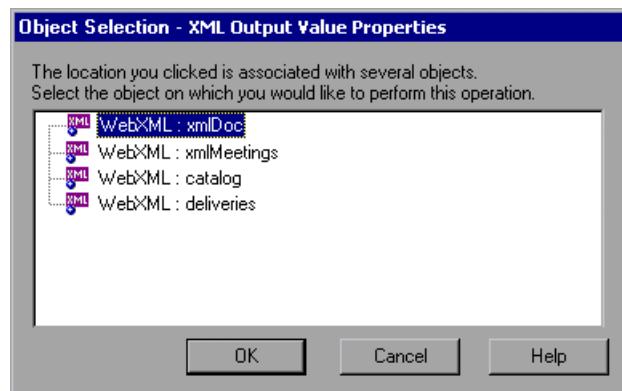
To create XML output values from an XML Web page or frame:



- 1 While recording, choose **Insert > Output Value > XML Output Value**. The mouse pointer turns into a pointing hand.

Tip: Hold the CTRL key to change the window focus or perform operations such as a right-click or mouseover to display a context menu. Note that pressing the CTRL key does not enable you to select an application from the Windows taskbar, therefore you must make sure that the window you want to access is not minimized.

- 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.

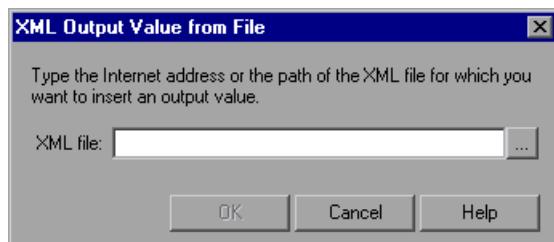


- 3 Select the XML item you want to specify for the output value step.

- 4 Click **OK**. The XML Output Value Properties dialog box opens.
- 5 Specify the values to output and their settings. For more information, see “Defining XML Output Values” on page 282.
- 6 When you have finished defining the output value details, click **OK**. QuickTest inserts an output value step in your test or component.

To create XML output values from an XML file:

- 1 Choose **Insert > Output Value > XML Output Value (File)**. The XML Output Value from File dialog box opens.



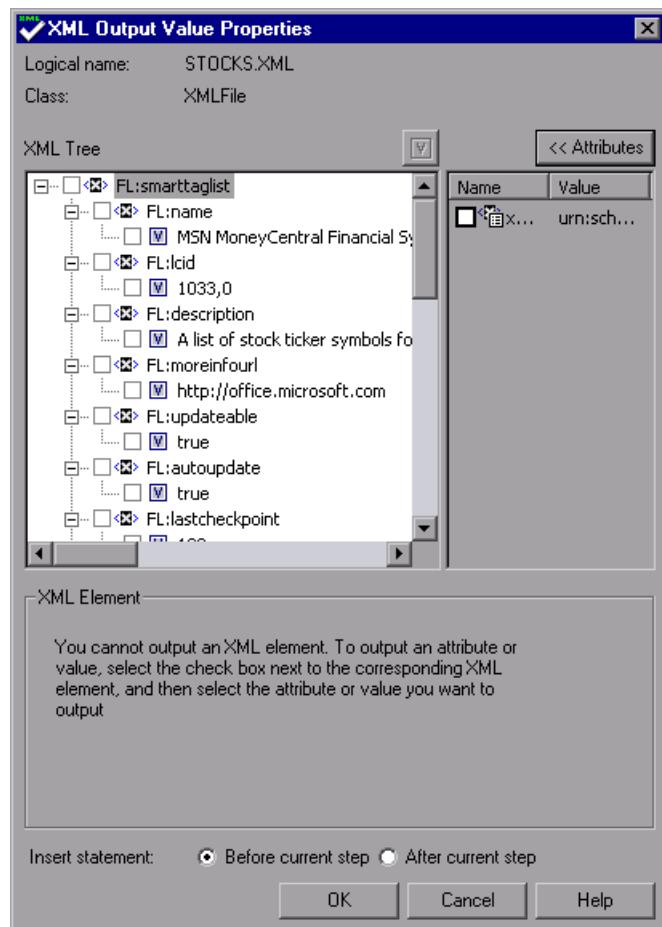
- 2 In the **XML File** box, enter the Internet address or the file path of the XML file. Alternatively, click the browse button to navigate to the XML file for which you want to create output values. You can specify an XML file either from your file system or from Quality Center.

Note: You can enter a relative path and QuickTest will search for the XML file in the folders listed in the Folders tab 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 Chapter 24, “Setting Folder Testing Options.”

- 3 Click **OK**. The XML Output Value Properties dialog box opens.
- 4 Specify the values to output and their settings. For more information, see “Defining XML Output Values” on page 282.
- 5 When you have finished defining the output value details, click **OK**. QuickTest inserts an output value step in your test or component.

Defining XML Output Values

The XML Output Value 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.



Note: QuickTest loads large XML files in the background. While the file is loading, an indication of this is shown in the XML Output Value Properties dialog box, and you can select only XML nodes that are already loaded.

Identifying the XML File or Object

The top part of the XML Output Value Properties dialog box displays information about the XML file or object:

Item	Description
Name	The name of the XML file or object.
Class	The type of test object on which you are creating an output value. This is either XMLFile (for files) or WebXML (for Web pages or frames).

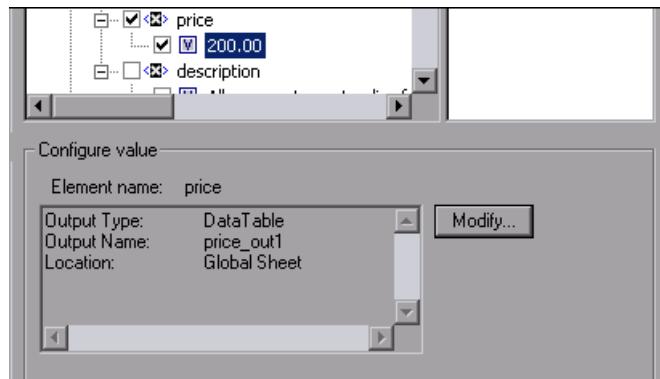
Choosing the Element Values and/or Attributes to Output

The XML Tree pane displays the hierarchy of the XML document, enabling you to select the element and/or attribute values that you want to output. This pane contains the following items:

Option	Description
XML Tree	The XML Tree displays the hierarchical relationship between each element and value in the XML document. Each element is displayed with a icon. Each value is displayed with a icon. Select the check box of an element value to output its value.
Attributes Pane	When you click the Attributes>> button, the attributes associated with the selected element from the tree are displayed in the Attributes pane to the right of the XML Tree. The attributes pane lists the name and value of each attribute. Select the check box of an attribute to output its value.

Specifying the Settings for the Output Value

When you select an element value or attribute, the **Configure value** area displays its current output settings.



When an element or attribute 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 261.

When you select an XML element or attribute value for output, you can:

- accept the displayed definition by selecting another element or attribute 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 for the output type displayed in the **Configure value** area. For more information, see “Specifying the Output Type and Settings” on page 261.

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 or component. For more information, see “Selecting the Location for the Output Value Step” on page 268.

14

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 and components.

This chapter describes:

- About Configuring Values
- Configuring Constant and Parameter Values
- Understanding and Using Regular Expressions
- Defining Regular Expressions

About Configuring Values

A number of dialog boxes, such as the Object Repository, Object Properties, and 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 and Step Generator, 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 or component.
- **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 test and components, see Chapter 12, “Parameterizing Values.”

You can edit a constant value in the **Configure value** area. In certain contexts, you can define a constant value, a Data Table parameter value, or an environment parameter value as 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 titlebar changes according to a file name, you can use a regular expression to identify a window whose titlebar has the specified product name, followed by a hyphen, and then any other text.

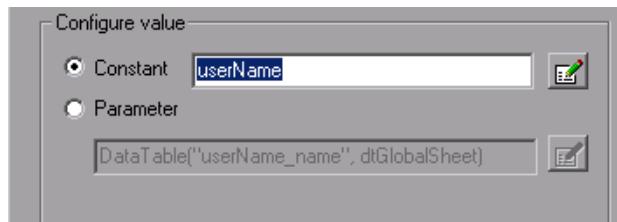
Configuring Constant and Parameter Values

You can define a value as a constant or a parameter:

- in the Value Configuration Options dialog box, by clicking the parameterization button  for a selected value, for example, in the Keyword View or Step Generator. For more information, see “Configuring a Selected Value” on page 289.
- in the **Configure value** area of a dialog box, by selecting a property or argument, for example, in the Checkpoint Properties dialog box.

Setting Values in the Configure Value Area

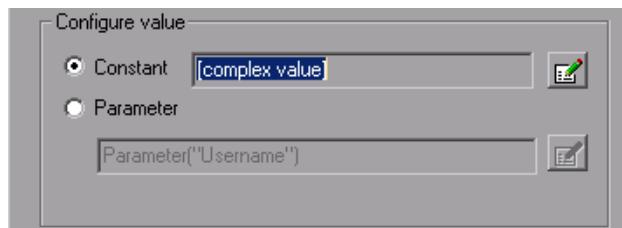
When you select an item in a dialog box containing a **Configure value** area, such as the Object Repository or Checkpoint Properties dialog box, you can select **Constant** or **Parameter** to set the value. The default is **Constant**.





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 290.

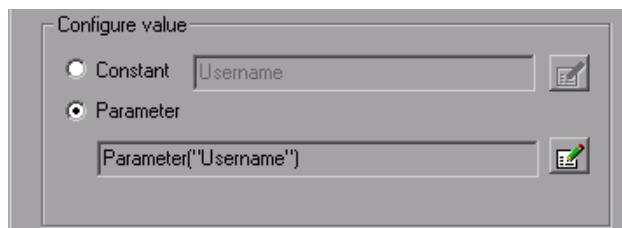
If the entire value cannot be displayed in the **Constant** box, it is shown as **[complex value]**. For example, the value of a list’s **all items** property is a multiline value, where each line contains the value of an item in the list.



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 288.

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 210.



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, Action, or Component Parameter Options” on page 213
- “Setting Data Table Parameter Options” on page 218
- “Setting Environment Variable Parameter Options” on page 229
- “Using Random Number Parameters” on page 233

For more information on using parameters in your test and components, see Chapter 12, “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 can not be displayed entirely in the **Constant** box), the Constant Value Options dialog box expands to show the entire contents of the value.

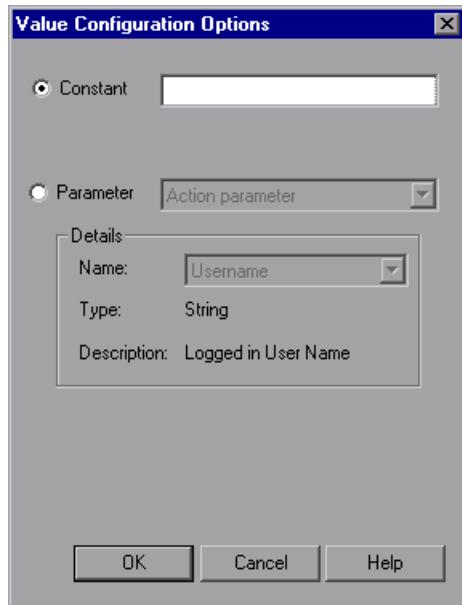


You can update the following options to edit the value of the constant:

- **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 290.
 - For information on defining a regular expression, see “Defining Regular Expressions” on page 293.

Configuring a Selected Value

When you click the parameterization button  for a selected value, the Value Configuration Options dialog box opens.



You can select one of the following options:

- **Constant**—defines a value that remains unchanged for the duration of the test or component. You can edit the value directly in the **Constant** box.
- **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 210.

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.

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, Action, or Component Parameter” on page 214
- “Defining the Settings for a Data Table Parameter” on page 219
- “Defining the Settings for an Environment Variable Parameter” on page 230
- “Defining Settings for a Random Number Parameter” on page 234

For more information on using parameters in your test and components, see Chapter 12, “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 when:

- defining the property values of an object in dialog boxes or in programmatic descriptions
- parameterizing a step
- creating checkpoints with varying values

For example, you can use a regular expression if you want to create a text checkpoint on a date text string, but the displayed date changes according to the current date. If you define the date as a regular expression, the checkpoint checks that the captured text string matches the expected date format, rather than checking 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 291
- “Using Regular Expressions in Checkpoints” on page 292

For information on defining regular expressions, including regular expression syntax, see “Defining Regular Expressions” on page 293.

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 in dialog boxes, such as the Object Properties or Object Repository dialog box, or in programmatic descriptions. For more information on programmatic descriptions, see “Using Programmatic Descriptions” on page 878.

For example, your 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 Web site or 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 286.

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 286.

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 878.

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, refer to the Regular Expressions section in the Microsoft VBScript documentation (choose **Help > QuickTest Professional Help** to open the QuickTest Professional Help. Then choose **VBScript Reference > VBScript > User's Guide > Introduction to Regular Expressions**).

Using the Backslash Character

A backslash (\) instructs QuickTest to treat the next character as a literal character, if it is otherwise a special character. The backslash (\) can also instruct QuickTest to recognize certain ordinary characters as special characters. For example, QuickTest recognizes \n as the special 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:

mercurytours.mercuryinteractive.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:

mercurytours\\.mercuryinteractive\\.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 298. For more information on the | regular expression character, see “Matching One of Several Regular Expressions” on page 299.

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

In order 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 foobar 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 in order to find zero or more occurrences of any character (except \n).

For example,

start.*

matches start, started, starting, starter, etc.

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)

15

Working with the Keyword View

The Keyword View provides an easy way to create, view, and modify tests and components in a graphical easy-to-use format.

This chapter describes:

- About Working with the Keyword View
- Understanding the Keyword View
- Working with Steps in the Keyword View
- Setting Keyword View Display Options
- Viewing Properties of Step Elements in the Keyword View
- Using Conditional and Loop Statements in the Keyword View
- Working with Breakpoints in the Keyword View

About Working with the Keyword View

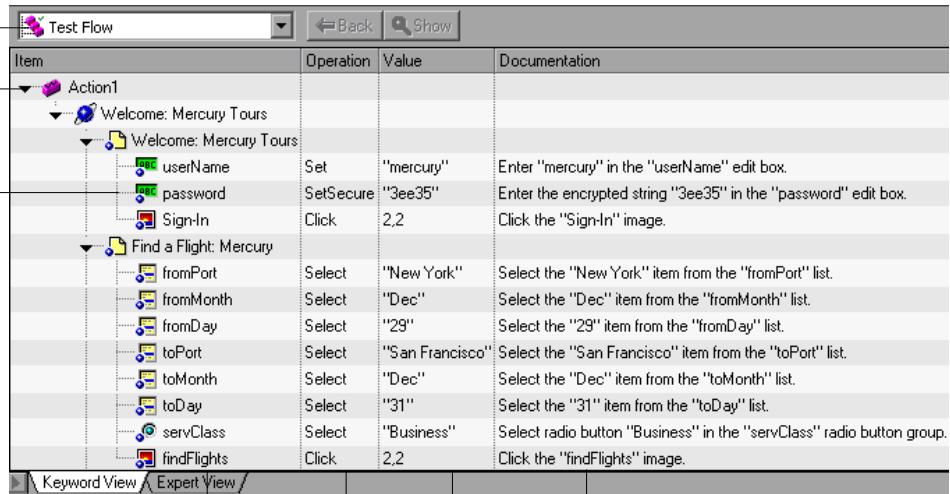
The Keyword View enables you to create and view the steps of your test or component in a modular, table format. Each step is a row in the Keyword View, comprised of individual parts that you can easily modify. You create and modify steps by selecting items and operations in the Keyword View and entering information as required. Each step is automatically documented as you complete it, enabling you to view a description of your test or component in understandable sentences. You can also use these descriptions as instructions for manual testing, if required.

You can use the Keyword View to easily add steps to your test or component, and also to modify existing steps. You simply 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. The Keyword View does not require any programming knowledge. The programming required to actually perform each step of the test or component is done automatically behind-the-scenes by QuickTest.

Note: The Keyword View replaces the Tree View found in earlier versions of QuickTest. Most of the operations you could perform from the Tree View can be performed in a similar manner from the Keyword View. For example, right-click on a step to access context-sensitive options for that step, such as inserting checkpoints, output values, and action-related operations.

Understanding the Keyword View

The Keyword View is comprised of a table-like view, where each step is a separate row in the table, and each column represents different parts of the steps. The columns displayed vary according to your selection. For more information, see “Setting Keyword View Display Options” on page 323.



The screenshot shows the Keyword View interface with the following structure:

- Action toolbar:** Displays "Test Flow" and includes Back and Show buttons.
- Action:** A tree view showing "Action1" expanded to reveal "Welcome: Mercury Tours".
- Step:** A table with four columns: Item, Operation, Value, and Documentation.

Items listed in the Step table:

Item	Operation	Value	Documentation
Welcome: Mercury Tours			
Welcome: Mercury Tours	Set	"mercury"	Enter "mercury" in the "userName" edit box.
Welcome: Mercury Tours	SetSecure	"3ee35"	Enter the encrypted string "3ee35" in the "password" edit box.
Welcome: Mercury Tours	Click	2,2	Click the "Sign-In" image.
Find a Flight: Mercury	Select	"New York"	Select the "New York" item from the "fromPort" list.
Find a Flight: Mercury	Select	"Dec"	Select the "Dec" item from the "fromMonth" list.
Find a Flight: Mercury	Select	"29"	Select the "29" item from the "fromDay" list.
Find a Flight: Mercury	Select	"San Francisco"	Select the "San Francisco" item from the "toPort" list.
Find a Flight: Mercury	Select	"Dec"	Select the "Dec" item from the "toMonth" list.
Find a Flight: Mercury	Select	"31"	Select the "31" item from the "toDay" list.
Find a Flight: Mercury	Select	"Business"	Select radio button "Business" in the "servClass" radio button group.
Find a Flight: Mercury	Click	2,2	Click the "findFlights" image.

At the bottom, tabs for "Keyword View" (selected) and "Expert View" are visible.

Keyword View columns

 The Action toolbar enables you to view either the flow of all the action calls in your test or the content of a specific action in your test. For more information, see Chapter 17, “Working with Actions”. The Action toolbar is only available if at least one of the actions in your test is reusable, and is not available for components.

Each step you perform on your application during a recording session is recorded as a row in the Keyword View. For example, the Keyword View could contain the following rows:



The screenshot shows a single row in the Keyword View table:

Item	Operation	Value	Documentation
Welcome: Mercury Tours	Set	"mercury"	Enter "mercury" in the "userName" edit box.
Welcome: Mercury Tours	SetSecure	"3ee35"	Enter the encrypted string "3ee35" in the "password" edit box.
Welcome: Mercury Tours	Click	2,2	Click the "Sign-In" image.

These rows show the following three steps that are all performed on the **Welcome: Mercury Tours** page of the Mercury Tours sample web site:

- **mercury** is entered in the **userName** edit box.
- The encrypted string **3ee35** 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.

Tip: You can print the contents of the Keyword View to your Windows default printer, or preview it on screen before printing. For more information, see “Printing a Test or Component” on page 99.

Understanding the Keyword View Columns

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. For more detailed information on each column, see “Modifying a Step” on page 314.

Note: If you do not see one or more of the columns described below in your Keyword View, you can use the Keyword View Options dialog box to display them. For more information, see “Setting Keyword View Display Options” on page 323.

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 or components, 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 icon, press the minus key (-) on your keyboard number pad, press the left arrow key on your keypad, or right-click the item and choose **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, choose **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 choose **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, choose **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 row) 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 or functions) 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.

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 or component.

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.

Documentation Column

Read-only auto-documentation of what the step does, in an easy-to-understand sentence.

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 column containing a list is selected:
 - You can press ENTER or SHIFT+F4 to open the list for that column.
 - 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 letter(s). The typed sequence is highlighted white.
- 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 last cell in a row, the right arrow key moves the focus to the first cell in the next row.
 - 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 the **Value** column 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 column), you can type a letter to jump to the next row that starts with that letter.

Working with Steps in the Keyword View

You can use the Keyword View to add steps at any point in your test or component. You can also modify or delete steps as required, using standard editing commands to easily make changes.

You can also view properties for items such as checkpoints, output values, and actions, use conditional and loop statements to manipulate the flow of your test or component, and insert breakpoints to assist you in debugging it.

Adding an Action

 Actions are the highest level of the test hierarchy, and contain all the steps that are part of that action. In the Keyword View, you can view either the flow of all the action calls in the test, or the content of a specific reusable action, using the options in the Actions toolbar.

You can insert a new action, a call to an action, or a copy of an action, to your test. For more information about inserting and using actions in the Keyword View, see Chapter 17, “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 “Moving Actions and Steps in the Hierarchy” on page 321.

Note: Components do not contain actions. When working with components, each component is a single entity. It cannot contain multiple components or nested calls to other components.

Adding a Standard Step

You can use the Keyword View to add a step at any point in your test or component. You can either add a step below the currently selected step, at the end of a test or component, or at the beginning of a new test or component. 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 313.

Tip: You can also add a step using the Step Generator. For more information, see “Inserting Steps Using the Step Generator” on page 467.

To add a standard step:

- 1 Select the row after which you want to add the new step, and choose **Insert > New Step** or press F8. 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 Recorded Object Hierarchy” on page 311.
-

Tip: If you want to add a step at the end of your test or component, or at the beginning of a new test or component, click in the first empty row of the Keyword 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 Recorded Object Hierarchy” on page 311.

- 2 Specify the content of the step by modifying it, as described in “Modifying a Step” on page 314.

Understanding the QuickTest Recorded Object Hierarchy

When you add a new step to your test or component in the Keyword View, the step is added as a sibling step or sub-step of the selected step, according to the QuickTest recorded object hierarchy. Following is a description of the recorded hierarchy, and then specific information describing where new steps are inserted in this hierarchy.

The recorded object hierarchy is composed of two or more levels of test objects. The top level is an object that represents a window, dialog, or browser type object, depending on the environment. Depending on the actual object on which you performed an operation, that object may be recorded as a second level object, for example, Window > WinToolbar, or there may be another object at the second level, and then the object on which you performed the operation is recorded as a third level object, for example, Browser > Page > WebButton.

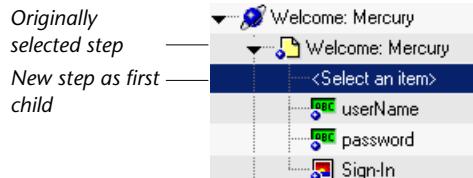
Note that when testing ActiveX objects in a browser, the top-level ActiveX object is recorded within the standard Web object hierarchy, for example, Browser > Page > ActiveX.

Even though the object on which you record may be embedded in several levels of objects, the recorded hierarchy does not include these objects. For example, even if the WebButton object on which you record is actually contained in several nested WebTable objects, which are all contained within a Browser and Page, the recorded hierarchy is only Browser > Page > WebButton.

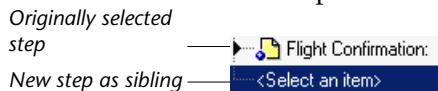
An object that can potentially contain a lower-level object is called a container object. All top-level objects in the recorded hierarchy are container objects. If a second-level object contains third-level objects according to the QuickTest recorded object hierarchy, then it is also considered a container object. For example, in the step Browser > Page > Edit > Set “David”, Browser and Page are both container objects.

When you add a new step to the Keyword View, it is added either as a sibling step or sub-step of the currently selected step, as follows:

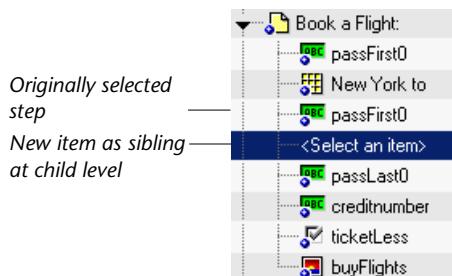
- If the selected step is a container object, and the branch for the selected step is expanded, the new step is inserted as the first sub-step of the container object.



- If the selected step is at the end of the test or component, and it is a container object, and the branch for the selected step is collapsed, the new step is inserted as a sibling step to the selected step, (after any sub-steps of the selected step).



- If the selected step is at the lowest level of the recorded hierarchy, the new step is inserted as a sibling step immediately after the selected step.

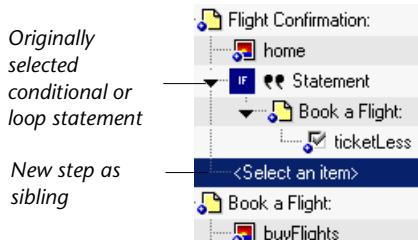


Adding a Standard Step After a Conditional or Loop Block

After you add a conditional or loop statement to your test or component, 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 about conditional and loop statements, see Chapter 20, “Adding Steps Containing Programming Logic.”

To add a standard step outside of 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 choose **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 “Modifying a Step” on page 314.

Adding Other Types of Steps

In addition to adding standard statement steps to your test or component 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 a Step” on page 314.

- You can insert a checkpoint step, for more information see Chapter 6, “Understanding Checkpoints”.
- You can insert an output value step, for more information see Chapter 13, “Outputting Values”.
- You can insert a step that sends information to the results, a step that puts a comment line in your test or component, a step that synchronizes your test or component with your application, or a step that measures a transaction in your test (transactions are not available for components). For more information see Chapter 20, “Adding Steps Containing Programming Logic”.
- You can insert a step that calls a WinRunner test or function, for more information see Chapter 39, “Working with WinRunner”.

Modifying 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-sensitive 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 within a test, action, or component. For more information, see “Moving Actions and Steps in the Hierarchy” on page 321.

To modify a step, click in the cell for the part of the step you want to modify and specify the content of the call, as described below. Each cell in the step row represents a different part of the step.

Item

Click in the **Item** cell, then click the arrow button and select the item on which you want to perform the step from the displayed list. If you have just entered a new step, the list is displayed automatically as soon as you create the new step. You can choose an item from one of the following:

- A test object in the object repository. You can either choose a test object from the list, or choose **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 also select an object directly from your application and add it to the object repository so that you can use it in the step. For more information, see “Understanding the Select Object for Step Dialog Box” on page 319.
- 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 467.

Operation

Click in the **Operation** cell, then click the 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 in order to view the list of available operations in the **Operation** column.

Value

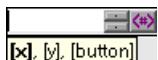
Click the **Value** cell and enter the argument values for the selected operation, or the value for the statement. The **Value** cell is partitioned according to the number of possible arguments of the selected operation. Each partition contains different options, depending on the type of argument that can be entered in the partition, as follows:

Argument Partition	Argument Type	Instructions
	String	Enables you to enter any alphanumeric string enclosed by quotes. If you do not enter the quotes, QuickTest adds them automatically.
	Integer	Enables you to enter any number, or use the up and down arrows to select a number.

Argument Partition	Argument Type	Instructions
 False	Boolean	Enables you to select a True or False value from the drop-down list.
	Any	<p>A variant argument type, which enables you to enter any of the above argument value types. For example, if the operation requires a string as the argument value, 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.</p> <p>Note: Some operations have arguments of Any type, but actually expect to receive either a string or a number. If you are specifying an argument value of type Any, you must specify it in the format that is expected by the operation.</p>

Note: If you want to specify a variable as an argument value, you can do this only in the Expert View.

When you click in the **Value** cell, a tooltip displays information regarding 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.



You can also parameterize the value in any partition of the **Value** column. Click the parameterization button  to open the Parameter Options dialog box in which you can parameterize the value. For more information, see Chapter 12, “Parameterizing Values”.

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.

Assignment

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.

Comment

Click in the **Comment** cell to enter textual notes about the step. Text you enter in this cell is also displayed as an inline comment in the Expert View.

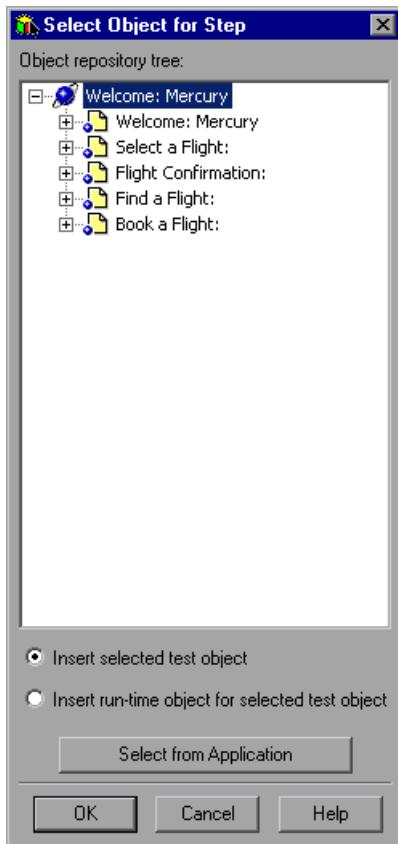
Note: You can also enter a comment on a new line below the currently selected step by choosing **Insert > Step > Comment**. For more information, see “Adding Comments” on page 498.

Documentation

This cell displays an explanation (in read-only format) 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.

Understanding the Select Object for Step Dialog Box

The Select Object for Step dialog box displays the object repository tree and enables you to select an object for your step from the object repository or from your application, or enter an **Object** statement for a selected test object.



You can select any object in the object repository tree for your new step, or you can select the **Insert run-time object for selected test object** option to enter an **Object** statement in your test or component. For more information on the object repository, see Chapter 4, “Managing Test Objects.” For more information on **Object** statements, see “Accessing Run-Time Object Properties and Methods” on page 899.

If the object that you want to use in the new step is not in the object repository, you can select an object from your application.

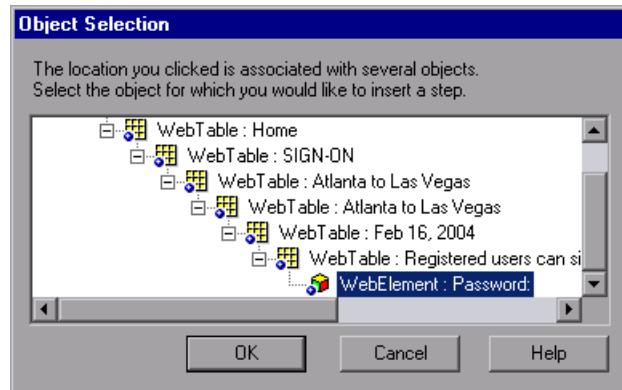
When you click **OK**, the object is displayed in the **Item** column in the Keyword View. You can now specify the operation for the selected object. For more information, see “Modifying a Step” on page 314.

To select an object from your application:

- 1** Click the **Select from Application** button. QuickTest is minimized.
- 2** Use the pointing hand to click on the required object in your application.

Tip: You can hold the CTRL key to change the window focus or perform operations such as a right-click or mouseover to display a context menu. If the window containing the object you want to click is partially hidden by another window, you can also hold the pointing hand over the partially hidden window for a few seconds until the window comes into the foreground and you can point and click on the object you want. Note that pressing the CTRL key does not enable you to select an application from the Windows taskbar, therefore you must make sure that the window you want to access is not minimized.

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 object repository tree in the Select Object for Step dialog box.
- 4 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 “Modifying a Step” on page 314.

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.

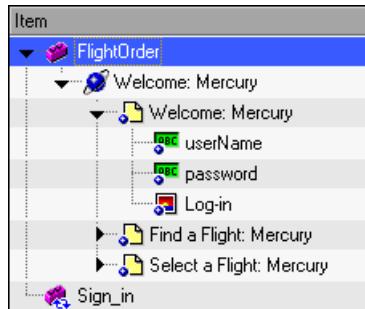
Moving Actions and Steps in the Hierarchy

You can move an action or step to a different location within a test or component by simply dragging it up or down the tree in the **Item** column and dropping it at the required location. You can also move a step by copying or cutting it to the Clipboard and then pasting it in the required location. When you move an action or step, you also move all of its sub-steps, if any.

Note:  If you copy an action (**Edit > 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 354. You cannot use the **Copy** and **Paste** options to move an action within the test hierarchy.

You can only move an action or step to a different location in the same level in the hierarchy (a sibling level). You cannot move an action or step to be above the first action or step in a hierarchy. However, you can move the first action or step in a hierarchy below any other action or step in the hierarchy.

For example, in the hierarchy shown below, you cannot drag the **Sign_in** action above the **FlightOrder** action, but you can drag the **FlightOrder** action below the **Sign_in** action. You can also move the **Select a Flight: Mercury** step (and sub-steps) above the **Find a Flight: Mercury** step (and sub-steps), and you can move the **userName** step below the **password** step. You cannot move the **password** step above the **userName** step.



Note: You cannot copy an item if one of its cells is in edit mode.

Deleting an Item

You can delete an item in the Keyword View. When an item has both an operation and sub-steps defined for it, as in the image below, you can choose whether to delete only the operation of the item, or to delete the item and all of its sub-steps.

Item	Operation	Value
▼ Action1		
▼ Welcome: Mercury	Navigate	"http://newtours.mercuryinteractive.com"
▼ Welcome: Mercury		
▼ userName	Set	"nicole"
▼ password	SetSecure	"3ee357f628811830704e"
▼ Sign-In	Click	21,2

Note: You cannot delete an item if one of its cells is in edit mode.

To delete an item:

- 1** Select the row for the item you want to delete.
- 2** Choose **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).

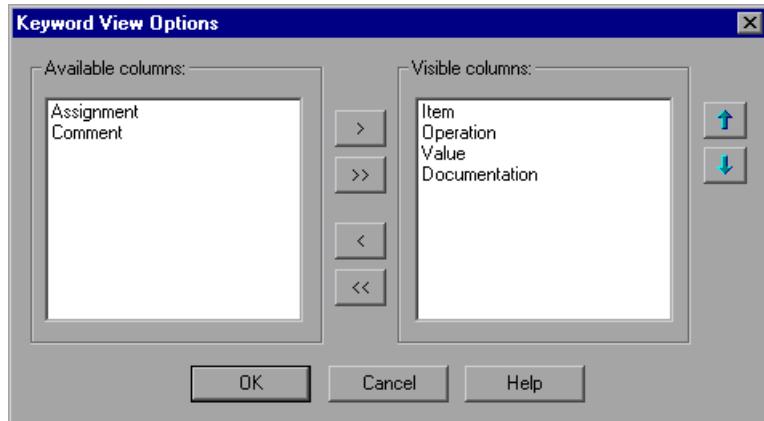
Setting Keyword View Display Options

You can specify which columns you want to display in the Keyword View, and also specify the order in which they are displayed.

Tip: You can choose to display only the **Item** and **Documentation** columns and 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 or Component” on page 99.

To specify Keyword View display options:

- 1 Choose **Tools > Keyword View Options**. The Keyword View Options dialog box opens:



The **Available columns** box lists columns not currently displayed in the Keyword View. The **Visible columns** box lists 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** boxes.

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.

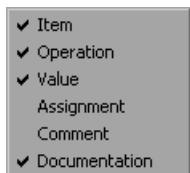
- 3 In the **Visible columns** box, set the order in which columns appear in the Keyword View by selecting one or more columns and then using the up and down arrow buttons.



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.
-

Tip: You can also 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.



You can also 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 for which you want to view the properties, 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.

Using Conditional and Loop Statements in the Keyword View

You can control the flow of your test or component with conditional statements and loop statements. Using conditional statements, you can incorporate decision making into your tests or components. 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

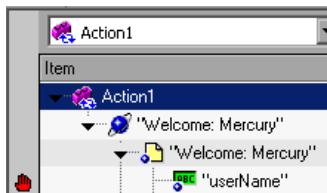
After you insert a conditional or loop statement in the Keyword View, you can insert or record steps after the statement to include them in the conditional or loop block.

For more information on including conditional and loop statements in your test or components, see Chapter 20, “Adding Steps Containing Programming Logic”. For more information on inserting steps after a conditional or loop block, see “Adding a Standard Step After a Conditional or Loop Block” on page 313.

Working with Breakpoints in the Keyword View

You can easily 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.

You insert a breakpoint in the Keyword View by clicking in the left margin at the point where you want to insert the breakpoint, or by selecting a step and pressing F9 or choosing **Debug > Add/Remove Breakpoint**. A red breakpoint icon is displayed. You can remove a breakpoint by clicking the breakpoint icon, or by selecting a step and pressing F9 or choosing **Debug > Add/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 or component contains breakpoints.

For more information on breakpoints, see Chapter 22, “Debugging Tests and Components”.

16

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 record and run tests or components on objects that are not normally recognized by QuickTest.

This chapter describes:

- ▶ About Learning Virtual Objects
- ▶ Understanding Virtual Objects
- ▶ Understanding the Virtual Object Manager
- ▶ Defining a Virtual Object
- ▶ Removing or Disabling Virtual Object Definitions

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 record a test on 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 record a 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 a test or component, 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. 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. A virtual object *collection* is a group of virtual objects that is stored in the Virtual Object Manager under a descriptive name.

Note: QuickTest does not support virtual objects for analog or low-level recording. For additional information about low-level recording, see “Recording and Running Tests” on page 1003.

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 or component, 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.

Note: The Web page or application window must be the same size and in the same position when recording and running tests or components as it was when you defined the virtual object.

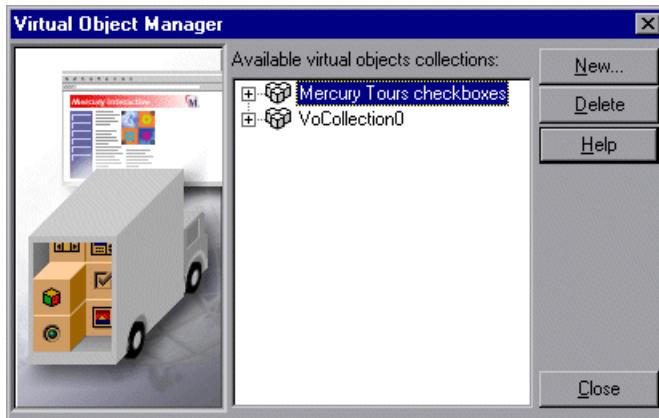
You can disable recognition of virtual objects without deleting them from the Virtual Object Manager. For additional information, see “Removing or Disabling Virtual Object Definitions” on page 338.

Notes: You can use virtual objects only when recording and running a test or component. You cannot insert any type of checkpoint on a virtual object, or use the Object Spy to view its properties.

In order 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 contains all the virtual object collections defined on your computer. From the Virtual Object Manager, you can define and delete virtual objects and collections.



Available virtual object collections list—Displays the virtual object collections defined on your computer and the virtual objects contained in each one. Use the + and - signs next to a collection to view or hide the virtual objects defined in that collection.

New—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 333.

Delete—Deletes the selected virtual object or virtual object collection. For more information, see “Removing or Disabling Virtual Object Definitions” on page 338.

Note: The virtual object collections displayed in the Virtual Object Manager are stored on your computer and not with the tests or components that contain virtual object steps. This means that if you use a virtual object in a test or component 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.

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 you can click or double-click and that record a **Click** or **DblClick** step. 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.

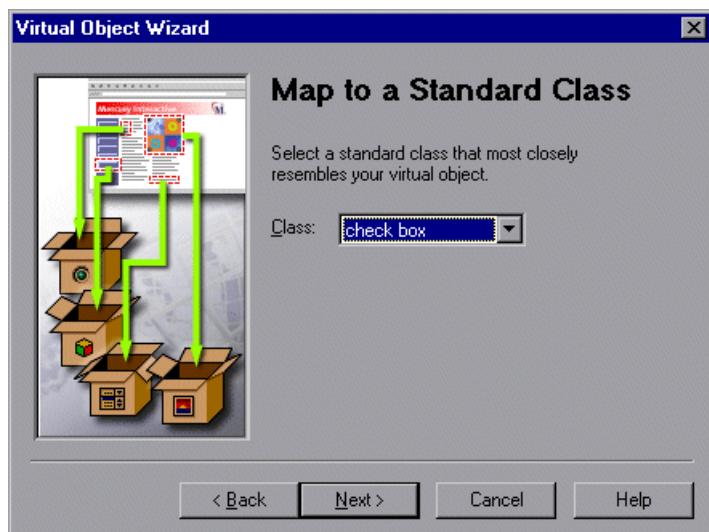
To define a virtual object:

- 1 With QuickTest open (but not in record mode), open your Web site or application and display the object containing the area you want to define as a virtual object.

- 2** In QuickTest, choose **Tools > Virtual Objects > New Virtual Object**. Alternatively, from the Virtual Object Manager, click **New**. The Virtual Object Wizard opens. Click **Next**.

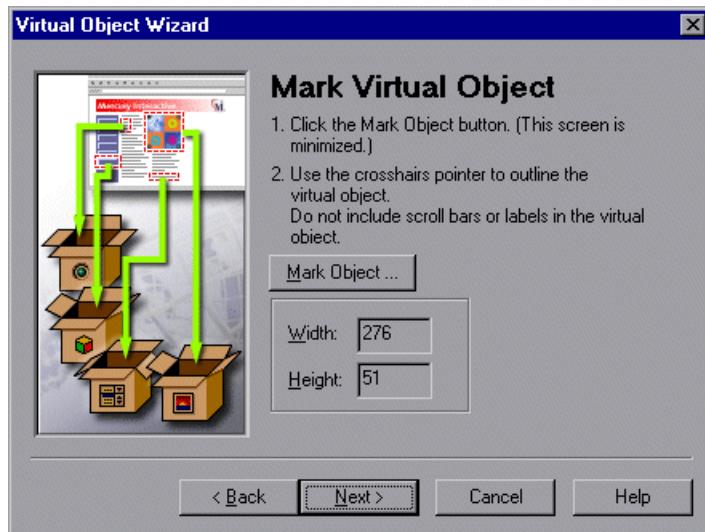


- 3** Select a standard class to which you want to map your virtual object.



If you select the list class, specify the number of rows in the virtual object. For the table class, select the number of rows and columns. Click **Next**.

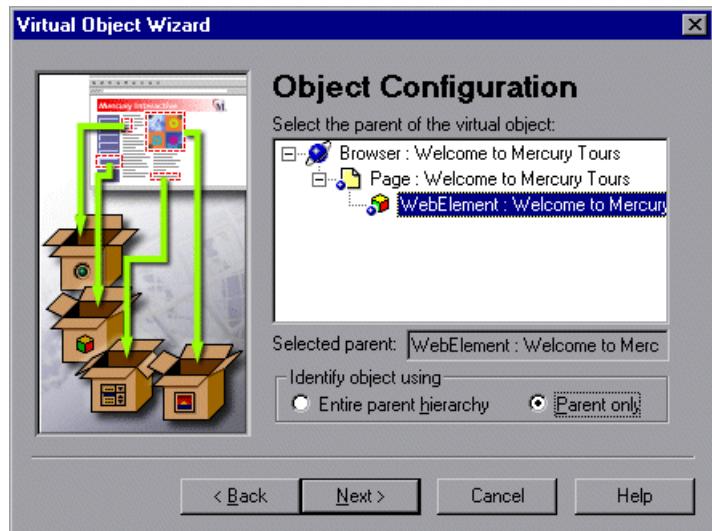
4 Click **Mark Object**.



The QuickTest window and the Virtual Object Wizard are minimized. 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. Click **Next**.

Note: The virtual object should not overlap other virtual objects in your application or Web page. If the virtual object overlaps another virtual object, QuickTest may not record or run tests or components correctly on the virtual objects.

- 5 Click an object in the object tree to assign it as the parent of the virtual object.

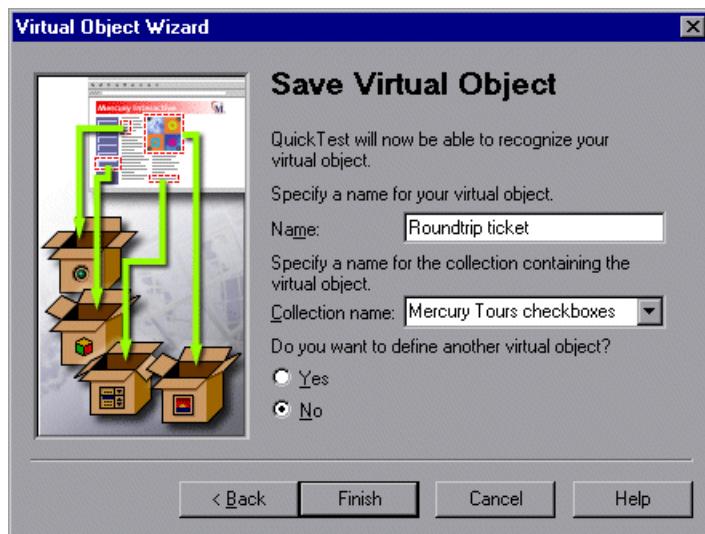


The coordinates of the virtual object outline are relative to the parent object you select.

- 6 In the **Identify object using** box, select how you want QuickTest to identify and map the virtual object.
- If you want QuickTest to identify all occurrences of the virtual object, select **parent only**. 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")`.
 - If you want QuickTest to identify the virtual object in one occurrence only, select **entire parent hierarchy**. QuickTest identifies the virtual object only if it has the exact parent hierarchy. For example, if the virtual object was defined using `Browser("A").Page("B").Image("C")`, QuickTest will not recognize it if the hierarchy changes to `Browser("X").Page("B").Image("C")`.

Click **Next**.

- 7 Specify a name and a collection for the virtual object. Choose from the list of collections or create a new one by entering a new name in the **Collection name** box.



- 8 To add the virtual object to the Virtual Object Manager and close the wizard, select **No** and then click **Finish**.

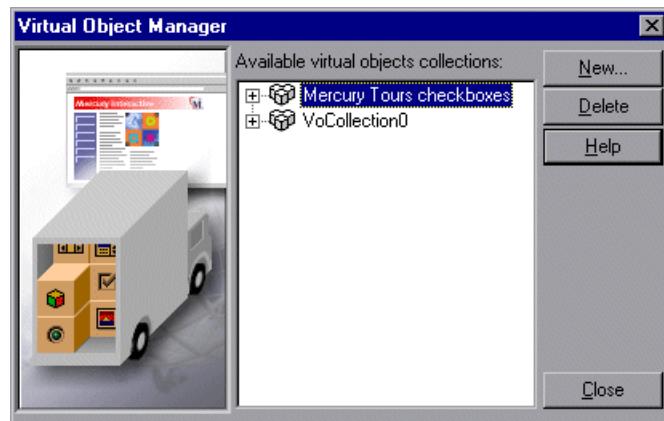
To add the virtual object to the Virtual Object Manager and define another virtual object, select **Yes** and then click **Next**. The wizard returns to the Map to a Standard Class screen, where you can define the next virtual object.

Removing or Disabling Virtual Object Definitions

You can remove virtual objects from your test or component by deleting them or by disabling recognition of these objects while recording.

To delete a virtual object:

- 1 Choose **Tools > Virtual Objects > Virtual Object Manager**. The Virtual Object Manager opens.



- 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** Choose **Tools > Options** or click the **Options** toolbar button. The Options dialog box opens.
- 2** In the General tab, 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 tab of the Options dialog box is cleared. For more information, see “Setting General Testing Options” on page 614.

17

Working with Actions



You can divide your test into actions to streamline the process of testing your application or Web site.

Note: Components do not contain actions, and cannot be divided into actions.

This chapter describes:

- ▶ About Working with Actions
- ▶ Using Multiple Actions in a Test
- ▶ Using Global and Action Data Sheets
- ▶ Using the Action Toolbar
- ▶ Creating New Actions
- ▶ Inserting Calls to Existing Actions
- ▶ Nesting Actions
- ▶ Splitting Actions
- ▶ Using Action Parameters
- ▶ Setting Action Properties
- ▶ Setting Action Call Properties
- ▶ Sharing Action Information
- ▶ Exiting an Action
- ▶ Removing Actions from a Test

- Renaming Actions
- Creating an Action Template
- Guidelines for Working with Actions

About Working with Actions

Actions help divide your test into logical units, like the main sections of a Web site, or specific activities that you perform in your application.

A test is made up 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 more modular and efficient tests.

An action consists of its own test script, including all the steps recorded within that action, and its own object repository if the test is in per-action repository mode. For more information on object repositories, see Chapter 34, “Choosing the Object Repository Mode.”

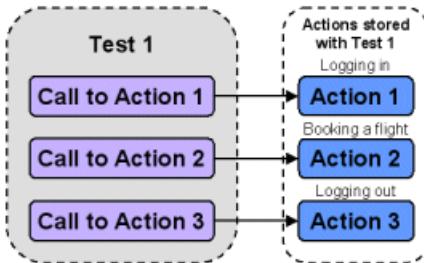
Each action is stored together with the test in which you created it. You can insert a call to an action that is stored with the test and, depending on the properties of the action, you may also be able to call an action stored with another test.

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. Once you have created 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 re-record 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,” below. For information on parameterizing tests, see Chapter 12, “Parameterizing Values,” and Chapter 13, “Outputting Values.”

Note: You can also run a single action, or part of an action, using the **Run from Step** option. For more information, see Chapter 21, “Running Tests and Components.”

You can 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 364.

Using Multiple Actions in a Test

When you create a test, it includes one action. All the steps you record 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, or by inserting calls to existing actions. There are three kinds of actions:

- **non-reusable action**—an action that can be called only in the test with which it is stored, and can be called only once.
- **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.
- **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 about creating and calling new actions, see “Creating New Actions” on page 348. For more information about inserting calls to existing actions, see “Inserting Calls to Existing Actions” on page 352.

By default, new actions are non-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 360). 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 23, “Analyzing Test 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. 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.

Note: When you parameterize the value of an object property using a Data Table parameter, you cannot select **Current action sheet (local)** if your test uses a shared object repository file. For more information, see Chapter 34, “Choosing the Object Repository Mode.”

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 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 about setting action iteration preferences, see "Inserting Calls to Existing Actions" on page 357.

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 18, “Working with Data Tables.” For more information about parameterization, see Chapter 12, “Parameterizing Values.” For more information about output values, see Chapter 13, “Outputting Values.”

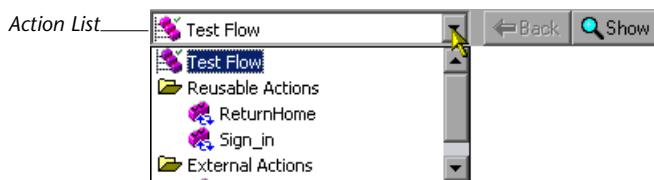
Using the Action Toolbar

The Action toolbar contains options that enable you to view all action calls in the test flow or to view the details of a selected action. By default, the Action toolbar is hidden in the Keyword View when you open QuickTest. The first time you insert a reusable or external action into a test, the Action toolbar is automatically displayed above the Keyword View.



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.”

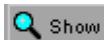
The *Action List* enables you to view either the entire test flow (the calls to the actions in the test) or you can view the steps for a selected reusable or external action. The test flow displays the overall flow of your test with all the action calls in your test. The test flow also enables you to 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 about reusable actions, see “Setting General Action Properties” on page 368.

There are three ways to open the action view for a reusable or external action in the Keyword View:

- In the test flow, double-click the call to the action you want to view.
- In the test flow, 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.



Note: In the Expert View, the Action List is always visible and the Expert View always displays the script for the selected action. For more information on the Expert View, see Chapter 36, “Working with the Expert View.”

Creating New Actions

You can create new actions and add calls to them during a recording session or while designing or editing your test.

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 360.

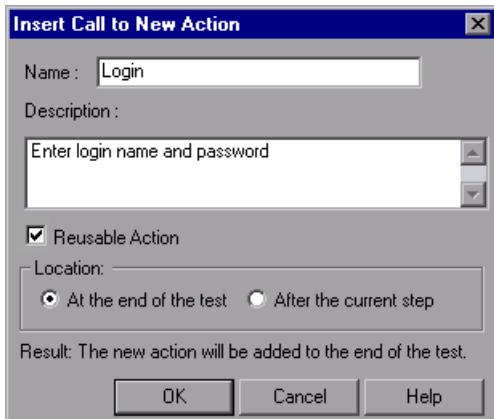
You can also split an existing action into two actions. For more information on splitting actions, see “Splitting Actions” on page 362.

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 While recording or editing your test, choose **Insert > Call to New Action** or click the **Insert Call to New Action** button.



The Insert Call to New Action dialog box opens.



- 3 Type a new action name or accept the default name.
- 4 If you wish, 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 select the action you want to call. For more information, see “Setting General Action Properties” on page 368.

- 5 Select **Reusable Action** if you want to be able to call the action from other tests or multiple times from within this test. You can also set or modify this setting at a later time using the Action Properties dialog box.

For more information about reusable actions, see “Using Multiple Actions in a Test” on page 344. For more information about the Action Properties dialog box, see “Setting Action Properties” on page 366.

- 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 about inserting action calls within actions, see “Nesting Actions” on page 360.

- 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 about moving actions, see “Moving Actions and Steps in the Hierarchy” on page 321.
- 8** If you inserted the call to the new action while editing your test, make sure your new action is selected and click **Record** to add steps to your new action.



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.

```
RunAction "Action2", onelteration, "MyValue", MyVariable
```

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", onelteration, Parameter("Axn1_In"),  
DataTable("Column1_out", dtLocalSheet)
```

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.

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 about the Expert View, see Chapter 36, “Working with the Expert View.” For more information on the **RunAction** statement, refer to the *QuickTest Professional Object Model Reference*.

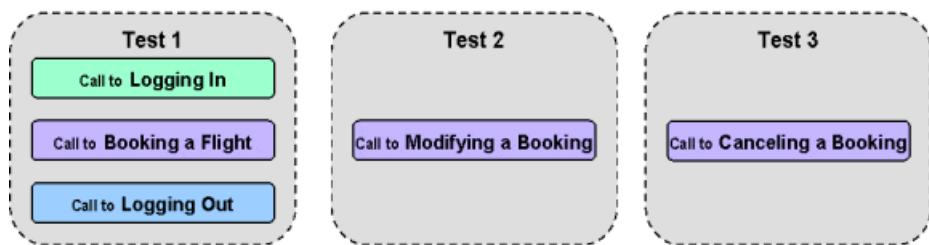
Inserting Calls to Existing Actions

When you plan a suite of tests, you may realize that each test requires some identical activities, such as logging in. For example, rather than recording the login process 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. Once you are satisfied with the action you created, you can insert calls to the existing action into other tests.

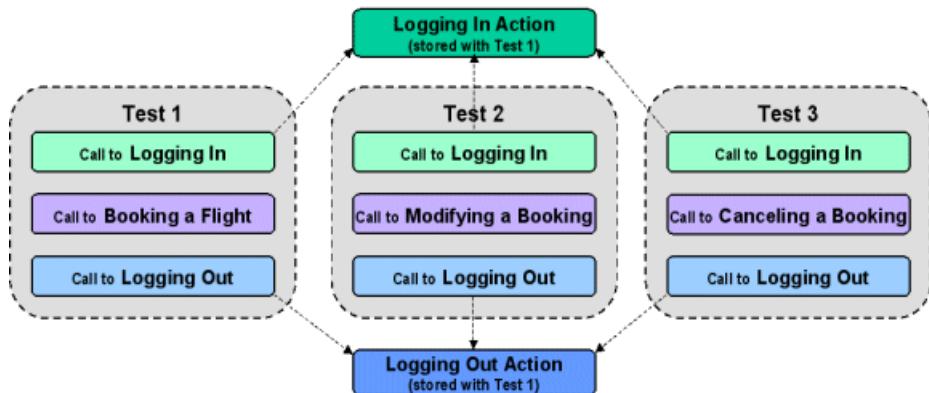
You can insert calls to an existing action by inserting a call to a copy of the action, or by inserting a call to the original action.

For example, suppose you wanted to record the following three tests in 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 as shown:



The following diagram shows an example of how to create your tests using calls to the original 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 uses per-action repository mode, the copied action's object repository is also copied together with the action.

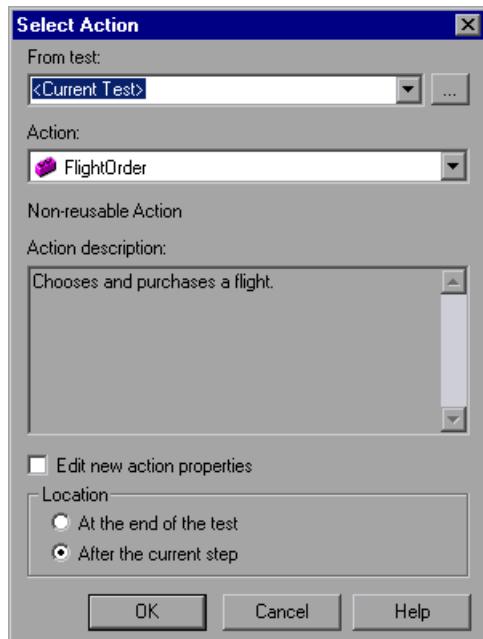
If the test into which you are copying uses a shared object repository, the copied action will use the same shared object repository as the calling test. Before running the test, confirm that the shared object repository contains all the objects that are in the copied action. Otherwise, the test may fail.

Note: If you are working in a test that uses per-action object repository mode, you cannot copy an action from a test using a shared object repository. For more information, see “Splitting and Copying Actions in Object Repository Per-Action Mode” on page 820.

The action is inserted into the test as an independent, non-reusable action (even if the original action was reusable). Once 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 While recording or editing your test, choose **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 tab of the Options dialog box. For more information, see Chapter 24, “Setting Folder Testing Options.”

- 3** In the **Action** list, select the action you want to insert. When you select an action, its type (**Non-reusable** or **Reusable Action**) and description, if one exists, are displayed. This helps you identify the action you want to copy. For more information about action descriptions see “Setting General Action Properties” on page 368.
-

Note: QuickTest disables the **Action** list if the test you are working in is using per-action repository mode and the test from which you want to copy uses a shared object repository. For more information, see “Splitting and Copying Actions in Object Repository Per-Action Mode” on page 820.

- 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 Properties” on page 366.
-

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 about inserting actions within actions, see “Nesting Actions” on page 360.

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 location options are 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 at a parallel (sibling) level within your test by dragging it to the desired location. For more information about moving actions, see “Moving Actions and Steps in the Hierarchy” on page 321.

Inserting Calls to Existing Actions

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 like linking to it. You can view the steps of the action in the action view, but you cannot modify them. If you call an external action, you can choose, however, 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.

If the test calling an action uses per-action repository mode, the called action’s object repository is read-only (as are the steps of the called action) in the test calling the action. If the test calling an action uses a shared object repository, the called action uses the same shared object repository as the calling test. In this case, before running the test, confirm that the shared object repository of the calling test contains all the objects that are in the called action. Otherwise, the test may fail.

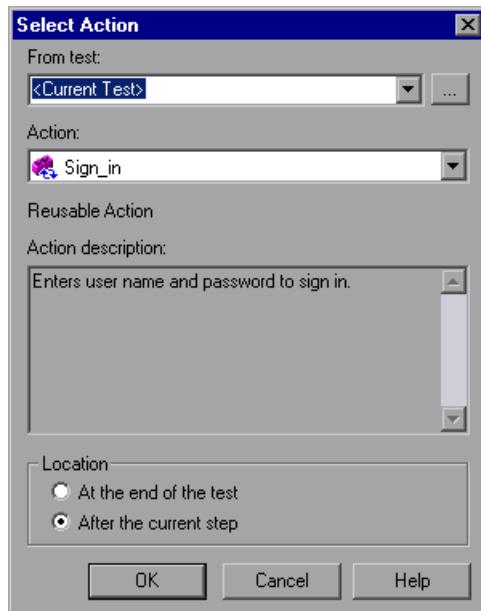
Note: If you are working in a test that uses per-action object repository mode, you cannot call an external action from a test using a shared object repository. For more information, see “Inserting Action Calls in Object Repository Per-Action Mode” on page 821.

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 Choose **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 tab of the Options dialog box. For more information, see Chapter 24, “Setting Folder Testing Options.”

- 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 about action descriptions, see “Setting General Action Properties” on page 368.

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. It also disables the option if the test in which you are working uses per-action object repository mode and the test from which you want to call an action uses a shared object repository. For more information, see “Inserting Action Calls in Object Repository Per-Action Mode” on page 821.

- 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 location options are disabled).

For more information about inserting actions within actions, see “Nesting Actions” on page 360.

- 5 Click **OK**. A call to the action  is inserted into the test flow. 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 about moving actions, see “Moving Actions and Steps in the Hierarchy” on page 321.
-

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.

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 action or another 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("memtype", dtGlobalSheet)	Select radio button <the value of the specified D
MemType	GetROPProperty	selected	Retrieve the current value of the selected propo
IF Statement		Mem_Type = "paid"	Check whether (Mem_Type = "paid") is true. If s
Paid_Mem			Call the Paid_Mem action.
ELSE IF Statement		Mem_Type = "free"	Otherwise, Check whether (Mem_Type = "free")
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").
            GetROPProperty ("value")
If Mem_Type="paid" Then
    RunAction "Paid_Mem", onelteration
ElseIf Mem_Type = "free" Then
    RunAction "Free_Mem", onelteration
Else
    RunAction "Preferred", onelteration
End If
```

For more information about inserting conditional statements, see “Using Conditional Statements” on page 484.

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 348, 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 352.

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:

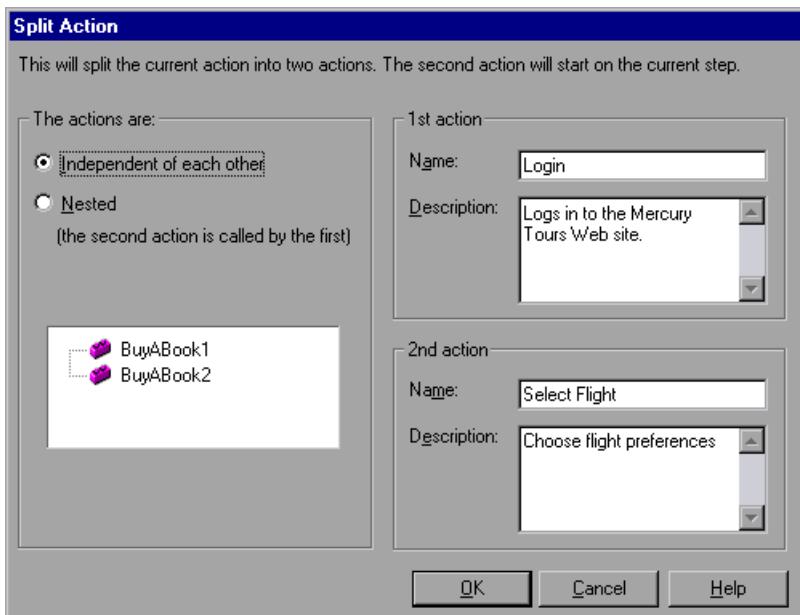
- an external action is selected
- the first step of an action is selected
- recording a test
- running a test
- you are working with a read-only test

To split an action:

- 1** Select the step before which you want the new (second) action to begin.



- 2** Choose **Step > Split Action**, click the **Split Action** button, or right-click the step and choose **Action > Split**. The Split Action dialog box opens.



- 3** Choose 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 wish, 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.

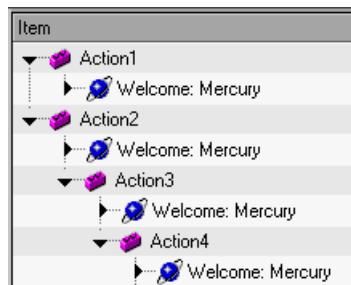
Using Action Parameters

Action parameters enable you to transfer values between actions and their nested actions in your test, from your test to a top-level action, or from a step in a top-level action back to the script or action 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 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. You specify the actual values that are provided to these parameters and the locations in which the output values are stored in the Parameter Values tab in the Action Call Properties dialog box.

You can specify input parameters for an action so it can receive input values from elsewhere in the test. Input values for an action can be retrieved from the test (for a top-level action) or from the parameters of the action that calls it (for a nested 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 (**Test > Settings > Parameters tab**) with the value that you want to use later in the test.
- 2 Define an input action parameter for Action2 (**Step > Action Properties > Parameters tab**) with the same value type as the input test parameter.
- 3 Parameterize the input action parameter value (**Step > Action Call Properties > Parameter Values tab**) using the input test parameter value you specified above.
- 4 Define an input action parameter for Action3 (**Step > Action Properties > Parameters tab**) with the same value type as the input test parameter.
- 5 Parameterize the input action parameter value.
 - Choose **Step > 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 *parameter* argument for the **RunAction** statement in the Expert View. For more information, see “Calling Actions with Parameters” on page 351.
- 6 Define an input action parameter for Action4 (**Step > Action Properties > Parameters tab**) with the same value type as the input test parameter.
- 7 Parameterize the input action parameter value.
 - Choose **Step > 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 *parameter* argument for the **RunAction** statement in the Expert View. For more information, see “Calling Actions with Parameters” on page 351.
- 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, or Component Parameters in Steps in the Expert View” on page 214.

An action's parameters are stored with the action and are the same for all calls to that action. If you change the action parameters defined for an action, and then view the action properties for a call to that same action in a different part of the test, you can see the action parameters have changed. However, the actual values specified for input action parameters and the locations specified for action output parameters 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 370, and “Setting Action Call Parameter Values” on page 379.

Setting Action Properties

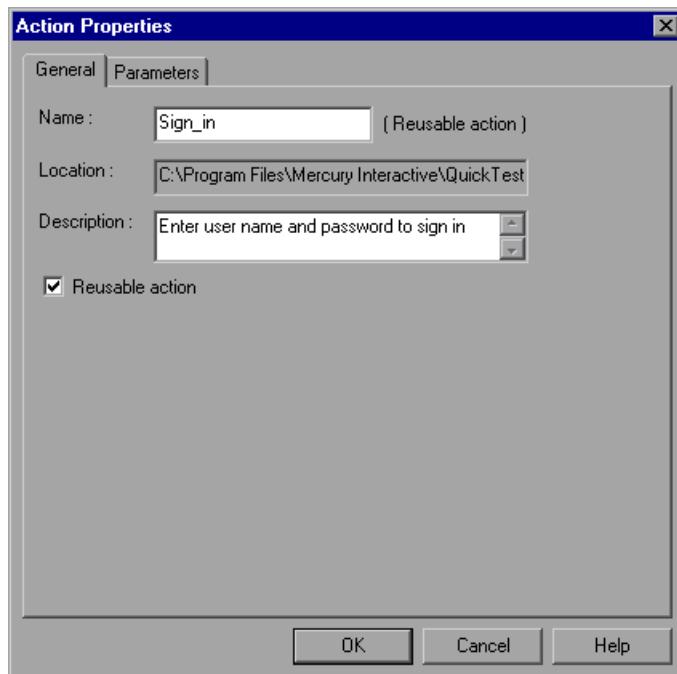
The Action Properties dialog box enables you to define options for the stored action. These settings apply each time the action is called. You can modify an action name, add or modify an action description, and set an action as reusable. You can also define input and output parameters to be used by the action, and, for an external action, you can set the Data Table and object repository definitions.

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 350.

You can open the Action Properties dialog box while recording or editing your test by:

- Choosing **Step > Action Properties** from the Keyword View when an action node is highlighted or from the Expert View.
- Right-clicking an action node in the Keyword View and selecting **Action Properties**.

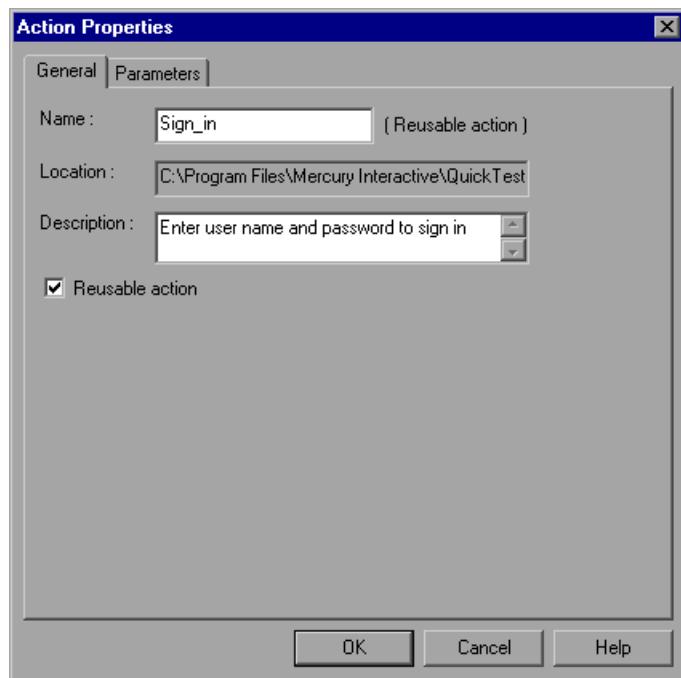
The Action Properties dialog box always contains the General tab and the Parameters tab as shown below:



Note: In addition to the tabs described above, the Action Properties dialog box for an external action also contains an External Action tab. For more information, see “Setting Properties for an External Action” on page 373.

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.



The General tab includes the following options:

Option	Description
Name	The name of the action.
Location	The folder or Quality Center path where the action is stored.
Description	<p>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 about inserting copies and calls to actions, see “Inserting Calls to Existing Actions” on page 352.</p> <p>Note: You can also add a description when inserting a call to a new action. For more information, see “Creating New Actions” on page 348.</p>
Reusable action	<p>Indicates whether the action is a reusable action. 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.</p> <p>When you change this setting, the action icon changes to a reusable  or non-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. The first time you convert an action to a reusable action within a test, the Test Flow box is displayed above the Keyword View. You can view the steps of the reusable action by selecting the action name in the Test Flow box.</p>

Notes:

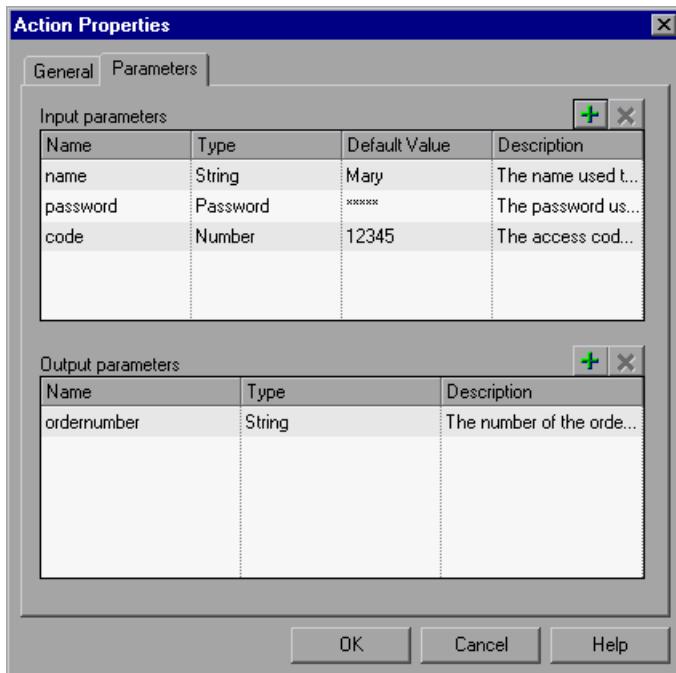
If the action is called more than once within the test flow or if the action is called by a reusable action, the **Reusable action** option is read-only. If you want to make the action non-reusable, remove the additional calls to the action from the test.

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 about the test flow and action views, see “Using the Action Toolbar” on page 347.

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 action that calls it (for a nested action). You can specify output parameters for an action, so that it can return values for use later in the test. For each input or output action parameter, you define a name and a type. 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.

For more information about using action parameters, see “Using Action Parameters” on page 364.



To add a new input or output action parameter:



- 1 Click the **Add** button next to 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.

- 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 positive or negative 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. The default value is an empty string.
 - **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 or 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 If you wish, click in the **Description** box, then enter a description of the parameter, for example, the purpose of the parameter in the action.

To modify an existing action parameter:

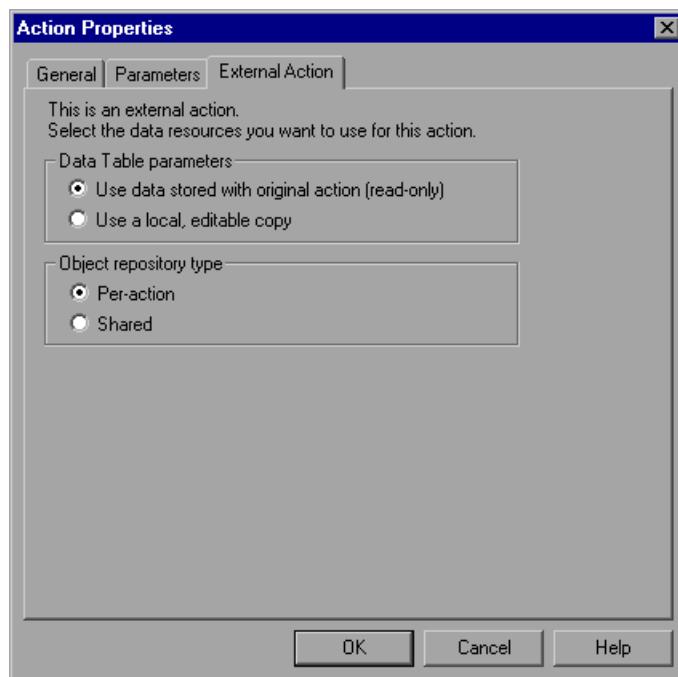
- 1 Select the parameter you want to modify from the **Input parameters** or **Output parameters** list.
- 2 Modify the values as necessary in the edit boxes of the parameter row.

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.

**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 and, if you are using the shared object repository mode, you can choose where to store the action's object information. You can set these options in the External Action tab of the Action Properties dialog box.



Note: When you open the Action Properties dialog box while working in a test in per-action repository mode or for an external action coming from a test that uses the shared object repository mode, the **Object repository type** options in the External Action tab are not available.

The External Action tab includes the following options:

Option	Description
Data Table parameters	<p>Indicates where to store the action's Data Table data:</p> <ul style="list-style-type: none">To use the original action's data, select Use data stored with the original action (read-only). When 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. When you select this option, a copy of the action's data sheet is stored in the test's Data Table and is independent of the original action. Changes to the original data sheet do not affect the calling test.
Object repository type	<p>Indicates which object repository mode the action uses:</p> <ul style="list-style-type: none">Choose Per-action to instruct QuickTest to refer to the external action's repository.Choose Shared to instruct QuickTest to use the test's shared object repository for this action. <p>Available only when the calling test uses a shared object repository and the external action comes from a test using per-action repository mode. For more information on object repository modes, see Chapter 34, "Choosing the Object Repository Mode."</p>

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, 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.

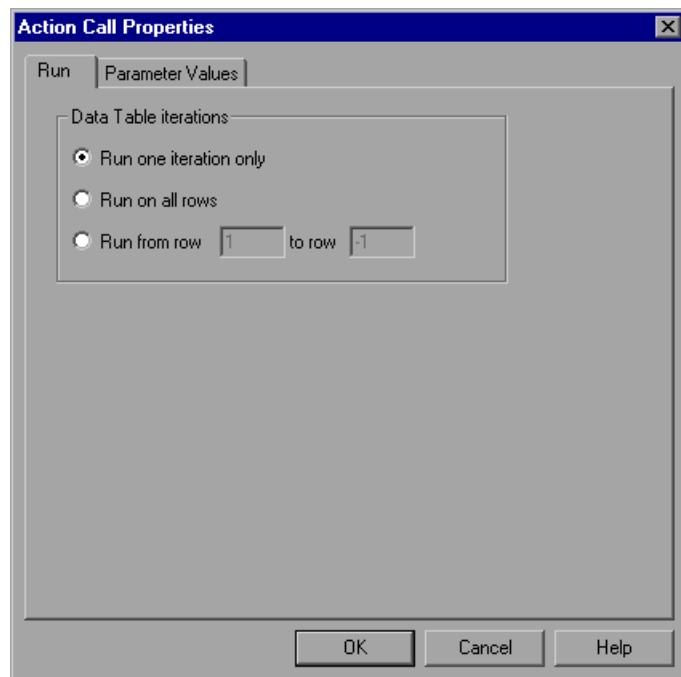
Note: The following sections describe how to define action call properties using the Action Call Properties dialog box. You can also define actions calls and action call parameters in the Expert View. For more information, see “Understanding Action Syntax in the Expert View” on page 350.

You can open the Action Call Properties dialog box while recording or editing your test by:

- Choosing **Step > Action Call Properties** from the Keyword View when an action node is highlighted.
- Right-clicking an action node in the Keyword View and selecting **Action Call Properties**.

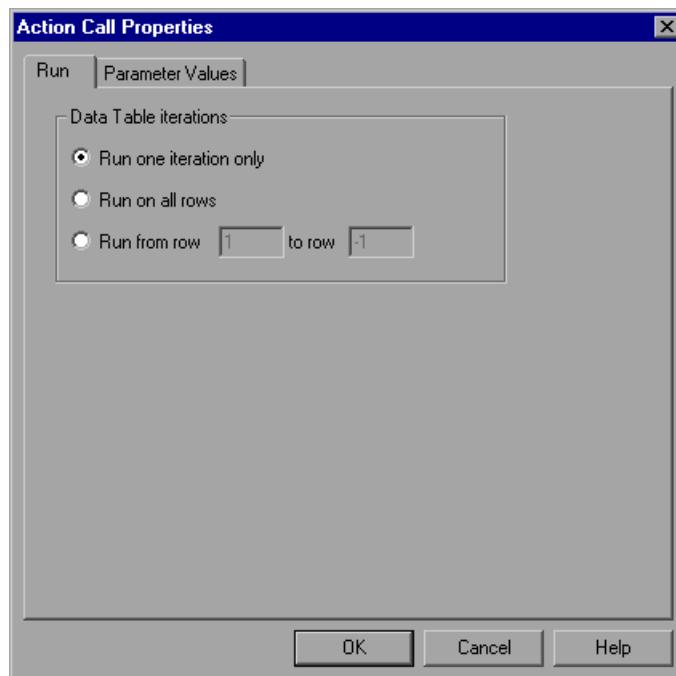
Part III • Creating Tests

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 as shown below:



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	<p>Runs the called action only once, using the row in the action's data sheet that corresponds to the current global iteration number. If the action's data sheet contains fewer rows than the global sheet, the last row of the action's data sheet is used for each subsequent test iteration.</p> <p>For example, suppose an action's data sheet has two rows and the global sheet has four rows. If you choose to run one iteration only for the action and you choose to run iterations on all rows of the global data sheet, then during each iteration of the test, this action will run only one iteration. The data that the action parameters use during each repetition of the test are based on the iteration number for the test. During the first iteration of the test, Data Table parameters in the action take data from the first row of the action's data sheet. In the second iteration of the test, Data Table parameters in the action take data from the second row of the action's data sheet. In the third and subsequent iterations of the test, the Data Table parameters in the action continue to take data from the second (last) row of the action's data sheet.</p>
Run on all rows	<p>Runs the called action with the number of iterations according to the number of rows in the action's Data Table.</p>
Run from row __ to row __	<p>Runs the called action with the number of iterations according to the specified row range.</p>

Notes:

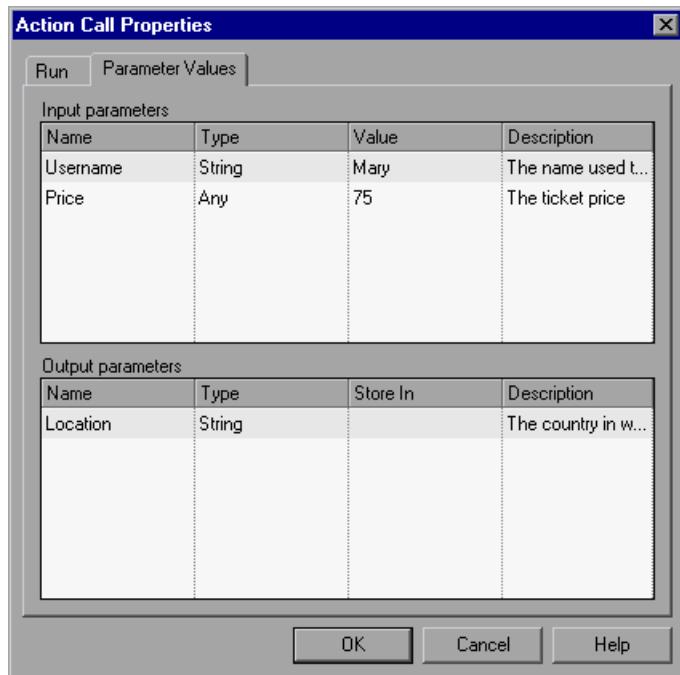
If you run multiple iterations on an action, the action must begin and end at the same point in the application, so that the application is in the proper location and state to run the next iteration of the action.

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 tab in the Test Settings dialog box. For more information, see Chapter 25, "Setting Options for Individual Tests or Components."

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.

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. If you do not set a value for an input action parameter in the Action Call Properties dialog box, the default value that is specified in the Action Properties dialog box is used.



For more information about defining input and output action parameters, see “Setting Action Properties” on page 366. For general information about using action parameters, see “Using Action Parameters” on page 364.

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 370.

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 action), Data Table parameter, environment parameter, or random number parameter. For more information, see Chapter 12, “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 or action parameter, a Data Table parameter, or an environment parameter. For more information, see “Storing Return Values and Action Output Parameter Values” on page 482.

- 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. For more information, “[Storing Values in Test, Action or Component Parameters](#)” on page 252.
- 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](#),” below.
- 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 383.
- 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 383.

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 13, “Outputting Values.” For more information on parameterization, see Chapter 12, “Parameterizing Values.” For more information about DataTable objects and methods, see Chapter 18, “Working with Data Tables,” and refer to the *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 12, “Parameterizing Values.” For information on the **Environment** object, refer to the *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

Once 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.Item("DateCheck")
```

For more information about the Dictionary object, refer to the VBScript Reference documentation (**Help > QuickTest Professional Help > VBScript Reference > Script Runtime**).

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 about these functions, refer to the *QuickTest Professional Object Model Reference*. For more information about the Test Results, see Chapter 23, “Analyzing Test Results.”

Removing Actions from a Test

The procedures and effects of removing calls to non-reusable actions, external actions, or reusable actions are different.

- When you remove a call to a non-reusable action, you also delete the action itself entirely as well as the action’s data sheet.
- When you remove a call to a reusable or external action, you remove the action from your test flow, but the action remains stored with the test in which it was created and is still displayed in the action list.
- When you remove an external action, you remove all calls and the action from the action list. The original action is not affected.
- When you remove a reusable action stored with your test, you delete all calls and the action entirely. This will cause any test calling this action to fail.

Removing a Non-Reusable Action

Removing a call to a non-reusable action from your test, deletes the action itself entirely as well as the action’s data sheet.

To remove a non-reusable action:

- 1 In the Keyword View, select the action you want to remove and press the **Delete** key on your keyboard or select **Edit > Delete**. A delete confirmation message box opens.
- 2 Click **Yes** to confirm.

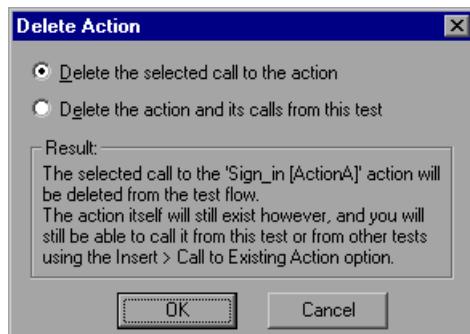
Removing a Call to a Reusable or External Action from the Test Flow

You should choose to remove a call to a reusable or external action if you want to remove the action from the test flow, but you still want the action to be available for other calls. When you choose this option, the action still exists even though it is removed from your test flow, and the action's data sheet remains. You can still view the action (and edit a reusable action) by selecting it from the Action List in the Keyword View or in the Expert View.

After you remove a call to an action, you can insert the action back into the test from which it was removed, or into any other test using the **Insert Call to Copy of Action** or **Insert Call to Existing Action** options. For more information see “Inserting Calls to Existing Actions” on page 352.

To remove a call to a reusable or external action from the test flow:

- 1 Select the **Test Flow** view from the Action List in the Keyword View.
- 2 Highlight the action you want to remove.
- 3 Choose **Edit > Delete** or press the **Delete** key on your keyboard. The Delete Action dialog box opens.



- 4 Choose **Delete the selected call to the action** and click OK. The action call is deleted. The action remains in the test's Action List.

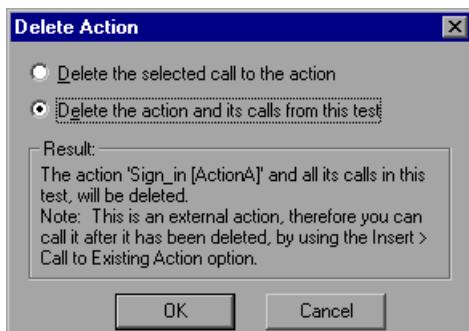
Removing an External Action from the Test

When you remove an external action from the test, the action is removed from the action list and the corresponding action sheet is removed from the Data Table. Columns related to this action that are located in the global sheet are not removed.

After you remove an external action from your test, you can reinsert it by choosing **Insert > Call to Existing Action** and locating the test with which the original action is stored. For more information, see “[Inserting Calls to Existing Actions](#)” on page 352.

To remove an external action from the test:

- 1** Select the action you want to remove from the Action List.
- 2** Highlight the action icon in the test.
- 3** Choose **Edit > Delete** or press the **Delete** key on your keyboard. The Delete Action dialog box opens.



- 4** Choose **Delete the action and its calls from this test** and click **OK**. The action and all calls to the action are removed from the test. The reusable action in the original test is not affected.

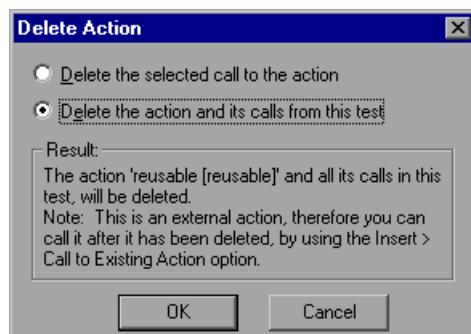
Removing a Reusable Action from the Test

You should choose to remove a reusable action from the test only if you are sure that you no longer need the action and that no other test calls this action. This option deletes the action content entirely.

Note: Deleting a reusable action that is called by other tests will cause those tests to fail.

To remove a reusable action from the test:

- 1 Select the action you want to remove from the Action List.
- 2 Highlight the action icon in the test.
- 3 Choose **Edit > Delete** or press the **Delete** key on your keyboard. The Delete Action dialog box opens.



- 4 Choose **Delete the action and its calls from this test** and click **OK**. The action is permanently deleted and can no longer be inserted in, or accessed by, any test. Any test calling this action will fail.

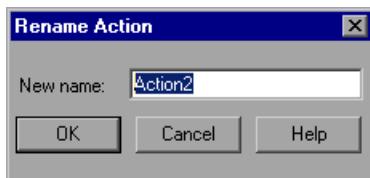
Renaming Actions

You can rename an action from the Keyword View or Expert View using the Action Properties dialog box or the Rename Action dialog box.

Note: 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 choose **Edit > Rename Action**. In the Expert View display the action that you want to rename and choose **Edit > Rename Action**. The Rename Action dialog box opens.

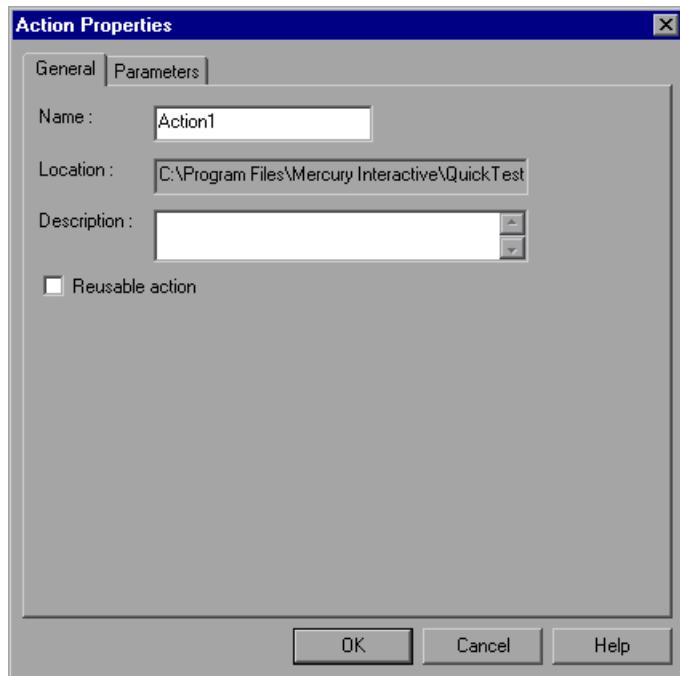


- 2 Enter a new name for the action in the **New name** box.
- 3 Click **OK** to save the change.

Tip: You can also press F2 to open the Rename Action dialog box.

To rename an action in the Action Properties dialog box:

- 1 In the Keyword View, select or right-click an action and choose **Step > Action Properties**. In the Expert View, choose **Step > Action Properties**. The Action Properties dialog box opens.



- 2 Enter a new action name in the **Name** box of the General tab.
- 3 Click **OK** to save the change.

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.

Guidelines for Working with Actions

Consider the following guidelines when working with actions:

- If your action runs more than one iteration, the action must ‘clean up after itself.’ In other words, 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 on. To parameterize the ‘book a flight’ action, you choose **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 choose **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 will run one time for each row in the global data sheet. Within each test, each parameterized action will be repeated depending on the number of rows in its data sheet and according to 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 > Rename Action**.

- 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**.
- When you insert a call to an external action, the action is inserted in read-only format, so the **Record** button is disabled. If you want to continue recording, you first need to insert a call to a local action into your test, or select a step from a local action that already exists in your test.
- Reusable actions help you to maintain your tests, but it is important to consider the effects of making an action reusable. Once you make an action reusable, be sure to consider how changes to that 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.
- 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 re-record the required steps, if necessary, after the application is modified.

Guidelines for Working with Action Parameters

Consider the following guidelines when working with action parameters:

- Input action parameter values can only be used within steps of the calling action.
- Output action parameter values can only be used within steps of the calling action, and only after the action has been called.
- To use an action parameter value from one action (or from the test) within the step of another action, you must pass the value from action to action down the test hierarchy to the action in which you want to use it (Test -> Action A -> Action B -> Action C -> Step 1).
- To use an action output value from one action within the step of another action, you must pass the value from action to action up the test hierarchy to the action in which you want to use it (Step 1-> Action C-> Action B -> Action A -> Test 1).
- You can only use an action parameter to parameterize a value if the value types are both the same.
- You can use an action output parameter value of any type retrieved from a called action, in subsequent steps of the calling action as a variable. 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).

18

Working with Data Tables

QuickTest enables you to insert and run steps that are driven by data stored in the Data Table.

This chapter describes:

- About Working with Data Tables
- Working with Global and Action Sheets
- Saving the Data Table
- Editing the Data Table
- Importing Data from a Database
- Using Formulas in the Data Table
- Using Data Table Scripting Methods

About Working with Data Tables

The data your test or component uses is stored in the *design-time* Data Table, which is displayed in the Data Table pane at the bottom of the screen while you insert and edit steps.

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 execute mathematical formulas within the cells. You can use the **DataTable**, **DTSheet** and **DTParam** utility objects to manipulate the data in any cell in the Data Table.

You can insert Data Table parameters and output values into your test or component. 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.

Note:  You can insert Data Table parameters and output values to components, but you can store the values only in the first row of the Data Table. Component iterations are defined for the business process test in Quality Center and are not affected by the Data Table. For more information, refer to the *Business Process Testing User's Guide*.

During the run session, QuickTest creates a *run-time* Data Table—a live version of the Data Table associated with your test or component. 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 or component. 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 581.

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 or component 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** menu. Alternatively you can add a **DataTable.Import** statement to the beginning of your test or component to import the run-time Data Table that was exported at the end of the previous run session. For more information on these methods, refer to the *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.

Note:  When working with components, the Data Table has only one type of sheet for the component.

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:

	Arrivals	B	C	D	E	F	G
1	San Francisco						
2	New York						
3	Paris						
4							
5							
6							
7							

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.

	Departure	B	C	D	E	F	G
1	New York						
2	Paris						
3	Los Angeles						
4							
5							
6							
7							

For more information about creating global and action parameters, see Chapter 12, “Parameterizing Values.”

Saving the Data Table

The Data Table contains the values that QuickTest substitutes for Data Table parameters when you run a test or component, as well as any other values or formulas you enter. Whenever you save your test or component, 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 tab of the Test Settings dialog box.

For more information on the Test Settings dialog box, see Chapter 25, “Setting Options for Individual Tests or Components.”

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 12, “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.

Notes:

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.

The external Data Table file may be locked by QuickTest, for example, if another user is working with the same external file when you open your test. If the file is locked, a message regarding locked resources opens. For more information, see “Creating, Opening, and Saving Tests with Locked Resources” on page 109.

 When working with components, the Data Table is always saved with your component, and cannot be saved in a different location.

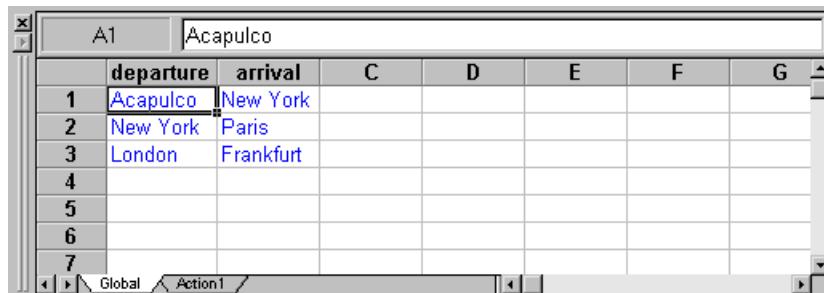
Editing the Data Table

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 in Microsoft Excel, tabbed text files (.txt), or ASCII format.

For information on supported versions of Microsoft Excel, refer to the *QuickTest Professional Readme*.



The Data Table pane is displayed when the **Data Table** button is enabled.



	A1	Acapulco					
	departure	arrival	C	D	E	F	G
1	Acapulco	New York					
2	New York	Paris					
3	London	Frankfurt					
4							
5							
6							
7							

Global Action1

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. QuickTest runs the iterations of your action based on the settings selected in the Run tab of the Action Properties dialog box. The number of iterations that a test runs is equal to the number of rows in the Global sheet.

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).

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.

Note: QuickTest underlines each row to which you add data in the Data Table. When you run your test using the **Run on all rows** option (**Test > Settings > Run tab**, or **Action Call Properties, Run tab**), it runs one iteration for each underlined row. If you use the **Clear** option from the table's Edit menu (or **CTRL+X**), or select a cell and press **Delete** on the keypad, the data is deleted from the cells, but the row is not deleted and the underline remains. This means that QuickTest will run an iteration for this row even though there is no data in it. If you want to delete an entire row from the Data Table, you must use the **Edit > Delete** option (**CTRL+K**).

Data Table Specifications

The main limitations for 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**—9.9999999999999E307
- **Largest negative number**— -9.9999999999999E307
- **Smallest positive number**—1E-307
- **Smallest negative number**— -1E-307
- **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)

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 or component. 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 or component.

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
- Sheet
- Edit
- Data
- Format

File Menu

The following commands are available in the File menu:

File Command	Description
Import From File	<p>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.</p> <p>Notes: The table file you import replaces all data in all sheets of the table, and the first row in each Microsoft 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, and that the file contains at least the same number of sheets as the current Data Table.</p> <p>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.</p>
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	<p>Creates, edits, or deletes 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, enter the first item into a cell in the table. Drag the cursor, from the bottom right corner of the cell, and QuickTest automatically fills in the cells in the range according to the autofill list.</p> <ul style="list-style-type: none"> • 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, where you can browse to a previously created list. • Save—Opens the Save As dialog box, where you can save a new list.
Encrypt	<p>Encodes the text in the selected cells. Note that you cannot decrypt data that has been encrypted.</p> <p>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 416.</p>

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 (%).
Fraction	Sets format to fraction in numerator denominator form, i.e. 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 on.

Note: Combo box and list cells, conditional formatting, and other special cell formats are not supported in the Data Table.

You can also execute Data Table menu commands using shortcut keys. For more information, see “Executing Commands Using Shortcut Keys” on page 22.

Using Data Table Files with Quality Center

When working with Quality Center and Data Tables, you must save the Data Table file as an attachment in your Quality Center project before you specify the Data Table file in the Resources tab of the Test Settings dialog box.

You can add a new or existing Data Table file to your Quality Center project. Note that if you add an existing Data Table from the file system to a Quality Center project, it will be a copy of the one used by tests not in the Quality Center project, and thus once you save the file to the project, changes made to the Quality Center repository 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 a .xls extension.
- 2** In Quality Center, add the Data Table file to the project as an attachment.
- 3** In the Test Settings dialog box, click the **Resources** tab.
- 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

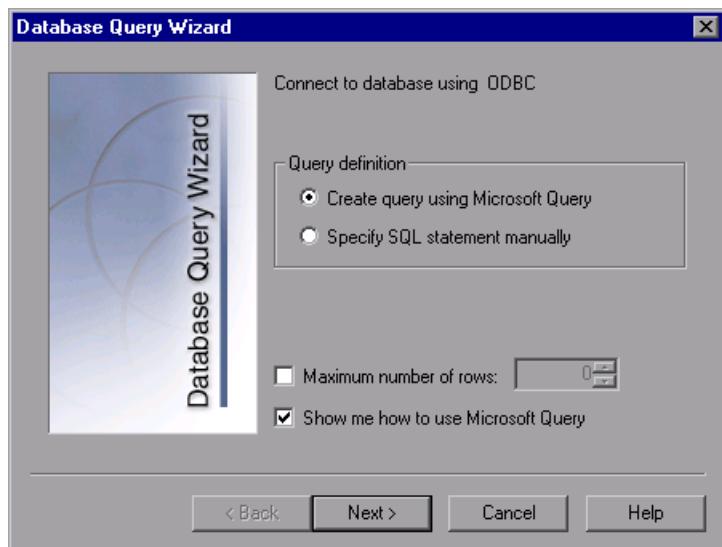
You can import data from a database by selecting a query from Microsoft Query or by manually specifying an SQL statement.

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.

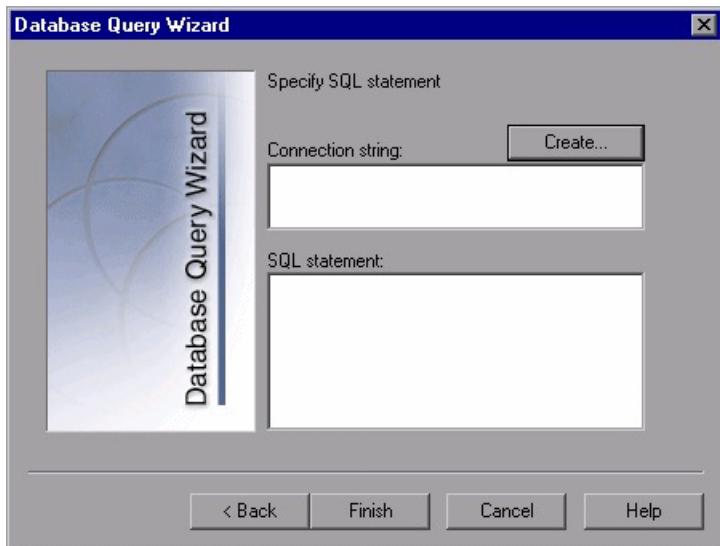
To import data from a database:

- 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. Once 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 additional 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 about creating a query, see “Creating a Query in Microsoft Query,” below.

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, refer to the *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, choose **File > Exit and return to QuickTest** to close Microsoft Query and return to QuickTest.

For additional information on working with Microsoft Query, refer to the Microsoft Query documentation.

Using Formulas in the Data Table

You can use any Microsoft Excel formula 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, i.e. integers, strings, etc. 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".

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 additional information on using worksheet functions, refer to 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 rest of the rows in order 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.

A2	=A1+1
	Date
1	3/28/2000
2	3/29/2000
3	3/30/2000
4	3/31/2000
5	4/1/2000
6	4/2/2000
7	4/3/2000
8	

For more information about using parameters, see Chapter 12, “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.

A1	=:\$B1=337	
	Total_Price	Total_Price_out
1	TRUE	337
2		

A FALSE result in the checkpoint column during a test run causes the test to fail.

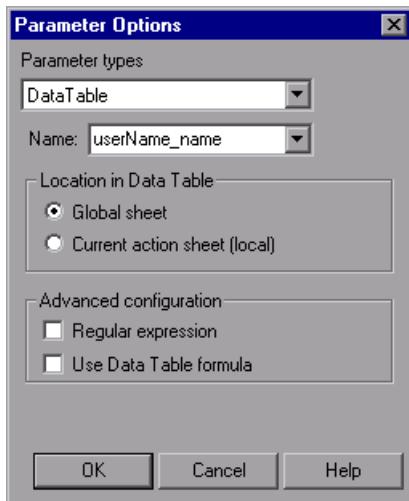
Once you finish adding the checkpoint, you can modify the default formula in the first column to perform the check you need.

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 6, “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 6, “Understanding Checkpoints.”

- 7 Click **OK**. The two columns are added to the table, and the checkpoint step is inserted into your test or component.
- 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.

Inserting Encoded Passwords into Method Arguments and Data Table Cells

You can encode passwords in order 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 406.

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.

Using Data Table Scripting Methods

QuickTest provides several Data Table methods that enable you to retrieve information about 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 about working in the Expert View, see Chapter 36, “Working with the Expert View.”

From a programming perspective, the Data Table is made up of three types of objects—`DataTable`, `DTSheet` (sheet), and `DTPParameter` (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, refer to the *QuickTest Professional Object Model Reference*.

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 describes:

- About Defining and Using Recovery Scenarios
- Deciding When to Use Recovery Scenarios
- Defining Recovery Scenarios
- Managing Recovery Scenarios
- Setting the Recovery Scenarios List for Your Tests or Components
- Programmatically Controlling the Recovery Mechanism

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 running tests or components unattended—the test or component is suspended until you perform the operation needed to recover. For information on when to use recovery scenarios, see “Deciding When to Use Recovery Scenarios” on page 421.

The Recovery Scenario Manager provides a wizard that guides you through the process of defining a *recovery scenario*—a definition of an unexpected event and the operation(s) 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 or component.

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 Operation(s)**—The operation(s) that need to be performed in order to continue running the test or component. 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 once the recovery operations have been performed, and from which point in the test or component QuickTest should continue, if at all. For example, you may want to restart a test or component from the beginning, or skip a step entirely and continue with the next step in the test or component.

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 it with that test or component. A test or component can have any number of recovery scenarios associated with it. You can prioritize the scenarios associated with your test or component to ensure that trigger events are recognized and handled in the required order. For more information, see “Adding Recovery Scenarios to Your Test or Component” on page 457.

When you run a test or component for which you have defined recovery scenarios and an error occurs, QuickTest looks for the defined trigger event(s) 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 or component. For more information, see “Programmatically Controlling the Recovery Mechanism” on page 462.

Note: If you choose **On error** in the **Activate recovery scenarios** box in the Recovery tab of the Test Settings dialog box or Business Component Settings dialog box, the recovery mechanism does not handle triggers that occur in the last step of a test or component. 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 or component, you can do this by adding an extra step to the end of your test or component.

Deciding When to Use Recovery Scenarios

If you can predict that a certain event may happen at a specific point in your test or component, it is highly recommended to handle that event directly within your test or component 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 operations are activated only occur after a step returns an error, which can potentially occur several steps after the one that actually caused the error. The alternative, checking for trigger events after every step, may slow performance.

You should use recovery scenarios only for unpredictable events, or events that you cannot synchronize with a specific step in your test or component. For example, a recovery scenario can handle a printer error by clicking the default button in the Printer Error message box. You cannot handle this error directly in your test or component, since you cannot know at what point the network will return the printer error. You could try to handle this event in your test or component by adding an **If** statement immediately after the step that sent a file to the printer, but if the network takes time to return the printer error, your test or component may have progressed several steps before the error is displayed. Therefore, for this type of event, only a recovery scenario can handle it.

For more information on optional steps, see “Using Optional Steps” on page 525. For more information on inserting programming statements such as **If** statements, see Chapter 20, “Adding Steps Containing Programming Logic.”

Defining Recovery Scenarios

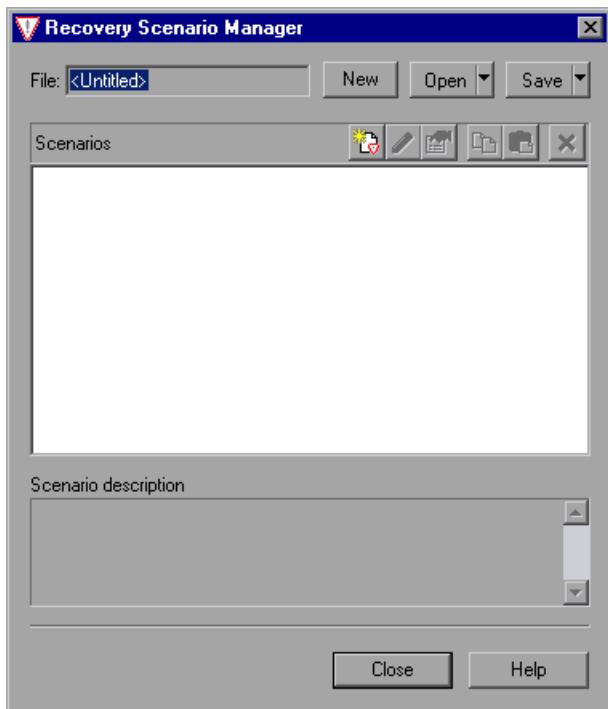
The Recovery Scenario Manager dialog box enables you to create recovery scenarios and save them in recovery files. You create recovery scenarios using the Recovery Scenario Wizard, which leads you through the process of defining each of the stages of the recovery scenario. You then save the recovery scenarios in a recovery file, and associate them with specific tests or components.

Creating a Recovery File

You save your recovery scenarios in a recovery file. A recovery file is a convenient way to organize and store multiple recovery scenarios together. You can create a new recovery file or edit an existing one.

To create a recovery file:

- 1 Choose **Tools > 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 either use this new file, or click the **Open** button to choose an existing recovery file. Alternatively, you can click the arrow next to the **Open** button to select a recently-used recovery file from the list.

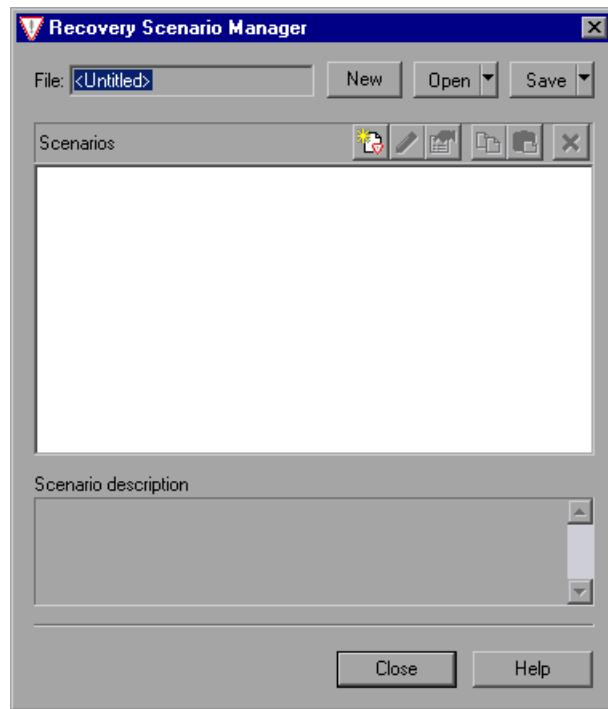
Open ▾

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 recovery scenarios.

The Recovery Scenario Manager dialog box displays the name of the currently open recovery file, a list of the scenario(s) saved in the recovery file, and a description of each scenario.



The Recovery Scenario Manager dialog box contains the following toolbar buttons:

Option	Description
 New	Creates a new recovery file. For more information, see “Creating a Recovery File” on page 423.
 Open ▾	Opens an existing recovery file. You can also click the arrow to select a recovery file from the list of recently-used recovery files.
 Save ▾	Saves the current recovery file. For more information, see “Saving the Recovery Scenario in a Recovery File” on page 450.
	Opens the Recovery Scenario Wizard, in which you define a new recovery scenario. For more information, see “Understanding the Recovery Scenario Wizard” on page 426.
	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 454.
	Displays summary properties for the selected recovery scenario in read-only format. For more information, see “Viewing Recovery Scenario Properties” on page 452.
	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 455.
	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 455.
	Deletes a recovery scenario. For more information, see “Deleting Recovery Scenarios” on page 454.

Note: Each recovery scenario is represented by an icon that indicates its type. For more information, see “Managing Recovery Scenarios” on page 451.

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 five main steps:

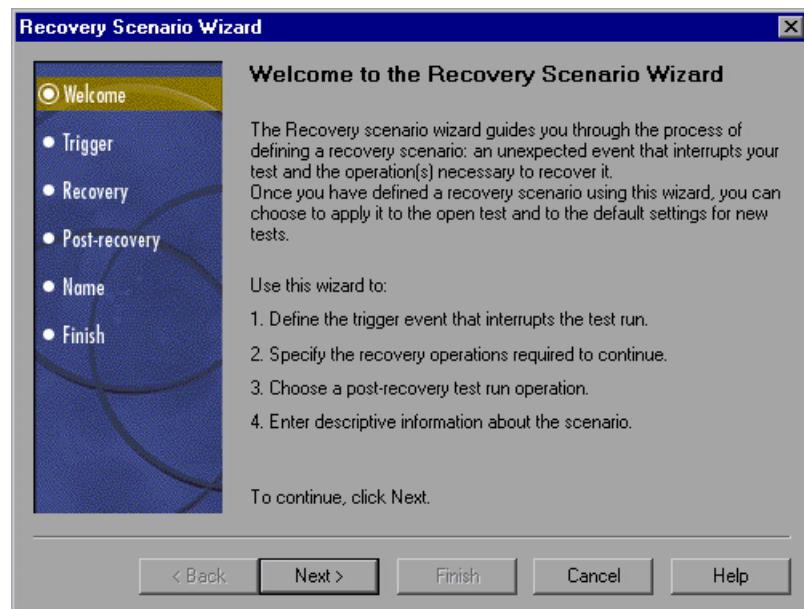
- defining the trigger event that interrupts the run session
- specifying the recovery operation(s) required to continue
- choosing a post-recovery test run operation
- specifying a name and description for the recovery scenario
-  specifying whether to associate the recovery scenario to the current 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.

Welcome to the Recovery Scenario Wizard Screen

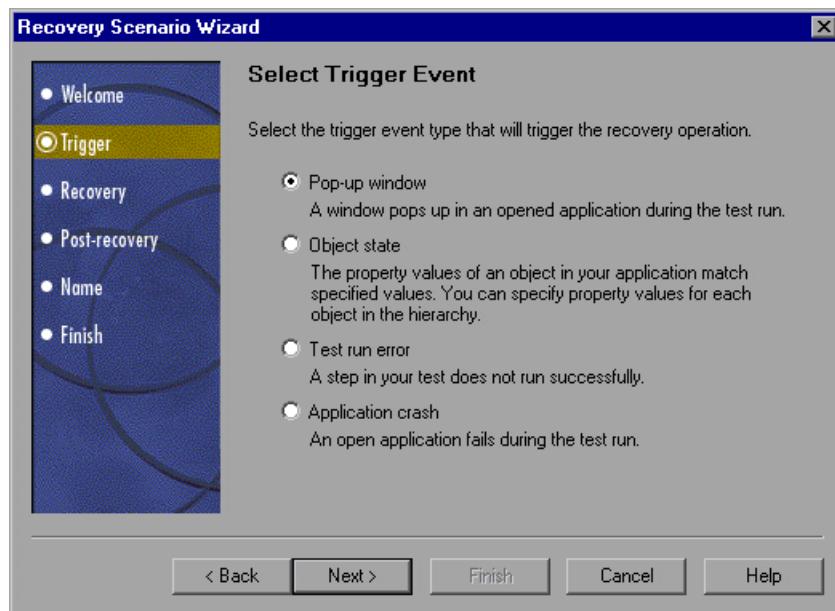
The Welcome to the Recovery Scenario Wizard screen provides general information about the different options in the Recovery Scenario Wizard, and provides an overview of the stages involved in defining a recovery scenario.



Click **Next** to continue to the Select Trigger Event screen.

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 in order to continue the run session.

Select this option and click **Next** to continue to the Specify Pop-up Window Conditions screen.

- **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.

- **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 in order to continue the run session.

Select this option and click **Next** to continue to the Select Test Run Error screen.

- **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.

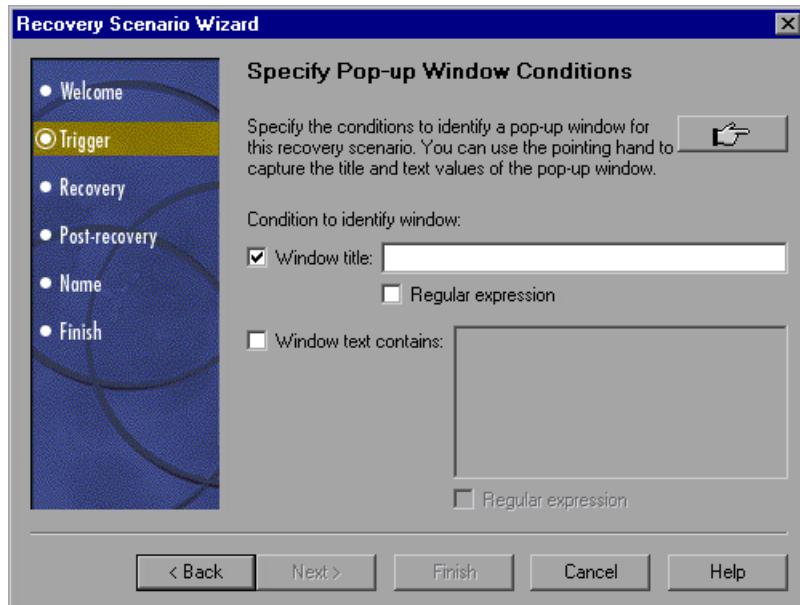
Select this option and click **Next** to continue to the Recovery Operations screen.

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 replay 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 or component. If you need to recover from an unexpected event or error that may occur in the last step of a test or component, you can do this by adding an extra step to the end of your test or component.

Specify Pop-up Window Conditions Screen

If you chose a **Pop-up window** trigger in the Select Trigger Event screen, the Specify Pop-up Window Conditions screen opens.



Click the pointing hand and then click the pop-up window to capture the window title and textual content of the window.

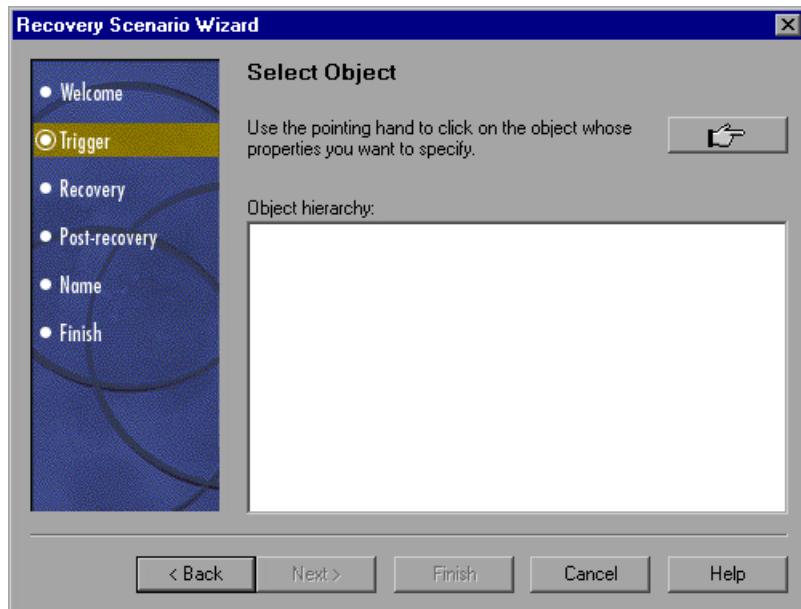
Tip: Hold the CTRL key to change the window focus or perform operations such as a right-click or mouseover to display a context menu. Note that pressing the CTRL key does not enable you to select an application from the Windows taskbar, therefore you must make sure that the window you want to access is not minimized.

You can choose whether you want to identify the pop-up window according to its **Window title** and/or **Window text**. You can also 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 290.

Click **Next** to continue to the Recovery Operations screen.

Select Object Screen

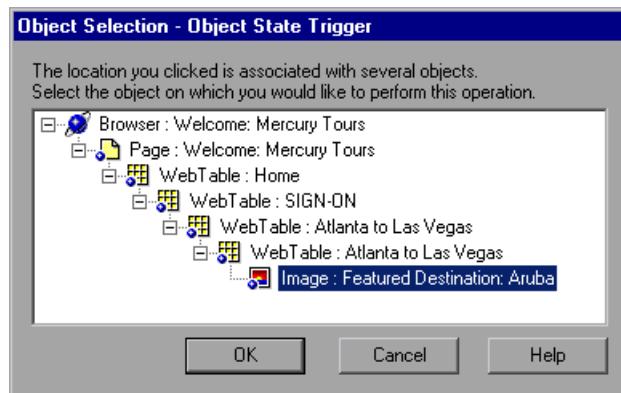
If you chose an **Object state** trigger in the Select Trigger Event screen, the Select Object screen opens.



Click the pointing hand and then click the object whose properties you want to specify.

Tip: Hold the CTRL key to change the window focus or perform operations such as a right-click or mouseover to display a context menu. Note that pressing the CTRL key does not enable you to select an application from the Windows taskbar, therefore you must make sure that the window you want to access is not minimized.

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**.

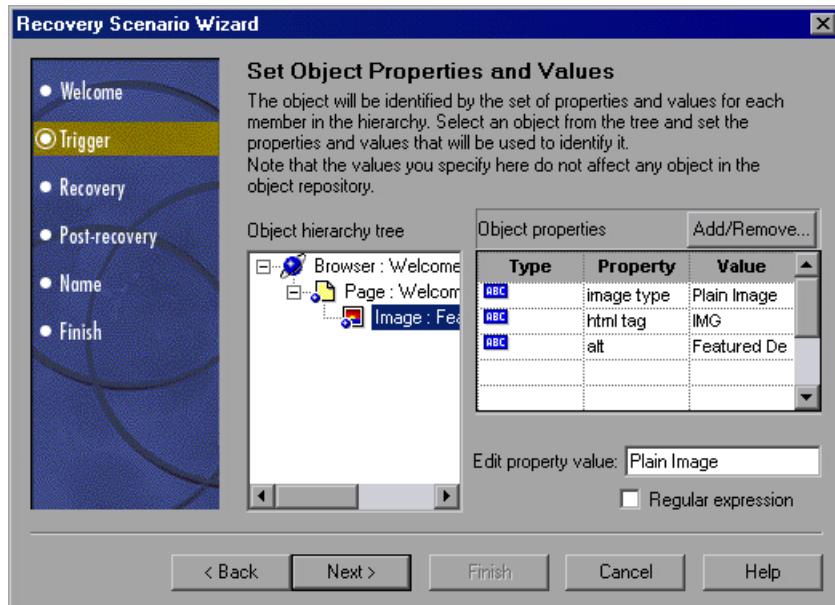
Note: The hierarchical object selection tree also enables you to select an object that QuickTest would not ordinarily record (a non-parent object), such as a web table.

The selected object and its parents are displayed in the Select Object screen.

Click **Next** to continue to the Set Object Properties and Values screen.

Set Object Properties and Values Screen

After you select the object whose properties you want to specify in the Select Object screen, the Set Object Properties and Values screen opens.



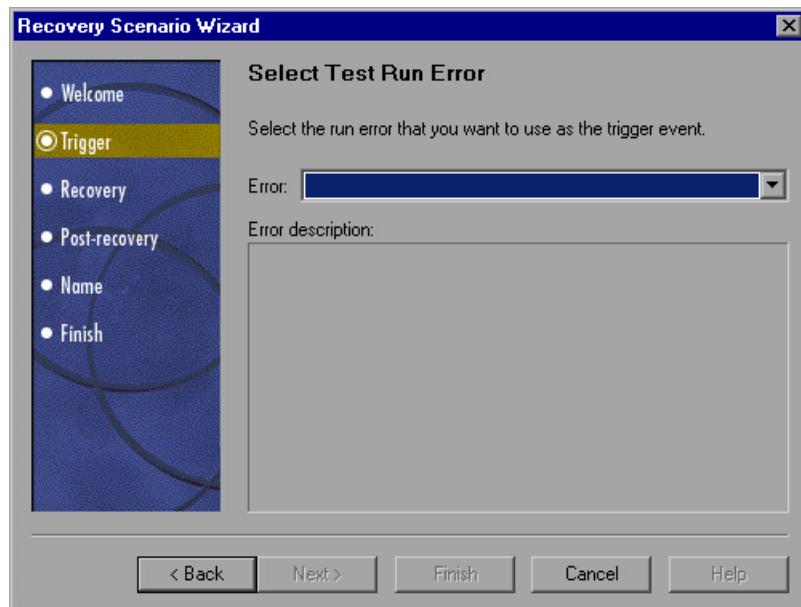
For each object in the hierarchy, in the **Edit property value** box, you can modify the property values used to identify the object. You can also click the **Add/Remove** button to add or remove object properties from the list of property values to check. Note that an object is identified only by its 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 290.

Click **Next** to continue to the Recovery Operations screen.

Select Test Run Error Screen

If you chose a **Test run error** trigger in the Select Trigger Event screen, the Select Test Run Error screen opens.



In the **Error** list, choose the run error that you want to use as the trigger event:

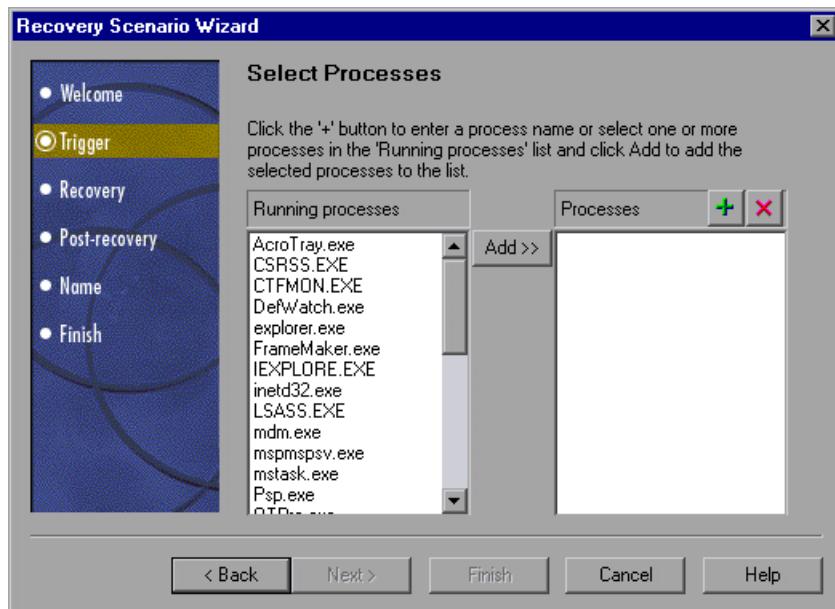
- **Any error**—Any error code that is returned by a step.
- **Item in list or menu is not unique**—Occurs when more than one item in the list, menu, or tree has the name specified in the method argument.
- **Item in list or menu not found**—Occurs when QuickTest cannot identify the list, menu, or tree 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.

Select Processes Screen

If you chose an **Application crash** trigger in the Select Trigger Event screen, 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.



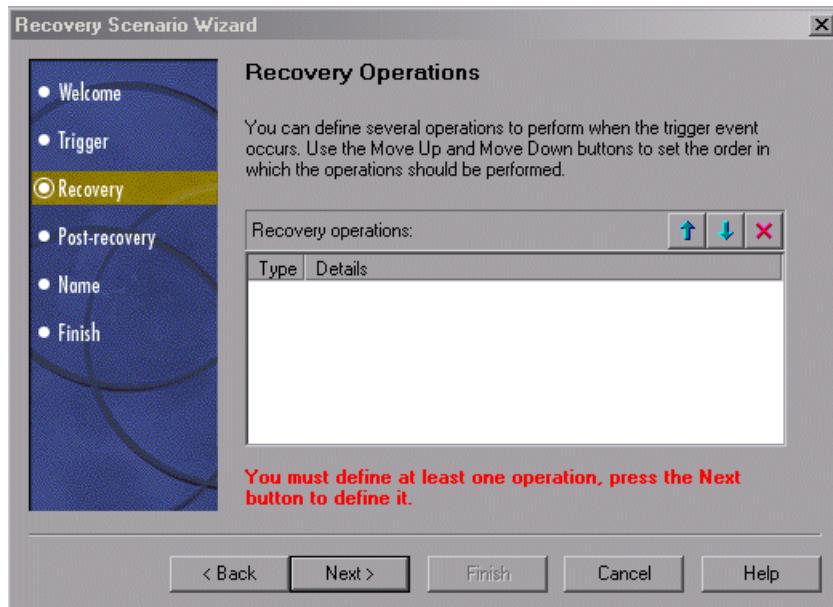
To remove a process from the **Processes** list, select it and click the **Remove Process** button.

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.

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.

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.

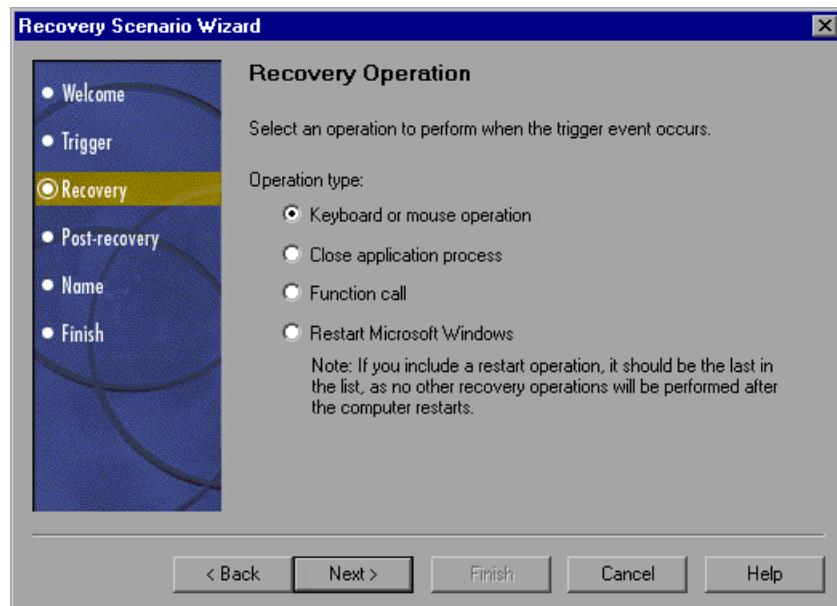
Note: If you define a **Restart Microsoft Windows** recovery operation, it is always inserted as the last recovery operation, and you cannot change its position in the list.

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.

Recovery Operation Screen

The Recovery Operation screen enables you to specify the operation(s) 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.

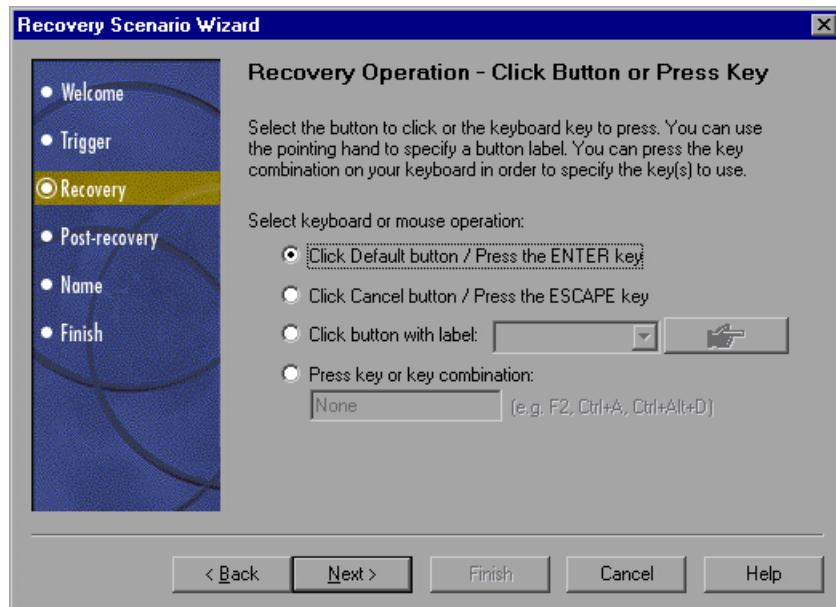
You can define the following types of recovery operations:

- **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.
- **Close application process**—QuickTest closes specified processes. Select this option and click **Next** to continue to the Recovery Operation – Close Processes screen.
- **Function call**—QuickTest calls a VBScript function. Select this option and click **Next** to continue to the Recovery Operation – Function screen.
- **Restart Microsoft Windows**—QuickTest restarts Microsoft Windows. Select this option and click **Next** to continue to the Recovery Operations screen.

Note: If you use the Restart Microsoft Windows recovery operation, you must ensure that any test or component associated with this recovery scenario is saved before you run it. You must also configure the computer on which the test or component is run to auto login 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, 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.

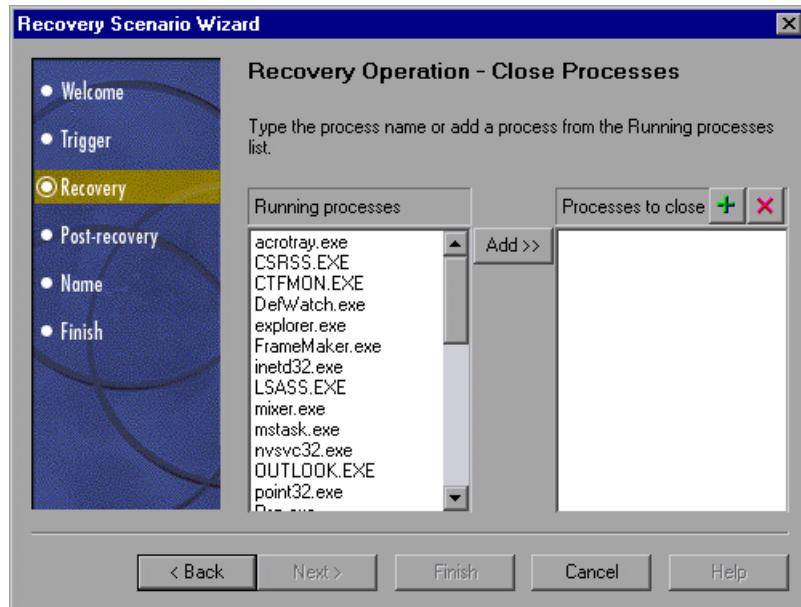
Tip: Hold the CTRL key to change the window focus or perform operations such as a right-click or mouseover to display a context menu. Note that pressing the CTRL key does not enable you to select an application from the Windows taskbar, therefore you must make sure that the window you want to access is not minimized.

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.

Recovery Operation – Close Processes Screen

If you chose a **Close application process** recovery operation in the Recovery Operation screen, 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.

You can add application processes to the **Processes to close** list by typing them in the **Processes to close** 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 to close** list, click the **Add New Process** button to enter the name of any process you want to add to the list.



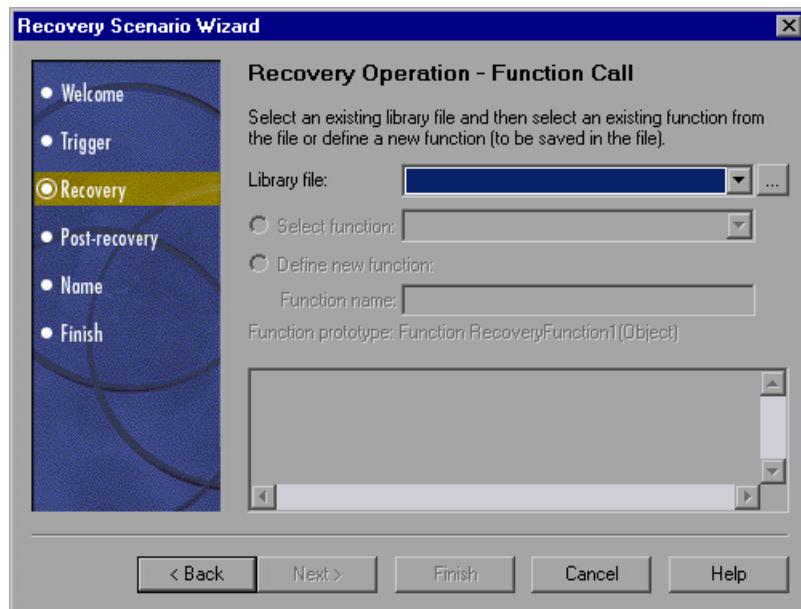
To remove a process from the **Processes to close** list, select it and click the **Remove Process** button.

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, the Recovery Operation – Function Call screen opens.



Select a recently specified library file in the **Library file** box. Alternatively, click the browse button to navigate to an existing library file.

Note: QuickTest automatically associates the library file you select with your test or component. Therefore, you do not need to associate the library file with your test or component in the Resources tab of the Test Settings dialog box or Business Component Settings dialog box, respectively.

After you select a library file, choose one of the following options:

- **Select function**—Choose an existing function from the library file you selected.
-

Note: Only functions that match the prototype syntax for the trigger type selected in the Select Trigger Event screen 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 library file you selected.
-

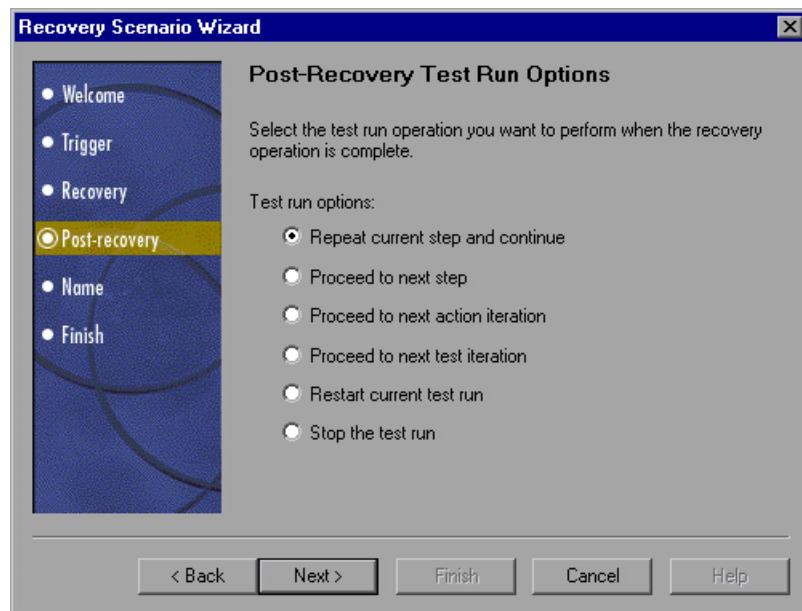
Note: If more than one scenario use a function with the same name from different library files, the recovery process may fail. In this case, information regarding the recovery failure is displayed during the run session.

Click **Next**. The Recovery Operations screen 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 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 460.

➤ **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 iteration**

Stops performing steps in the current action iteration and begins the next action iteration from the beginning (or the next action if no additional iterations of the current action are required).

►  **Proceed to next test iteration**

Stops performing steps in the current action and begins the next test iteration from the beginning (or stops running the test if no additional iterations of the current action are required).

► **Restart current test run**

Stops performing steps and re-runs the test or component from the beginning.

► **Stop the test run**

Stops running the test or component.

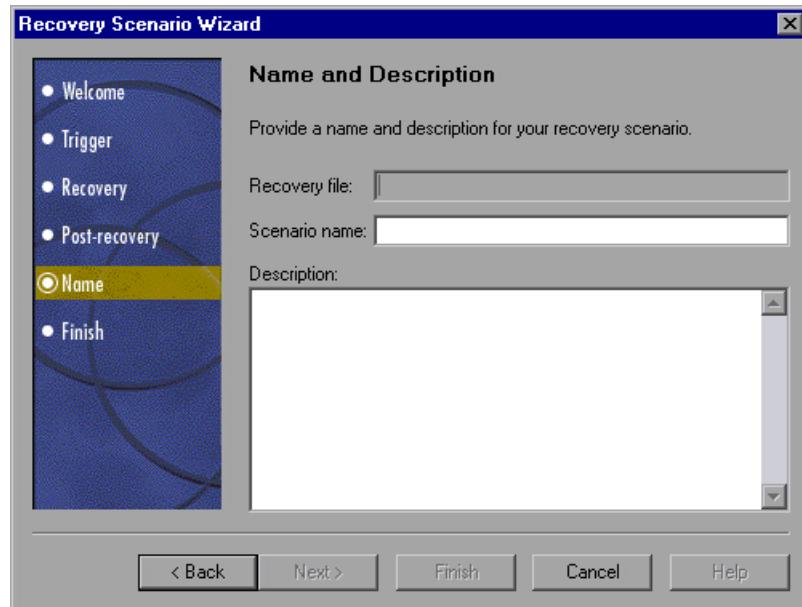
Note: If you chose **Restart Microsoft Windows** as a recovery operation, you can choose from only the last two test run options listed above.

Select a test run option and click **Next** to continue to the Name and Description screen.

Name and Description Screen

After you specify a test run option in the Post-Recovery Test Run Options screen, 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.

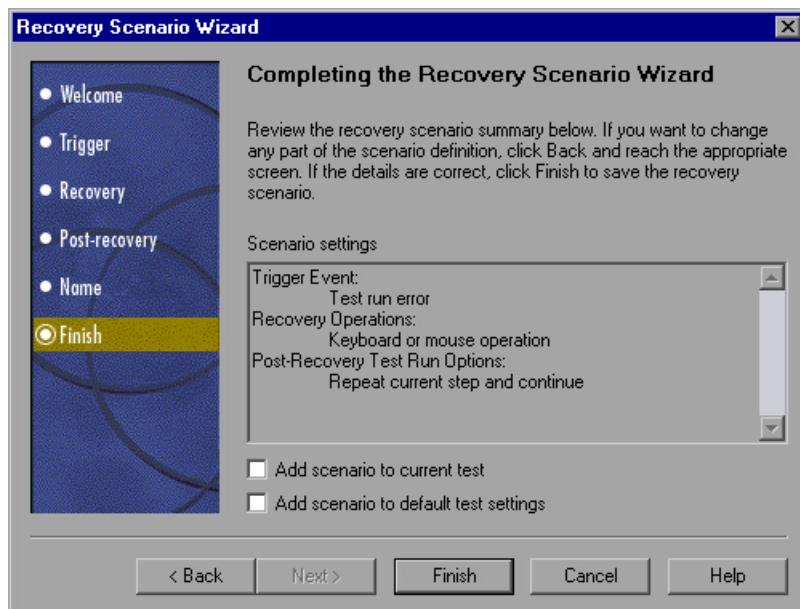


Enter a name and a textual description for your recovery scenario, and click **Next** to continue to the Completing the Recovery Scenario Wizard screen.

Completing the Recovery Scenario Wizard Screen

After you specify a recovery scenario name and description in the Name and Description screen 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 or component, and/or to add it to the default settings for all new tests.



Select the **Add scenario to current test** check box to associate this recovery scenario with the current test or component. When you click **Finish**, QuickTest adds the recovery scenario to the **Scenarios** list in the Recovery tab of the Test Settings dialog box or Business Component Settings dialog box, respectively.

 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 tab 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 or Component” on page 694.

Note:  You define the default recovery scenarios for all new components in the component template. For more information, see “Working with Component Templates” on page 987.

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.

To save a new or modified recovery file:



- 1 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 Attachment 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.

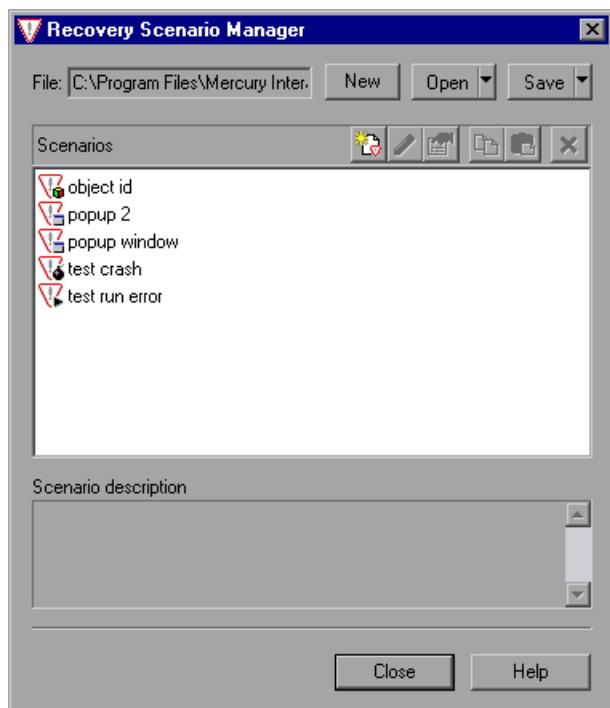
- 2 Choose the folder in which you want to save the file.

- 3 Type a name for the file in the **File name** box. The recovery file is saved in the specified location with the file extension **.qrs**.

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 above. 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.

Managing Recovery Scenarios

Once you have created 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 or component 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

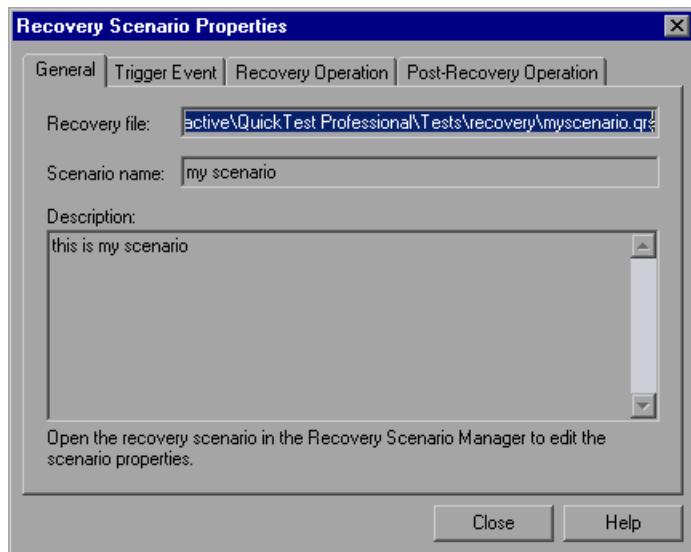
You can view properties for any defined recovery scenario.

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 operation(s) 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 422.

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 or component, 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.

Setting the Recovery Scenarios List for Your Tests or Components

After you have created recovery scenarios, you associate them with selected tests or components so that QuickTest will perform the appropriate scenario(s) 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 or component.

 You can also define which recovery scenarios will be used as the default scenarios for all new tests.

Note:  You define the default recovery scenarios for all new components in the component template. For more information, see “Working with Component Templates” on page 987.

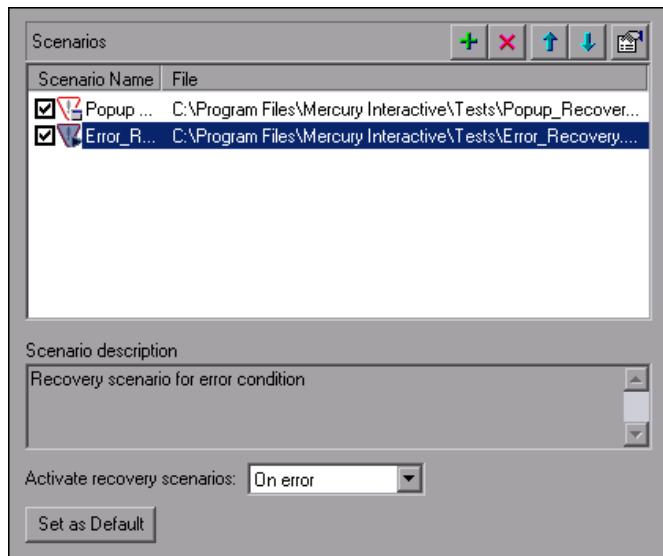
Adding Recovery Scenarios to Your Test or Component

After you have created recovery scenarios, you can associate one or more scenarios with a test or component in order to instruct QuickTest to perform the recovery scenario(s) during the run session if a trigger event occurs. The Recovery tab of the Test Settings dialog box or Business Component Settings dialog box lists all the recovery scenarios associated with the current test or component.

Tip: When a trigger event occurs, QuickTest checks for applicable recovery scenarios in the order in which they are displayed in the Recovery tab. You can change this order as described in “Setting Recovery Scenario Priorities” on page 459.

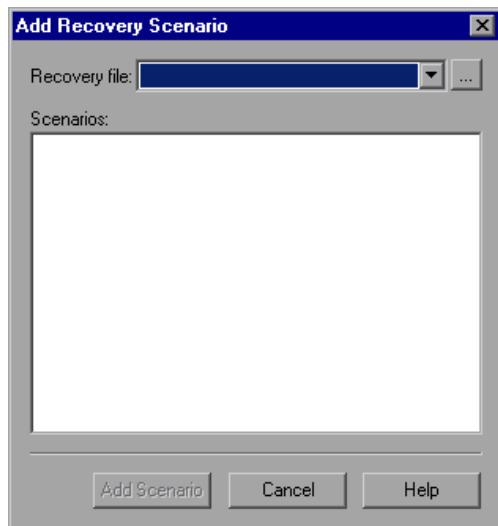
To add a recovery scenario to a test or component:

- 1 Choose **Test > Settings** or **Component > Settings**. The Test Settings dialog box or Business Component Settings dialog box opens. Select the Recovery tab.





- 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 scenario(s) you want to associate with the test or component. 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 scenario(s) that you want to associate with the test or component and click **Add Scenario**. The Add Recovery Scenario dialog box closes and the selected scenarios are added to the **Scenarios** list in the Recovery tab.

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 or component.

Viewing Recovery Scenario Properties

You can view properties for any recovery scenario associated with your test or component.

Note: You modify recovery scenario settings from the Recovery Scenario Manager dialog box. For more information, see “Modifying Recovery Scenarios” on page 454.

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 452.

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 tab of the Test Settings dialog box or Business Component 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.
 - 3 Repeat steps 1-2 for each scenario whose priority you want to change.

Removing Recovery Scenarios from Your Test or Component

You can remove the association between a specific scenario and a test or component using the Recovery tab of the Test Settings dialog box or Business Component Settings dialog box. After you remove a scenario from a test or component, 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 or component:

- 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 or component.



Enabling and Disabling Recovery Scenarios

You can enable or disable specific scenarios and determine when QuickTest activates the recovery scenario mechanism in the Recovery tab of the Test Settings dialog box or Business Component Settings dialog box. When you disable a specific scenario, it remains associated with the test or component, but is not performed by QuickTest during the run session. You can enable the scenario at a later time.

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.
 - **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.

Note: Choosing **On every step** may result in slower performance during the run session.

Tip: You can also enable or disable specific scenarios or all scenarios associated with a test or component programmatically during the run session. For more information, see “Programmatically Controlling the Recovery Mechanism” on page 462.

Setting Default Recovery Scenario Settings for All New Tests

 You can click the **Set as Default** button in the Recovery tab 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.

 You define the default recovery scenarios for all new components in the component template. For more information, see “Working with Component Templates” on page 987.

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, refer to the *QuickTest Professional Object Model Reference*.

20

Adding Steps Containing Programming Logic

After recording a test or component, 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 describes:

- About Adding Steps Containing Programming Logic
- Inserting Steps Using the Step Generator
- Using Conditional Statements
- Using Loop Statements
- Generating "With" Statements for Your Test or Component
- Sending Messages to Your Test Results
- Adding Comments
- Synchronizing Your Test or Component
- Measuring Transactions

About Adding Steps Containing Programming Logic

The easiest way to create a test or component is to begin by recording typical business processes that you perform on your application or Web site. Then, to increase the power and flexibility of your test or component, you can add steps that contain programming logic to the recorded framework. Programming statements can contain:

- *recordable* test object methods: operations that a user can perform on an application or Web site.
- *non-recordable* test object methods: operations that users cannot perform on an application or Web site. You use these methods to retrieve or set information, or to perform operations triggered by an event.
- run-time methods of the object being tested.
- various VBScript programming commands that affect the way the test or component runs, such as conditions and loops. These are often used to control the logical flow of a test or component.
- supplemental statements, such as comments, to make your test or component easier to read, and messages that appear in the test results, to alert you to a specified condition.

Note: *Test object* methods are defined in QuickTest; *run-time* methods are defined within the object you are testing, and therefore are retrieved from them.

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 quickly and easily add steps that use test object methods, utility object methods, 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 36, “Working with the Expert View.”

You can incorporate decision-making into your test or component and define messages for the test results by using the appropriate dialog boxes.

In addition, you can improve the readability of your test or component using **With** statements. You can instruct QuickTest to automatically generate **With** statements as you record. But even after your basic test or component 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 quickly and easily, 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 methods and properties
- ▶ utility object methods and properties
- ▶ calls to library functions, 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.

Before you open the Step Generator to define a new step, you first select where in your test or component the new step should be inserted. For more information on the hierarchy of steps and objects, see “Understanding the QuickTest Recorded Object Hierarchy” on page 311.

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. You can then select the appropriate method 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 or component. 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 or Expert View while recording or editing your test or component. You can also open the Step Generator from the Active Screen while editing.

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** Choose **Insert > Step > Step Generator** or right-click the step and choose **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 470.

To open the Step Generator from the Active Screen while editing:



- 1** Confirm that the Active Screen is displayed. If it is not, choose **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 choose **Step Generator**.

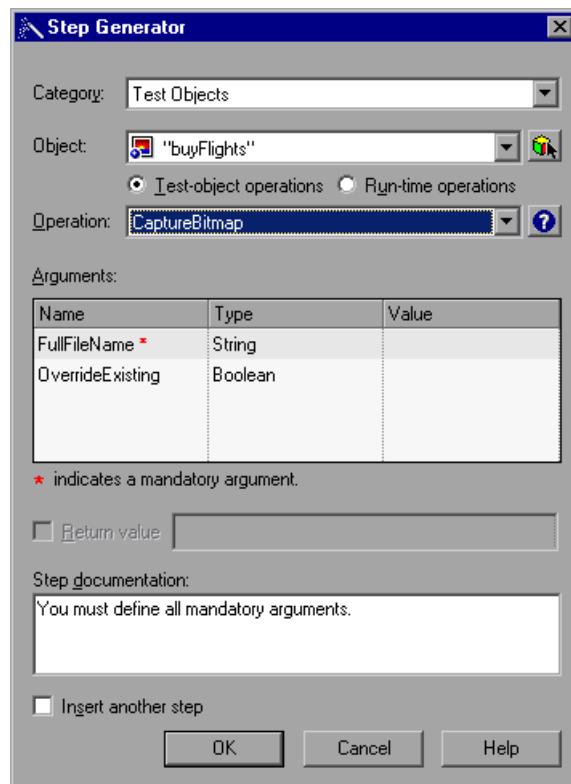
If the location you clicked is associated with more than one object, the **Object Selection - Step Generator** dialog box opens.



- 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,” below.

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, utility object methods, or function calls.



Note: The Step Generator dialog box that opens from the Expert View is similar to the dialog box that opens from the Keyword View (shown in the example above).

In the Expert View, the dialog box title is **Step Generator - Expert Mode** and the box title at the bottom of the dialog box is **Generated step**. For more information, see “Viewing the Generated Step in the Expert View” on page 473.

When the Step Generator dialog box opens, the object from the selected step is displayed in the **Object** 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 or component. You can then select the specific object and method 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 to your test or component.

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 or component; they do not modify the existing step.

Selecting the Type of Step to Add

In the **Category** list box, you can choose one of the following options:

- **Test Objects**—Enables you to select a test object and method for the step. For more information, see “Specifying a Test Object and Method for the Step” on page 474.
- **Utility Objects**—Enables you to select a utility object and method for the step. For more information, see “Specifying a Utility Object and Method for the Step” on page 479.
- **Functions**—Enables you to select a function for the step from one or all the available libraries. For more information, see “Specifying a Function for the Step” on page 481.

Specifying Argument Values

After you select the object and method or the function for the step, you can specify the relevant argument values. These values can be parameterized if required.

If the selected method or function 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 mouse 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 mouse 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 289.

Specifying the Location for the Return Value

If the selected method or function 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 482.

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 about 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 or component, 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 not available.

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.

Inserting Steps

After you define all mandatory argument values for the current operation, the following options are available:

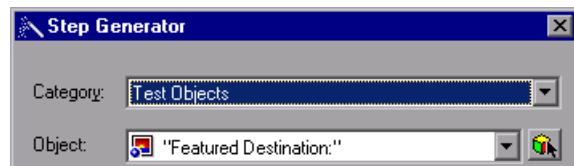
- To insert the current step and close the Step Generator, make sure the **Insert another step** check box is cleared. When you click **OK**, the step is added to your test or component and the Step Generator dialog box closes.
- To insert the current step and continue adding steps at the same location, 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 or component and the Step Generator dialog box remains open, enabling you to define another step.

Note: If you click **Insert** again before changing the step details, the Step Generator inserts another copy of the current step.

When you insert a new step using the Step Generator, it is added to your test or component 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 Recorded Object Hierarchy” on page 311.

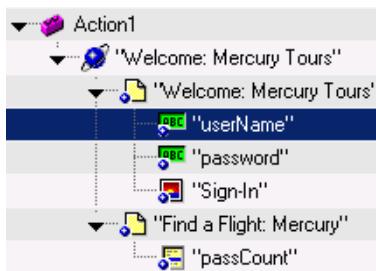
Specifying a Test Object and Method for the Step

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 or component. Alternatively, you can select any object from the object repository or from your application.

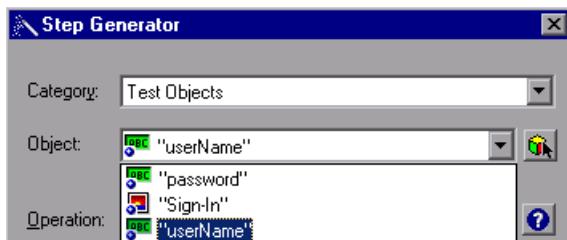


The list in the **Object** box contains all the objects in the object repository that are at the same hierarchical level and location as the currently selected step. You can select any of these objects for your new step.

For example, assume 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 in alphabetical order of their object names, and not according to the order of the steps in the test or component.



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 477.

After you select the object for the step, you can select the required operation type (test object method or, if available, run-time object method) and then you can select the method for the step.

Selecting the Method for a Test Object

If QuickTest can retrieve run-time methods for the selected test object, you can select the method type—**Test object method** or **Run-time object method**. (If QuickTest cannot retrieve run-time methods for the selected object, the **Run-time object method** option is not available.)

The **Operation** box displays the default method for the selected object. You can select a different method from the **Operation** box list, which contains all the methods that are available for the selected object.

For detailed information about a test object method and its syntax, you can click the **Operation Help** button to open the *QuickTest Professional Object Model Reference* for the selected method.

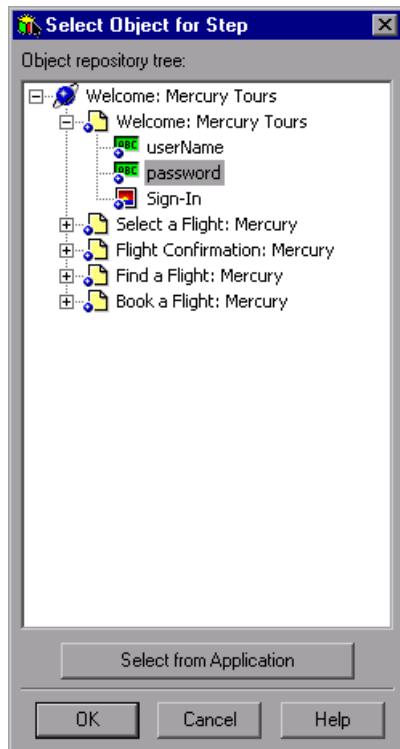
If you click the **Operation Help** button when a run-time method is selected, the *QuickTest Professional Object Model Reference* opens for the selected test object. For more information about specific run-time methods, refer to the documentation for the environment or application you are testing.

Note: If you select a run-time method, the Step Generator inserts a step using **.Object** syntax. For information on using the **Object** property, see “Accessing Run-Time Object Properties and Methods” on page 899.

After you select the method for your test object, you can define the relevant argument values. For more information, see “Specifying Argument Values” on page 472.

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 4, “Managing Test Objects.”

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.

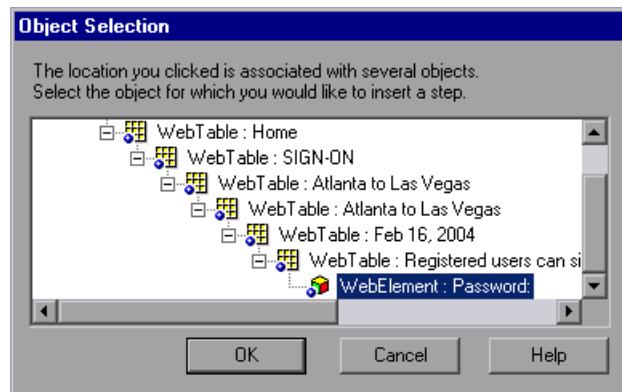
When you click **OK**, the selected object is displayed in the **Object** box in the Step Generator dialog box. You can now specify the method for the selected object. For more information, see “Selecting the Method for a Test Object” on page 476.

To select an object in your application for the new step:

- 1 Click the **Select from Application** button. QuickTest is minimized.
- 2 Use the pointing hand to click on the required object in your application.

Tip: You can hold the CTRL key to change the window focus or perform operations such as a right-click or mouseover to display a context menu. If the window containing the object you want to click is partially hidden by another window, you can also hold the pointing hand over the partially hidden window for a few seconds until the window comes into the foreground and you can point and click on the object you want. Note that pressing the CTRL key does not enable you to select an application from the Windows taskbar, therefore you must make sure that the window you want to access is not minimized.

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



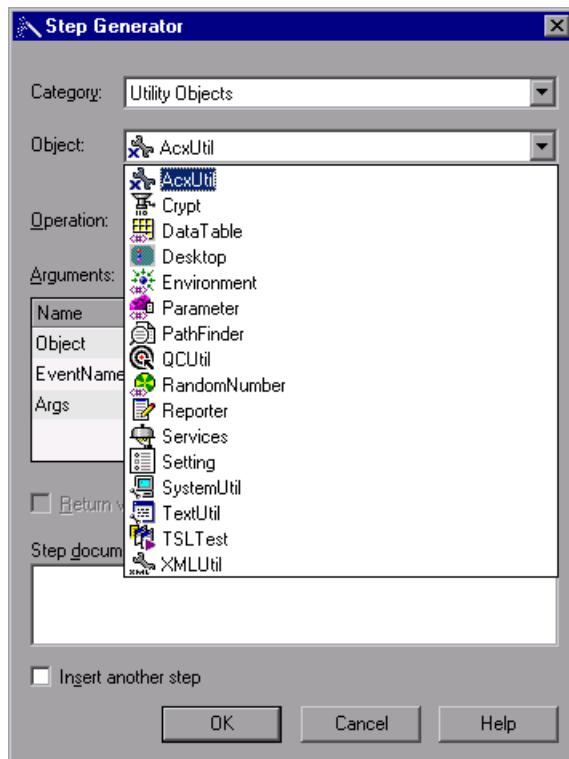
- 3 Select the object for the new step and click **OK**. The object is displayed in the **Object** box in the Step Generator dialog box.

You can now continue to specify the method for the selected object. For more information, see “Selecting the Method for a Test Object” on page 476.

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.

Specifying a Utility Object and Method 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, 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 more information about Utility objects, refer to the Utility Objects section of the *QuickTest Professional Object Model Reference*.

The **Operation** box displays the default method for the selected utility object. You can select a different method from the **Operation** box list, which contains all the methods that are available for the selected object.

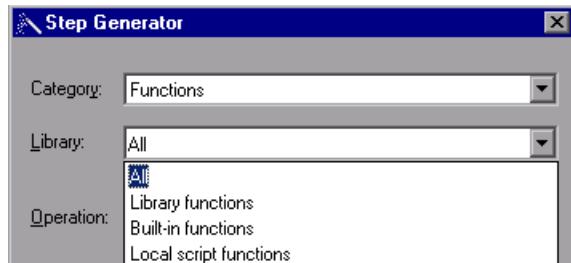
For detailed information about a utility object method and its syntax, you can click the **Operation Help** button to open the *QuickTest Professional Object Model Reference* for the selected method.



After you select the method for your utility object, you can define the relevant argument values. For more information, see “Specifying Argument Values” on page 472.

Specifying a Function for the Step

If you select **Functions** in the **Category** box list, you can choose 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 library file associated with your test or component. For more information about defining and using associated function libraries, see “Working with Associated Library Files” on page 908.
- **Built-in functions**—Enables you to select any standard VBScript function supported by QuickTest. For more information about 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 component.

You can select the required function from the **Operation** box list, which displays the functions available for the selected function type in alphabetical order.



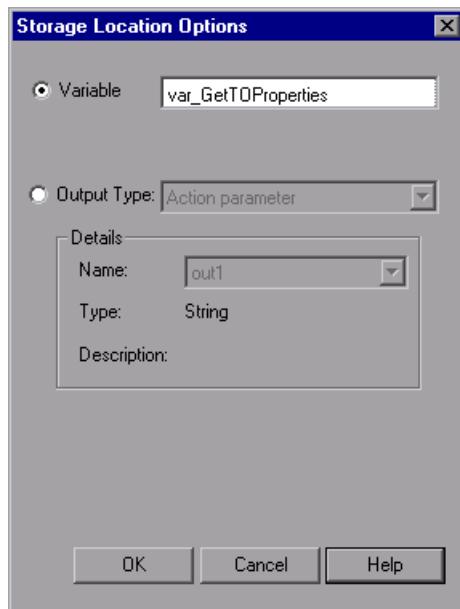
For detailed information about a selected built-in VBScript function, you can click the **Operation Help** button to open Microsoft’s VBScript Reference. This option is not available for library and local script functions.

After you select the function for the operation, you can define the relevant argument values. For more information, see “Specifying Argument Values” on page 472.

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 or enter a different variable name.
- **Output Type**—Stores the value in an 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 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 261.

For more information on default output definitions for output action parameter values, see “Understanding Default Output Definitions for Action Parameter Values,” below.

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 or Component Parameter” on page 262
- “Outputting a Value to the Data Table” on page 264
- “Outputting a Value to an Environment Variable” on page 266

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 **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 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 or component with conditional statements. Using conditional *If...Then...Else* statements, you can incorporate decision-making into your tests and components.

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 elseifstatements] End If

Or, you can use the block form syntax:

```
If condition Then
    [statements]
[ElseIf condition-n Then
    [elseifstatements] . .
[Else
    [elsestatements]
End If
```

Note: For additional information on conditional statements, refer to the VBScript documentation (choose **Help > QuickTest Professional Help > VBScript Reference**).

To insert a conditional statement:

- 1 In the Keyword View, select the step which you want the conditional statement to follow.
- 2 Choose **Insert > Step** and choose the statement type that you want to insert from the **Conditional Statement** 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
	If...Then statement
	ElseIf...Then statement
	Else statement

- 3 In the **Value** column, enter the required condition, for example:

Item_name = "New", CICount > 0

or

```
Browser("Welcome Mercury").Page("Welcome: Mercury").  
    WebEdit("Edit").Exist(5).
```

Tip: You can edit the value of a conditional statement at any time. You can change the condition in the Keyword View by selecting a different item in the dropdown list in the **Item** column.

- 4 To complete the **Then** part of the **If** statement, you can:

- Select the **If** statement step and record a new step to add it to your **Then** statement.
- Select the **If** statement step and choose **Insert > New Step** or press F8 to insert a new step into your **If** statement.
- Enter the statement manually in the Expert View.

Note: For more information on working in the Expert View, see Chapter 36, “Working with the Expert View.”

- 5 To nest an additional level to your statement, select the **If** statement in the Keyword View and choose one of the following options:

To add:	Choose:
an If statement	Insert > Step > Conditional Statement > If...Then
an Elseif statement	Insert > Step > Conditional Statement > Elseif...Then
an Else statement	Insert > Step > Conditional Statement > Else

To complete the new statement you can:

- Select the **If** statement step and record a new step to add it.
- Select the **If** statement step and choose **Insert > New Step** or press F8 to insert a new step into your **If** statement.
- Enter the statement manually in the Expert View.

For example, the test segment below (as it is displayed in the Keyword View) checks 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.

Item	Value	Documentation
Action1		Call the Action1 action.
Welcome: Mercury		
Welcome: Mercury		
password	"3ee357f628811830704e"	Enter the encrypted string "3ee357f628811830704...
IF		Check whether the "username" edit box exists. If so:
userName	"mercury"	Enter "mercury" in the "username" edit box.
ELSE		
Statement		
Reporter	1,"UserName Check","The ...	Report "The User Name field does not exist." to ...

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 "mercury"
    Else
        Reporter.ReportEvent micFail, "UserName Check", "The User Name field
        does not
        exist."
    End If
```

- 6 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 313.

Using Loop Statements

You can control the flow of your test or component 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 additional information on loop statements, refer to the VBScript documentation (choose **Help > QuickTest Professional Help > VBScript Reference**).

To insert a loop statement:

- 1 In the Keyword View, select the step which you want the loop statement to follow.

- 2** Choose **Insert > Step** and choose the statement type that you want to insert from the **Loop Statement** 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, or Browser("Welcome Mercury").Page("Welcome: Mercury").WebEdit("Edit").Exist(5).

Tip: You can edit the value of a loop statement at any time. For all loop types except **For...Next**, you can change the condition in the Keyword View by selecting a different item in the dropdown list in the **Item** column.

- 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 choose **Insert > New Step** or press F8 to insert a new step into your loop statement.
- Enter the statement manually in the Expert View.

Note: For more information on working in the Expert View, see Chapter 36, "Working with the Expert View."

For example, the test segment below (as it is displayed in the Keyword View) 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.

Item	Value	Documentation
Find a Flight:		
toDay	"items count"	Retrieve the current value of the "items count" property.
FOR Statement	For i = 1 To ItemsCount - 1	Repeat the following step(s) one or more times according to the loop condition.
toDay	i	Retrieve the value of the item with index i in the "toDay" list.
toDay	itemName	Select item itemName in the "toDay" list.

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 313.

Generating "With" Statements for Your Test or Component

You can instruct QuickTest to automatically generate **With** statements when you record a test or component 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 or component has no effect on the run session itself, only on the way your test or component appears in the Expert View. Generating **With** statements for your test or component 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")
```

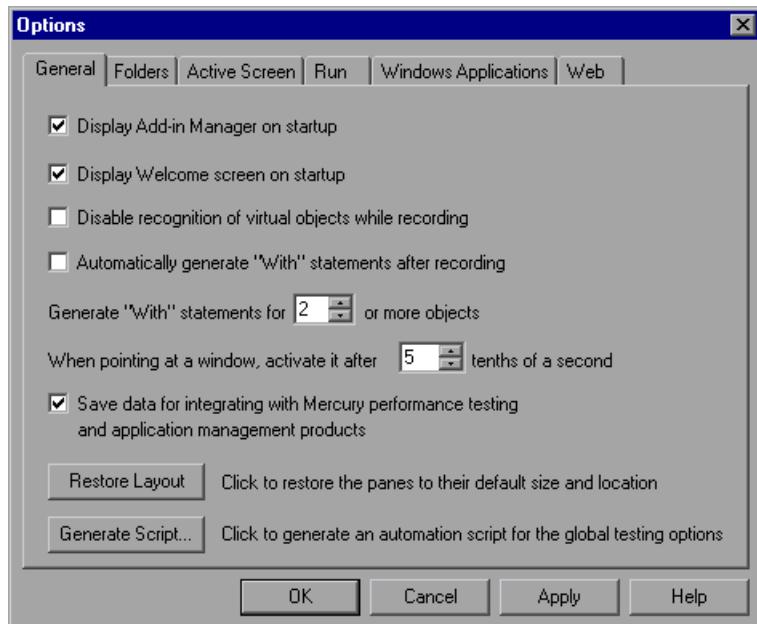
Automatically Generating "With" Statements

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 Choose **Tools > Options** or click the **Options** toolbar button. The Options dialog box opens.



- 2 In the **General** tab, 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,” below) 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.

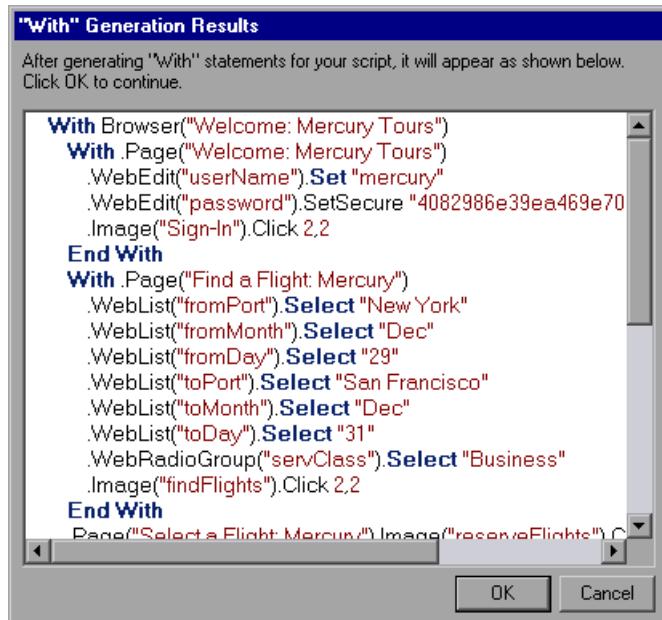
- 4 Begin recording your test or component. While recording, statements are recorded normally. When you stop recording, the statements in all actions recorded during the current recording session are automatically converted to the **With** format.

Generating “With” Statements for Existing Actions

You can instruct QuickTest to generate **With** statements for any action displayed in the Expert View.

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 tab 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, choose **Edit > Apply “With” to Script**. The “With” Generation Results window opens.



Note that each **With** statement contains only one object.

Note: 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 867.

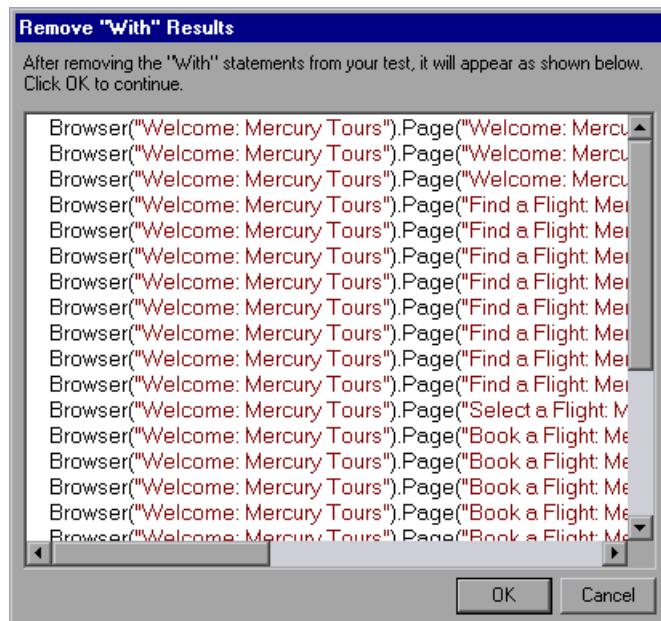
- 4 To confirm the generated results, click **OK**. The **With** statements are applied to the action.

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, choose **Edit > Remove “With” Statements**. The Remove “With” Results window opens.



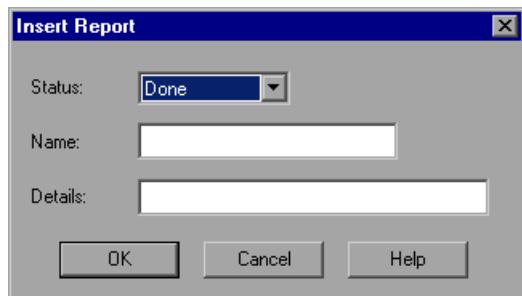
- 3 To confirm the results, click **OK**. The **With** statements are replaced with the standard statement format.

Sending Messages to Your 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 choose **Insert > Step > Report** or right-click a step and choose **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 or component 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 or component 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 **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.

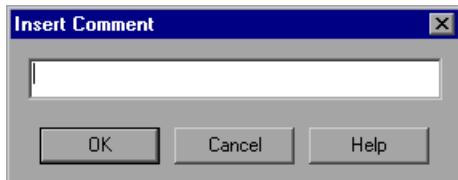
For more information on test results, see Chapter 23, “Analyzing Test Results”

Adding Comments

While programming, you can add comments to your test or component. A comment is an explanatory remark in a program. When you run a test or component, QuickTest does not process comments. You can use comments to explain sections of your tests and components to improve readability and to make them easier to update.

To add a comment in the Keyword View:

- 1 If the **Comment** column is not visible, right-click in any column header and select **Comment**.
- 2 To add a comment on the same line as a step, select the step and type your comment in the **Comment** column.
- 3 To add a comment on a separate line, select a step and choose **Insert > Step > Comment**, or right-click a step and choose **Insert Step > Comment**. The Insert Comment dialog box opens.



- 4 Type a comment and click **OK**. A comment statement is added to your test or component. In the Keyword View, the  icon indicates a comment.

In the Expert View, a comment is specified with an apostrophe ('). You can add comments in the Expert View by typing an apostrophe ('') at the end of a line or at the beginning of a separate line, followed by your comment.

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 391.

Synchronizing Your Test or Component

When you run a test or component, 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 or component 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 or component:

- You can insert a *synchronization point*, which instructs QuickTest to pause the test or component until an object property achieves the value you specify. When you insert a synchronization point into your test or component, 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 or component.
- 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 in order 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 or component 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 status 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 is 1.

Tip: QuickTest must be able to identify the specified object in order 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 503.

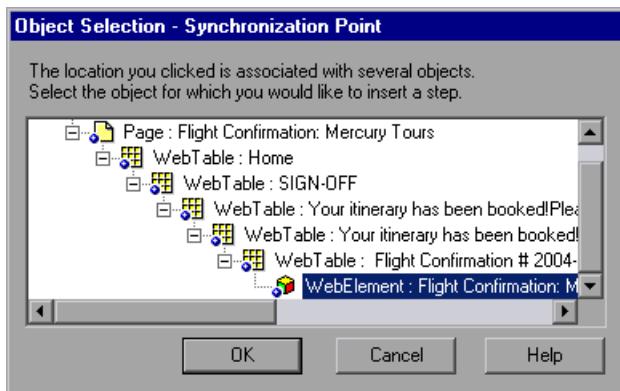
To insert a synchronization point:

- 1 Begin recording your test or component.
- 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, choose **Insert > Step > Synchronization Point**. The mouse pointer turns into a pointing hand.
- 4 Click the object in your application for which you want to insert a synchronization point.

Tip: You can also hold the CTRL key to change the window focus or perform operations such as a right-click or mouseover to display a context menu in order select the object you require. Note that pressing the CTRL key does not enable you to select an application from the Windows taskbar, therefore you must make sure that the window you want to access is not minimized.

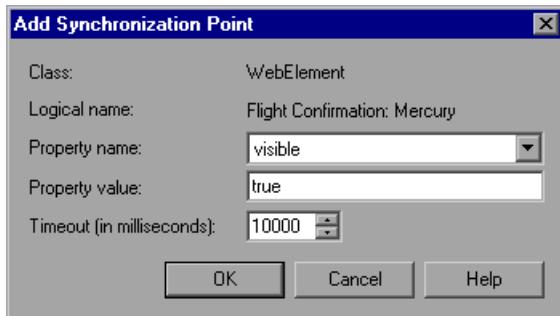
Note: It does not matter what property values the object has at the time that you insert the synchronization point.

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 test object 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 or component.
- 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 or component.

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
```

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 or component.

For example, the following statements instruct QuickTest to wait up to 20 seconds for the Flights Table dialog box to open.

```
bInDone=Window("Flight Reservation").Dialog("Flights Table").Exist  
counter=1  
While Not bInDone  
    Wait (2)  
    bInDone=Window("Flight Reservation").Dialog("Flights Table").Exist  
    counter=counter+1  
    If counter=10 then  
        bInDone=True  
    End if  
Wend
```

For more information on While, Exist, and Wait statements, refer to the *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 or component.

Alternatively, if you insert synchronization points and **Exist** and/or **Wait** statements for the specific areas in your test or component 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 or component.

-  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 **Test > Settings > Run** tab. For more information, see “Defining Run Settings for Your Test” on page 665.

Note:  The object synchronization timeout for a component is always 20 seconds (20000 milliseconds).

- To modify the amount of time that QuickTest waits for a Web page to load, change the **Browser Navigation Timeout** in the **Test > Settings > Web** tab or **Business Component > Settings > Web** tab. For more information, see “Defining Web Settings for Your Test or Component” on page 692.

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. 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.

A transaction can be inserted anywhere in your test, but must start and end within the same action. For more information about actions, see Chapter 17, “Working with Actions.”

Note:  Transactions can be defined only for tests. Components cannot include transactions.

During the test run, the Start Transaction signals the beginning of the time measurement. The time measurement continues until the End Transaction is encountered. The test report displays the time it took to perform the transaction.

Note: Your test must include transactions to be used by LoadRunner or the Business Process Monitor. LoadRunner and the Business Process Monitor use only the data that is included within a transaction, and ignore any data in a test outside of a transaction. For more information, see Chapter 42, “Working with Mercury Performance Testing and Application Management Products”.

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:

Start Transaction	 Services	StartTransaction	"ReserveSeat"	Start the "ReserveSeat" transaction.
	 Find a Flight: Mercury			
	 fromPort	Select	"London"	Select the "London" item in the "fromPort" list.
	 toPort	Select	"Frankfurt"	Select the "Frankfurt" item in the "toPort" list.
	 toDay	Select	"12"	Select the "12" item in the "toDay" list.
	 servClass	Select	"Business"	Select radio button "Business" in the "servClass" radio group.
	 airline	Select	"Blue Skies Airlines"	Select the "Blue Skies Airlines" item in the "airline" list.
	 findFlights	Click	65,12	Click the "findFlights" image.
	 Select a Flight: Mercury...			
	 outFlight	Select	"Blue Skies Airlines"	Select radio button "Blue Skies Airlines" in the "outFlight" radio group.
	 inFlight	Select	"Blue Skies Airlines"	Select radio button "Blue Skies Airlines" in the "inFlight" radio group.
	 reserveFlights	Click	46,8	Click the "reserveFlights" image.
End Transaction	 Services	EndTransaction	"ReserveSeat"	End the "ReserveSeat" transaction.
	 Book a Flight: Mercury_2			

The same part of the test is displayed in the Expert View as follows:

```

Services.StartTransaction "ReserveSeat"
Browser("Welcome: Mercury Tours").Page("Find a Flight: Mercury").
    WebList("fromPort").Select "London"
Browser("Welcome: Mercury Tours").Page("Find a Flight: Mercury").
    WebList("toPort").Select "Frankfurt"
Browser("Welcome: Mercury Tours").Page("Find a Flight: Mercury").
    WebList("toDay").Select "12"
Browser("Welcome: Mercury Tours").Page("Find a Flight: Mercury").
    WebRadioGroup("servClass").Select "Business"
Browser("Welcome: Mercury Tours").Page("Find a Flight: Mercury").
    WebList("airline").Select "Blue Skies Airlines"
Browser("Welcome: Mercury Tours").Page("Find a Flight: Mercury").
    Image("findFlights").Click 65,12
Browser("Welcome: Mercury Tours").Page("Select a Flight: Mercury").
    WebRadioGroup("outFlight").Select "Blue Skies Airlines"
Browser("Welcome: Mercury Tours").Page("Select a Flight: Mercury").
    WebRadioGroup("inFlight").Select "Blue Skies Airlines"
Browser("Welcome: Mercury Tours").Page("Select a Flight: Mercury").
    Image("reserveFlights").Click 46,8
Services.EndTransaction "ReserveSeat"

```

Inserting Transactions

During the test run, the Start Transaction signals the beginning of the time measurement. You define the beginning of a transaction in the Start Transaction dialog box.

To insert a transaction:

- 1 Click the step where you want the transaction timing to begin. The page is displayed in the Active Screen tab.
- 2  Click the **Start Transaction** button or choose **Insert > Start Transaction**. The **Start Transaction** dialog box opens.



- 3 Enter a meaningful name in the **Name** box.

Note: You cannot include spaces in a transaction name.

- 4 Decide where you want the transaction timing to begin:
 - To insert a transaction before the current step, select **Before current step**.
 - To insert a transaction after the current step, select **After current step**.
- 5 Click **OK**. A **Start Transaction** step is added to the Keyword View.

Ending Transactions

During the test run, the End Transaction signals the end of the time measurement. You define the end of a transaction in the End Transaction dialog box.

To end a transaction:

- 1 Click the step where you want the transaction timing to end. The page opens in the Active Screen.
- 2 Click the **End Transaction** button or choose **Insert > End Transaction**. The **End Transaction** dialog box opens.


- 3 The Name box contains a list of the transaction names you defined in the current test. Select the name of the transaction you want to end.
- 4 Decide where to insert the end of the transaction:
 - To insert a transaction before the current step, select **Before current step**.
 - To insert a transaction after the current step, select **After current step**.
- 5 Click **OK**. An **End Transaction** step is added to the Keyword View.

Part IV

Running and Debugging Tests and Components

21

Running Tests and Components

Once you have created a test or component, you can run it to check the behavior of your application.

This chapter describes:

- About Running Tests and Components
- Running Your Entire Test or Component
- Running Part of Your Test or Component
- Updating a Test or Component
- Using Optional Steps
- Running a Test Batch

About Running Tests and Components

When you run a test or component, QuickTest performs the steps it contains. If you have defined test or component parameters, QuickTest prompts you to enter values for them. When the test run is complete, QuickTest displays a report detailing the test results. For more information on viewing the test results, see Chapter 23, “Analyzing Test 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 additional information, see Chapter 25, “Setting Options for Individual Tests or Components,” and Chapter 17, “Working with Actions.”

You can run the entire test or component 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 or component to change the expected checkpoint values, and/or the Active Screen images and values.

You can run tests and components on objects with dynamic descriptions. For information, see Chapter 4, “Managing Test Objects.”

 You can set up a batch of tests and run them sequentially, using the QuickTest Test Batch Runner.

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 39, “Working with WinRunner.”

Running Your Entire Test or Component

QuickTest always runs a test or component from the first step, unless you specify otherwise. To run a test or component from a selected step, or a test from a selected action, you can use the **Run from Step** option. This feature is useful if you want to check a specific section of the test or component, without running it from the beginning. For more information, see “Running Part of Your Test or Component” on page 517.

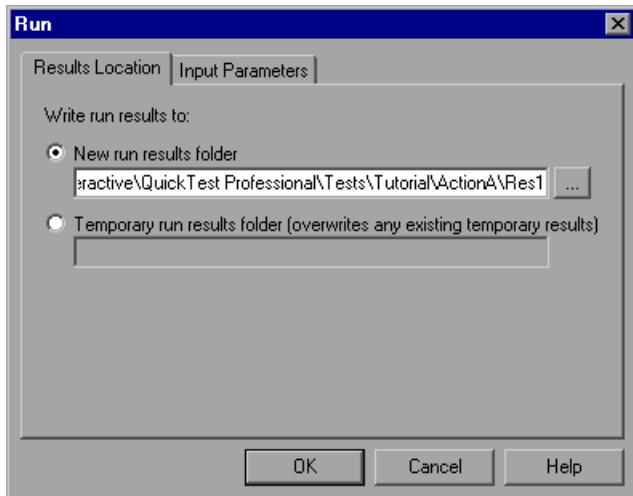
When you start to run a test or component, the Run dialog box opens, to enable you to specify the location for the results and to enter the values for any test or component parameters you have defined.

To run a test or component:

- 1 If your test or component is not already open, choose **File > Open** or click the **Open** button to open it.



- 2 Click the **Run** button on the toolbar, or choose **Test > Run** or **Component > Run**. The Run dialog box opens.



- 3 Specify the results location and the input parameter values (if applicable) for the run session. For more information, see “Understanding the Results Location Tab” on page 514, and “Understanding the Input Parameters Tab” on page 516.
- 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 test results, see Chapter 23, “Analyzing Test Results.”

Note: If you cleared the **View results when run session ends** check box in the Run tab 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 24, “Setting Global Testing Options.”

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



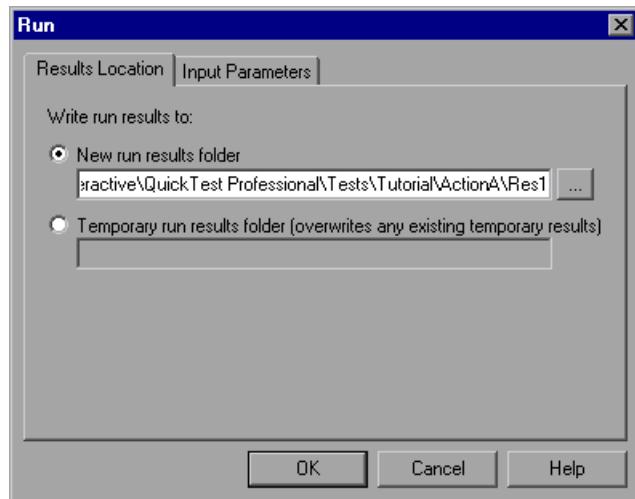
Click the **Pause** button in the Debug toolbar or choose **Debug > Pause**. The run pauses. To resume running a paused run session, click the **Run** button or choose **Test > Run** or **Component > Run**.



Click the **Stop** button or choose **Test > Stop** or **Component > Stop**. The run session stops and the Test Results window opens.

Understanding the Results Location Tab

The Results Location tab enables you to specify the location in which you want to save the test results.



Select one of the following options:

- **New run results folder**—This option displays the default path and folder name in which the test results are saved. By default, the results for a QuickTest test are stored in the test folder, and the results for components are stored in a Quality Center cache folder on your computer.

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.

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” on page 957.

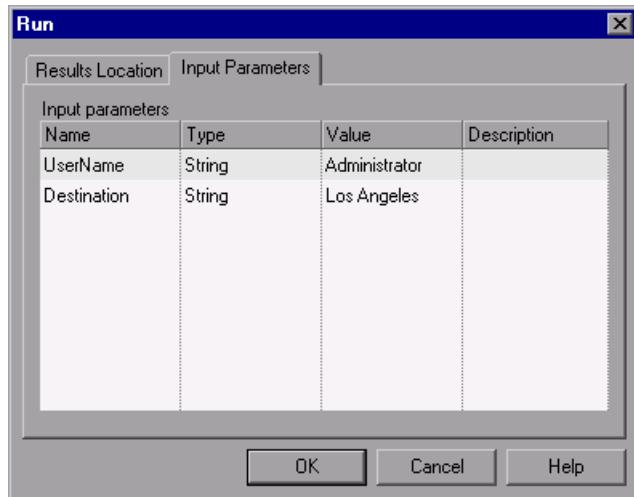
- **Temporary run results folder**—Saves the test run results in a temporary folder. This option overwrites any results previously saved in this folder.

Note: QuickTest stores temporary test results for all tests in <System Drive>\Documents and Settings\<user name>\Local Settings\Temp. The path in the text box of the **Temporary run results folder** option cannot be changed.

If you save test results to an existing test results folder, the contents of the folder are deleted when the run session starts.

Understanding the Input Parameters Tab

The Input Parameters tab enables you to specify the run-time values of input parameters to be used during the test run.



The Input Parameters tab displays the input parameters that were defined for the test (using the **Test > Settings > Parameters** tab) or for the component (using the **Business Component > Settings > Parameters** tab).

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 drop-down list. If you do not enter a value, QuickTest uses the default value from the Test Settings or Business Component Settings dialog box during the run session.

For more information on setting test or component parameters, see “Defining Parameters for Your Test or Component” on page 682. For more information about using parameters, see Chapter 12, “Parameterizing Values.”

Running Part of Your Test or Component

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

- You can run a component from a selected step to the end.
- You can run a test from a selected step to the end of the current action, if running from the Expert View, or to the end of the test, if running from the Keyword View:
 - Use the **Run from Step** option from the action view in the Expert View 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.
 - Use the **Run from Step** option from the Keyword View 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). 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.

Tips:

If you only want to run one iteration of your test, choose **Run one iteration only** from the Run tab 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 533.

For more information on actions, see Chapter 17, “Working with Actions.”

To run a test, action, or component from a selected step:

- 1** Open your application to the location matching the action or step you want to test.
- 2** Select the action or step where you want to start running the test or component:
 - In the Keyword View, highlight a step.
 - In the Expert View, place your cursor in a line of VBScript.
- 3** Choose **Test > Run from Step** or **Component > Run from Step**, or right-click and choose **Run from Step**.
- 4** In the Run dialog box, choose where to save the test results, and any input parameters you want to use, as described in “Understanding the Results Location Tab” on page 514, and “Understanding the Input Parameters Tab” on page 516.
- 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 test results, see Chapter 23, “Analyzing Test Results.”

The test results summary displays a note indicating that the test or component was run using the **Run from Step** option.

Note: If you cleared the **View results when run session ends** check box in the Run tab 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 24, “Setting Global Testing Options.”

Updating a Test or Component

When you update a test or component, QuickTest runs through the test or component to update the test object descriptions, the expected checkpoint values, and/or the Active Screen images and values. After you save the test or component, the updated data is used for subsequent runs.

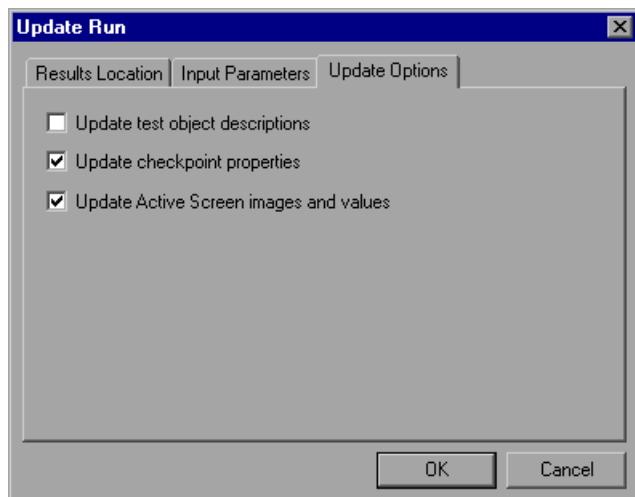
 When QuickTest updates tests, it runs through only one iteration of the test and one iteration of each action in the test. For information on actions, see Chapter 17, "Working with Actions."

Note: When QuickTest updates a test or component, it does not update parameterized values, such as Data Table data and environment variables. For information on parameterized values and environment variables, see Chapter 12, "Parameterizing Values."

To run a test or component to update the expected results:



- 1 If your test or component is not already open, choose **File > Open** or click the **Open** button to open it.
- 2 Choose **Test > Update Run** or **Component >Update Run**. The Update Run dialog box opens.



- 3 Specify the settings for the update run process. For more information, see “Understanding the Results Location Tab” on page 514, “Understanding the Input Parameters Tab” on page 516, and “Understanding the Update Options Tab” on page 521.
- 4 Click **OK**. The Update Run dialog box closes and QuickTest begins running the test or component update.

QuickTest runs the test or component 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 test results, see Chapter 23, “Analyzing Test Results.”

Note: If you cleared the **View results when run session ends** check box in the Run tab 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 24, “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:

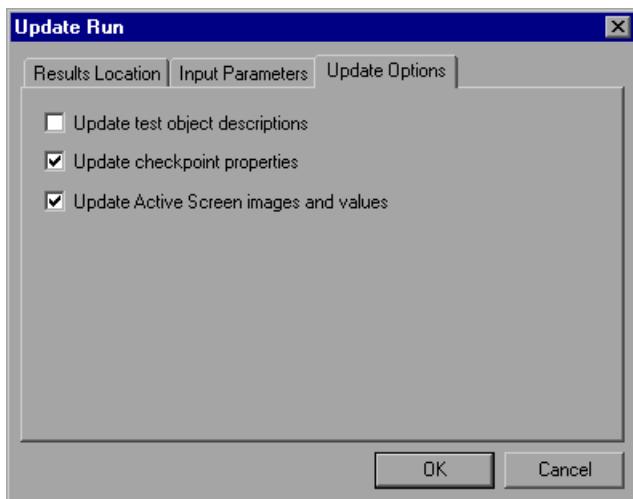
Step Name: Notifications:-Update Description

Step Done

Object	Details	Result	Time
Notifications:- Update Description	Test object's previous description: Text = Selection = Native Class = ListBox	Done	5/20/2003 - 10:43:11
	Test object's new description: Attached Text = Notifications:		

Understanding the Update Options Tab

The Update Options tab enables you to specify which aspects of your test or component you want to update: test object descriptions, expected checkpoint values, and/or Active Screen images and values. After you save the test or component, the results of the updated test or component 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 test object description according to the properties currently defined in the Object Identification dialog box for each object class. You can use this option to modify the set of properties used to identify an object.

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 test or component (or check it into a Quality Center project with version control support, if applicable) before updating it, so that you can return to the previously saved version if necessary.

This option can be especially useful when you want to record and debug your test or component using property values that are easy to recognize in your application (such as object labels), but may be language or operating system-dependent. Once you have debugged your test or component you can use the **Update Run** option to change the object descriptions to use more universal property values.

For example, suppose you designed a test for the English version of your application. The test objects are recognized according to the test object 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 non-language dependent properties to be used for the 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 to modify 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 also use the **Update test object descriptions** option to update the test object description property values.

When you run the test with **Update test object descriptions** selected, QuickTest finds the test object specified in each test 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 test run fails, and information about the failure is included in the test results.

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**, then if a Smart Identification definition enabled QuickTest to find the object, then **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**—QuickTest updates the expected checkpoint values to reflect any changes that may have occurred in your application since you recorded the test or component. For example, suppose you had defined a text checkpoint as part of your test, and the text in your application has changed since you recorded your test. You can update the test to update the checkpoint properties to reflect the new text.
-

Notes:

If you selected the **Save only selected area** check box when creating a bitmap checkpoint, the **Update Run** option 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. For more information, see “Checking Bitmaps” on page 169.

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 932. For more information on comment lines, see “Adding Comments” on page 498.

- **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 component 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 or component. You can update the test or component to update the dialog box appearance and its properties in the Active Screen.

Using Optional Steps

When running a test or component, if a step does not succeed in opening a dialog box, QuickTest does not necessarily abort the run session. It bypasses any step designated *optional* and continues the run. By default, QuickTest automatically marks as optional steps that open certain dialog boxes. You can manually designate additional steps as optional.

Note: An alternative method to bypassing dialog boxes is the use of recovery scenarios to click a button, press ENTER, or enter login information in a step. For more information, see Chapter 19, “Defining and Using Recovery Scenarios.”

Setting Optional Steps

When recording a test or component, the application you are testing may prompt you to enter a user name and password in a login window. When you run the test or component, however, the application does not prompt you to enter your user name and password, because it has retained the information that was previously entered. In this case, the steps that were recorded for entering the login information are not required and should, therefore, be marked optional.

When running a test or component, if a step in an optional dialog box does not open, QuickTest bypasses this step and continues to run the test. When the run session ends, a message is displayed for the step that failed to open the dialog box, but the step does not cause the test or component to fail.

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

Note: You can also add an optional step in the Expert View by adding **OptionalStep** to the beginning of the VBScript statement. For example:

```
OptionalStep.Browser("browser_name").Page("page_name").Link("link_name")
```

For information on working in the Expert View, see Chapter 36, “Working with the Expert View.” For information on the **OptionalStep** object, refer to the *QuickTest Professional Object Model Reference*.

Default Optional Steps

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

Dialog Box Titlebar
Auto Complete
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 Mercury products to run tests and components** in the Run tab of the Options dialog box. For more information, see Chapter 24, “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. Components cannot be used with Test Batch Runner.



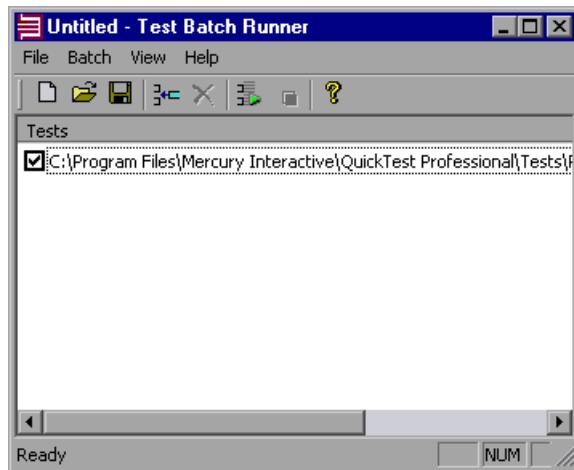
You can stop a test batch run at any time by clicking the **Stop** button.



To set up and run a test batch:

- 1** Choose **Programs > QuickTest Professional > Tools > Test Batch Runner** from the **Start** menu. The Test Batch Runner dialog box opens.
- 2** Click the **Add** button or choose **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 choose **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 choose **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 choose **Batch > Run**. If QuickTest is not already open, it opens and the tests run sequence begins. Once 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 about Test Results, see Chapter 23, “Analyzing Test Results.”

22

Debugging Tests and Components

By controlling and debugging your run sessions, you can identify and eliminate defects in your tests and components.

This chapter describes:

- About Debugging Tests and Components
- Using the Step Commands
- Pausing a Run Session
- Setting Breakpoints
- Removing Breakpoints
- Using the Debug Viewer
- Handling Run Errors
- Practicing Debugging a Test

About Debugging Tests and Components

After you create a test or component, you should check that it runs smoothly, without errors in syntax or logic. In order to detect and isolate defects in a test or component, you can use step commands and the **Pause** command to control the run session.

You can also do this by setting breakpoints. When the test or component stops at a breakpoint, you can use the Debug Viewer to check and modify the values of VBScript objects and variables. Also, if QuickTest displays a run-error message during a run session, you can click the **Debug** button on the error message to suspend the test or component and debug it.

Note: You can also use the **Run from Step** feature to debug your test or component. This runs your test or component from a selected step to the end. For additional information, see “Running Part of Your Test or Component” on page 517.

Using the Step Commands

You can run a single step of a test or component using the **Step Into**, **Step Out**, and **Step Over** commands.

Tip: To display the Debug toolbar, choose **View > Toolbars > Debug**.

Step Into



Choose **Debug > Step Into**, click the **Step Into** button, or press F11 to run only the current line of the active test or component. If the current line of the active test or component calls another action or a function, the called action/function is displayed in the QuickTest window, and the test or component pauses at the first line of the called action/function.

Step Out



Choose **Debug > Step Out** or click the **Step Out** button, or press SHIFT+F11 only after using **Step Into** to enter a action or a user-defined function. **Step Out** runs to the end of the called action or user-defined function, then returns to the calling action and pauses the run session.

Step Over



Choose **Debug > Step Over** or click the **Step Over** button, or press F10 to run only the current step in the active test or component. When 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 script is not displayed in the QuickTest window.

Using the Step Commands - An Example

Follow the instructions below to create a simple test and run it using the **Step Into**, **Step Out**, and **Step Over** commands.

To create the sample test:

- 1 Choose **File > New** to open a new test.
- 2 Click the **Expert View** tab to display the Expert View.
- 3 Enter the following lines exactly:

```
public Function myfunc()  
msgbox "one"  
msgbox "two"  
msgbox "three"  
End Function
```

```
myfunc  
myfunc  
myfunc
```

To run the test using the Step Into, Step Out, and Step Over commands:

- ➊ Press F9 (**Insert/Remove Breakpoint**) to add a breakpoint on the seventh line of the test (the first call to the **myfunc** function). The Expert View breakpoint symbol is displayed in the left margin. For more information, see “Setting Breakpoints” on page 533.
- ➋ Run the test. The test pauses at the breakpoint.
- ➌ Press F11 (**Step Into**). The execution arrow points to the first line within the function (msgbox "one").
- ➍ Press F11 (**Step Into**) again. A message box displays the text one.
- ➎ Click OK to close the message box. The execution arrow moves to the next line in the function.
- ➏ Continue pressing F11 (**Step Into**) until the execution arrow leaves the function and is pointing to the eighth line in the script (the second call to the **myfunc** function).
- ➐ Press F11 (**Step Into**) to enter the function again. The execution arrow points to the first **msgbox** line within the function.
- ➑ Press SHIFT+F11 (**Step Out**). Three message boxes open. 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 last line in the test.
- ➒ Press F10 (**Step Over**). The three message boxes open again. The execution arrow remains on the last line in the test.

Pausing a Run Session



You can temporarily suspend a run session by choosing **Debug > Pause** or clicking the **Pause** button. A paused test or component stops running when all previously interpreted steps have been run.



To resume running a paused test or component, click the **Run** button or choose **Test > Run** or **Component > Run**. The run continues from the point it was suspended.

Setting Breakpoints



By setting a breakpoint, you can stop a run session at a pre-determined place in a test or component. A breakpoint is indicated by a red hand icon in the left margin of the Keyword View, or a red circle icon in the left margin of the Expert View. QuickTest pauses the run when it reaches the breakpoint, before executing the step. You can examine the effects of the run up to the breakpoint, make any necessary changes, and then continue running the test or component from the breakpoint.

You can use breakpoints to:

- suspend a run session and inspect the state of your site or application
- mark a point from which to begin stepping through a test or component using the step commands

To set a breakpoint:



- 1 Click a step or a line in the test or component where you want the run to stop.
- 2 Choose **Debug > Insert/Remove Breakpoint**, press F9, or click the **Insert/Remove Breakpoint** button. The breakpoint symbol is displayed in the left margin of the Keyword View and Expert View.

Tip: You can also insert breakpoints by clicking in the left margin of the Keyword View. See “Working with Breakpoints in the Keyword View” on page 328.

Note: The breakpoints inserted are active only during the current QuickTest session. If you terminate your QuickTest session, you must insert new breakpoints to continue debugging the test or component in another session.

Removing Breakpoints

From the Debug menu you can remove a single breakpoint or all breakpoints defined for the current test or component.

- To remove a single breakpoint, click a line in your test or component with the breakpoint symbol and choose **Debug > Insert/Remove Breakpoint**, or click the **Insert/Remove Breakpoint** button.



The breakpoint symbol is removed from the left margin of the QuickTest window.

- To remove all breakpoints, choose **Debug > Clear All Breakpoints**, press F9, or click the **Clear All Breakpoints** button.



All breakpoint symbols are removed from the left margin of the QuickTest window.

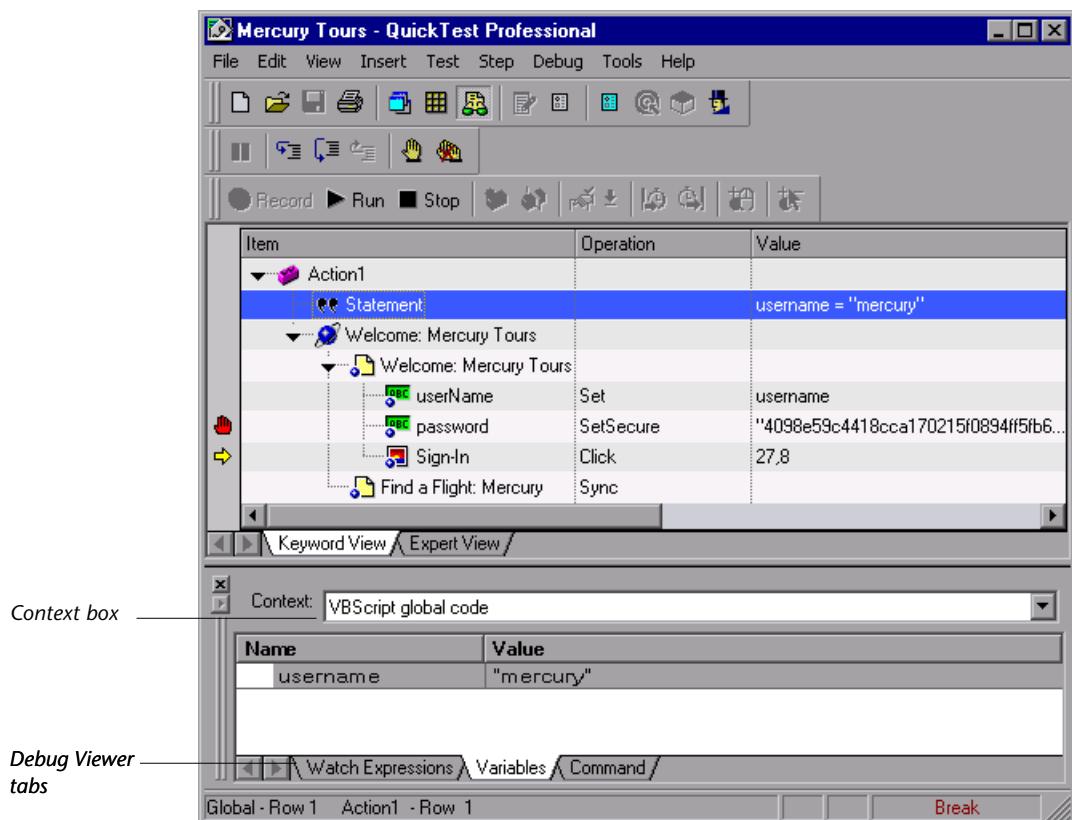
Using the Debug Viewer

You use the Debug Viewer pane to view, set, or modify the current value of objects or variables in your test or component, when it stops at a breakpoint, or when a step fails and you select the **Debug** option.

To open the Debug Viewer pane:

- 1 Run a test or component with one or more breakpoints.
- 2 When it pauses at a breakpoint, choose **View > Debug Viewer** or click the  button.

The Debug Viewer pane opens along the bottom of the QuickTest screen. If the Data Table or Active Screen are also displayed, the Debug Viewer pane is positioned below them.



The Debug Viewer tabs are used to display the values of variables and objects in the main script of the current action, or in a selected subroutine. In the **Context** box, you can choose between the main script of the action (VBScript: global code) and the subroutines and functions of the action.

Watch Expressions Tab

Use the Watch Expressions tab to view the current value of any variable or VBScript object that you enter in the Watch Expressions table. Paste or type the name of the object or variable into the **Name** column and press ENTER to view the current value in the **Value** column. If the value of the object or variable changes when you continue to run the test or component, the value in the Watch Expressions tab is updated. You can also change the value of the variable manually when the test or component pauses at a breakpoint.

Note: QuickTest updates the value of the object or variable in the Watch Expressions tab when running a test or component step-by-step.

Variables Tab

Use the Variables tab to view the current value of all variables, in the current action or selected subroutine, identified up to the point where the test or component stopped. If the value of a variable changes when the run continues, the value in the Variables tab is updated. (You can also change the value of the variable manually, during the breakpoint pause.)

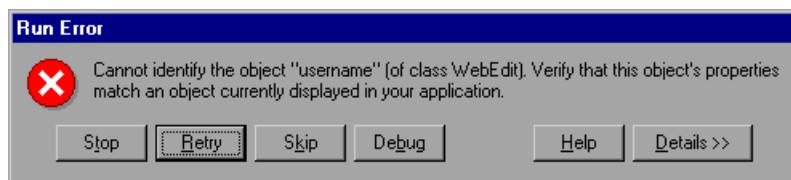
Note: QuickTest updates the value of the variable in the Variables tab when running a test or component step by step.

Command Tab

Use the Command tab to execute a line of script in order to set or modify the current value of a variable or VBScript object in your test or component. When the run continues, QuickTest uses the value that you set.

Handling Run Errors

The Run Error message box displayed during a run session offers a number of option buttons for dealing with errors encountered:



- **Stop**—Stops the run session.
The test results are displayed if QuickTest is configured to show test 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 or component.
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 component stopped, or you can use the step commands to control the remainder of the run session.
- **Help**—Opens QuickTest troubleshooting Help for the displayed error message. After you review the Help topic, you can select another button in the error message box.
- **Details**—Expands the message box to display additional information about the error.

Practicing Debugging a Test

Suppose you create an action in your test that defines variables that will be used in other parts of your test. You can add breakpoints to the action to see how the value of the variables change as you run the test. To see how the test handles the new value, you can also change the value of one of the variables during a breakpoint.

Step 1: Create the New Action

Open a test and insert a new action called “SetVariables.” For more information about inserting actions, see Chapter 17, “Working with Actions.”

Enter the VBScript code for the action in the Expert View, as follows:

```
Dim a  
a="hello"  
b="me"  
MsgBox a
```

For more information about the Expert View, see Chapter 36, “Working with the Expert View.”

Step 2: Add Breakpoints

Add breakpoints at line 3 and line 4. For more information about adding breakpoints, see “Setting Breakpoints” on page 533.

Step 3: Begin Running the Test

Run the test. The test stops at the first breakpoint, before executing that step (line of script).

Step 4: Check the Value of the Variables in the Debug Viewer Pane

Choose **View > Debug Viewer** to open the Debug Viewer pane.

Select the **Watch Expressions** tab on the Debug Viewer pane. In the first cell of the Name column, type "a" (without quotes) and press **Enter** on the keypad. The Value column indicates that the **a** variable value is currently **hello**, because the breakpoint stopped after the value of variable **a** was initiated. In the next cell of the Name column, type "b" (without quotes) and press **Enter** on the keypad. The Value column indicates that **Variable b is undefined**, because the test stopped before variable **b** was declared.

Select the **Variables** tab in the Debug Viewer pane. Note that the variable **a** is displayed with the value **hello**, because **a** is the only variable initiated, at this point in the test.

Step 5: 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** have both been updated in the Watch Expressions and Variables tabs.

Step 6: Modify the Value of a Variable Using the Command Tab



Select the **Command** tab in the Debug Viewer pane. Type: **a="This is the new value of a"** at the command prompt, and press **Enter** on the keypad. Click the **Run** button to continue running the test. The message box that appears displays the new value of **a**.

23

Analyzing Test Results

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

This chapter describes:

- About Analyzing Test Results
- Understanding the Test Results Window
- Viewing the Results of a Run Session
- Viewing Checkpoint Results
- Viewing Parameterized Values and Output Value Results
- Analyzing Smart Identification Information in the Test Results
- Deleting Test Results
- Submitting Defects Detected During a Run Session
- Viewing WinRunner Test Steps in the Test Results
- Customizing the Test Results Display

About Analyzing Test Results

When a run session ends, you can view the test 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, select or clear the **View results when run session ends** check box in the Run tab of the Options dialog box.

The Test Results window contains a description of the steps performed during the run session. For a component, or 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 tab of the Test Settings dialog box. For more information, see “Defining Run Settings for Your Test” on page 665.

After you run a test or component, the Test Results window displays all aspects of the run session, including:

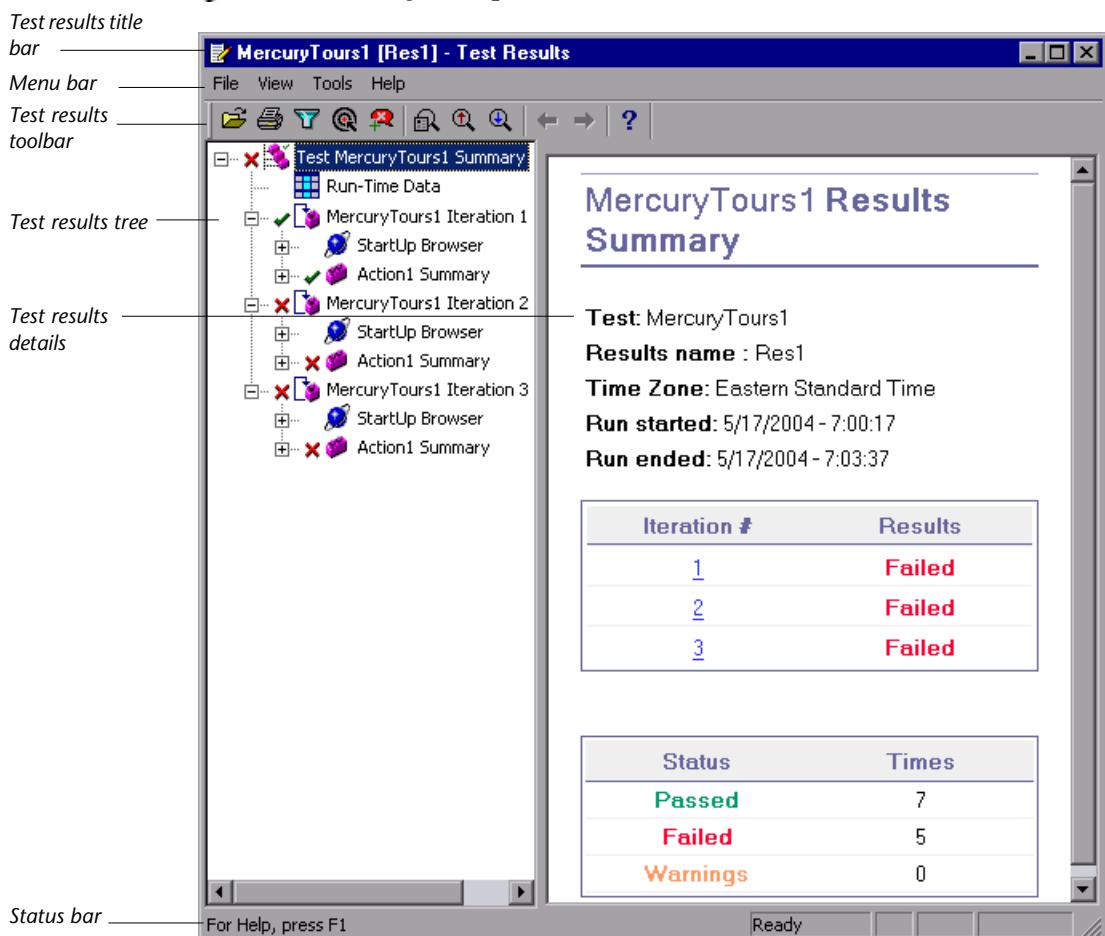
- a high-level results overview report (pass/fail status)
- the data used in all runs
- an expandable tree of the steps, specifying exactly where application failures occurred
- the exact locations in the test or component where failures occurred
- application snapshots that highlight any discrepancies, for each stage of the test or component
- detailed explanations of each step and checkpoint pass or failure, at each stage of the test or component

Understanding 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 “Defining Run Settings for Your Test” on page 665.

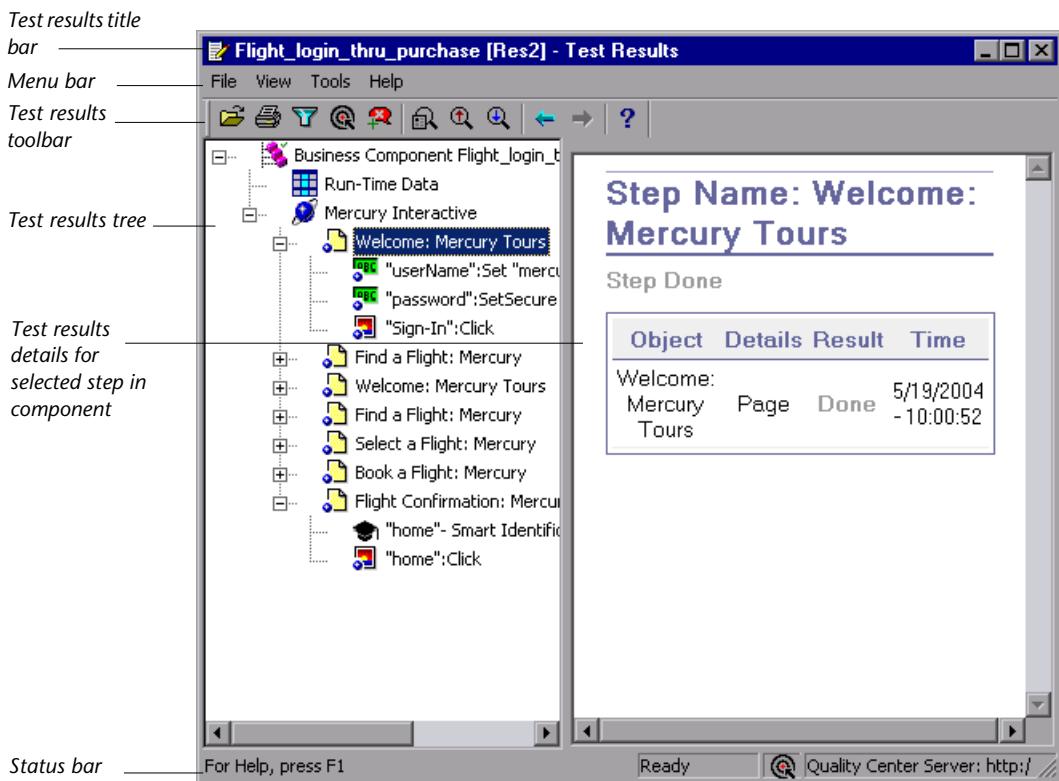
Note: You can open the Test Results window as a standalone application from the Start menu. To open the Test Results window, choose **Start > Programs > QuickTest Professional > Test Results Viewer.**

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 test results tree. Note that the results for a test are organized by action.

Below is an example of the test results for a business component, with a specific step selected:



The Test Results window contains the following key elements:

- **Test results title bar**—Displays the name of the test or component.
- **Menu bar**—Displays menus of available commands.
- **Test results toolbar**—Contains buttons for viewing test results (choose **View > Test Results Toolbar** to display the toolbar). For more information, see “Test Results Toolbar” on page 546.
- **Test results tree**—Contains a graphic representation of the test results in the test results tree. For more information, see “Test Results Tree”, below.

- **Test results details**—Contains details of the selected step. For more information, see “Test Results Details” on page 546.
- **Status bar**—Displays the status of the currently selected command (choose **View > Status Bar** to view the status bar).

Test Results Tree

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

- indicates a step that succeeded.
- 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.

The test results tree also includes the icon that displays the *run-time Data Table*—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.

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

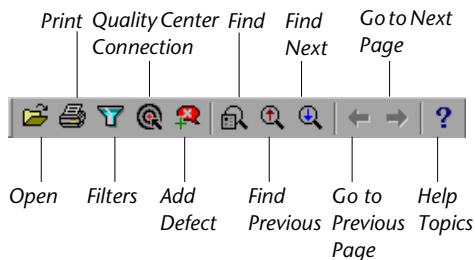
Test Results Details

By default, when the Test Results window opens, a test or component summary is displayed in the right portion of the window. Indicated are the test or component name, results name, the date and time of the run, the number of iterations (for a test), and whether an iteration passed or failed.

When you select a branch or step in the tree, the right pane displays detailed information for the selected item.

Test Results Toolbar

The Test Results toolbar contains buttons for viewing test results.



Viewing the Results of a Run Session

By default, at the end of the run session, the results are displayed in the Test Results window. (You can change the default setting in the Options dialog box. For more information, see “Defining Run Settings for Your Test” on page 665.)

In addition, you can view the results of previous runs of the current test or component, and results of other tests and components. You can also preview test results on screen and print them to your default Windows printer.

To view the results of a run:

- 1** If the Test Results window is not already open, click the **Results** button or choose **Test > Results** or **Component > Results**.

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

- If there are test results for the current test or component, they are displayed in the Test Results window. For information on the Test Results window, see “Understanding the Test Results Window” on page 542.
- If there are several test results for the current test or component, or if there are no test results for the current test or component, the Open Test Results dialog box opens. You can select the test results for any test or component, or you can search for the test results (**results.xml**) file anywhere in the file system. Click **Open** to display the selected results in the Test Results window. For additional information on viewing test results, see “Opening Test Results to View a Selected Run” on page 551.

Note: Results files for QuickTest Professional version 6.5 and earlier are saved with a **.qtp** file extension.

- 2** You can collapse or expand a branch in the test results tree to select the level of detail that the tree displays.
 - 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 test results tree, choose **View > Collapse All** or right click a branch and select **Collapse All**.

- 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.

If you just opened the test results, the tree expands one level at-a-time. If the tree was previously expanded, it reverts to its former state.

- 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 test results tree, choose **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.

- 3** You can view the results of an individual iteration, action, or step. The results can be one of three types:

- Iterations, actions, and steps that contain checkpoints are marked **Passed** or **Failed** in the bottom right part of the Test Results window and are identified by the icon ✓ or ✗ in the tree pane.
- Iterations, actions, and steps that ran successfully, but do not contain checkpoints, are marked **Done** in the bottom right part of the Test Results window.
- Steps that were not successful, but did not cause the test or component to stop running, are marked **Warning** in the bottom right part of the Test Results window and are identified by the icon ! or ! ✗.

Note: A test, component, iteration, or action containing a step marked **Warning** may still be labeled **Passed** or **Done**.



- 4 To filter the information displayed in the test results window, click the **Filters** button or choose **View > Filters**. The Filters dialog box opens.



The default filter options are displayed in the image above. The Filters dialog box contains the following options:

Iterations area (available for tests only):

- **All**—Displays test results from all iterations.
- **From iteration X to X**—Displays the test results from a specified range of test iterations.

Status area:

- **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 or component 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 area:

- **All**—Displays all steps from all nodes in the test or component.
- **Show only actions**—Displays the action nodes in the test (not the specific steps in the action nodes).



- 5** To find specific steps within the test results, click the **Find** button or choose **Tools > Find**.



- 6** To move between previously selected nodes within the test results tree, click the **Go to Previous Node** or **Go to Next Node** buttons.



- 7** To view the results of other run sessions, click the **Open** button or choose **File > Open**. For additional information, see “Opening Test Results to View a Selected Run” on page 551.



- 8** To print test results, click the **Print** button or choose **File > Print**. For additional information, see “Printing Test Results” on page 555.

Note: If you have Quality Center installed, you can add a defect to a Quality Center project. For additional information, see “Submitting Defects Detected During a Run Session” on page 597.

- 9** Choose **File > Exit** to close the Test Results window.
-

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 about saving run session information, see “Choosing Which Steps to Report During the Run Session” on page 904, or refer to the *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.

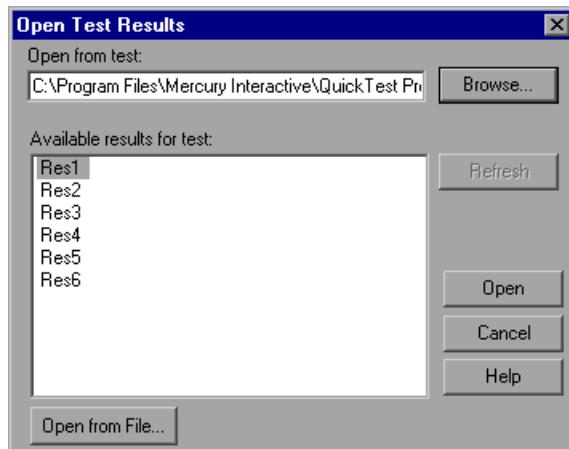
Opening Test Results to View a Selected Run

You can view the saved results for the current test or component, or you can view the saved results for other tests or components.

You select the test results to open for viewing from the Open Test Results dialog box, which opens when:



- you choose **File > Open** from within the Test Results window.
- you click the **Results** button in the QuickTest window or choose **Test > Results** or **Component > Results**, when there are several results, or no results, for the current test or component.



The results of run sessions for the current test or component are listed. To view one of the results sets, select it and click **Open**.

Tip: To update the results list after you change the specified test or component path, click **Refresh**.

To view results of runs for other tests or components, you can search in your file system by test or component, or by their result files. If you are connected to Quality Center, you can also search in Quality Center by business process test or by QuickTest test (if saved in Quality Center).

Searching for Results in the File System

By default, the results for a QuickTest test that is saved in the file system are stored in the test folder. The results for components are stored in a Quality Center cache folder on your computer.

When you run your test or component, 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 “Understanding the Results Location Tab” on page 514.

To search for results in the file system by test or component:

- 1 In the Open Test Results dialog box, enter the path of the folder that contains the results file for your test or component, or click the Browse button to open the Open Test dialog box.
- 2 Find and highlight the test or component whose results you want to view, and click **Open**.
- 3 In the Open Test Results dialog box, highlight the test result set you want to view, and click **Open**. The Test Results window displays the selected results.

To search for results in the file system by result file:

- 1 In the Open Test Results dialog box, click the **Open from File** button to open the Select Results File dialog box.
- 2 Browse to the folder where the test or component results file is stored.
- 3 Highlight the results (.xml) file you want to view, and click **Open**. The Test Results window displays the selected results.

Notes: By default, result files for tests are stored in **<Test>\<ResultName>\Report**.

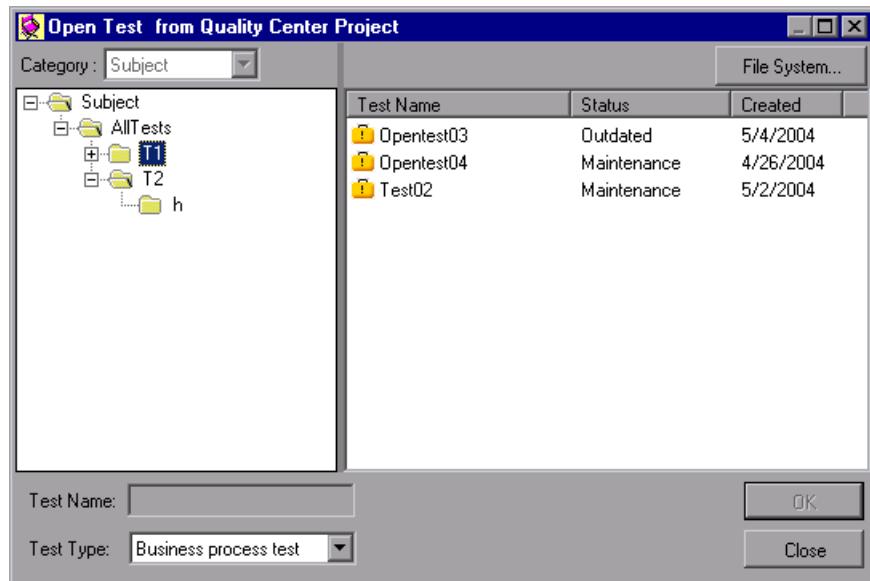
Results files for QuickTest Professional version 6.5 and earlier are saved with a **.qtp** file extension. In the Select Results File dialog box, only results files with an **.xml** extension are shown by default. To view results files with a **.qtp** extension in the Select Results File dialog box, select **Test Results (*.qtp)** in the **Files of type** box.

Searching for Results in Quality Center

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 test results. You can search in Quality Center for a QuickTest test or a business process test. You can view the results of all the components in the business process test. You cannot search by individual component in Quality Center.

To search for results in Quality Center by test type:

- 1 In the Open Test Results dialog box, enter the path of the folder that contains the results file for your QuickTest test or business process test, or click the Browse button to open the Open Test from Quality Center Project dialog box.



- 2 Select **QuickTest test** or **Business process test** in the **Test Type** list.
- 3 Find and highlight the test whose test results you want to view, and click **Open**.
- 4 In the Open Test Results dialog box, highlight the test result set you want to view, and click **Open**. The Test Results window displays the selected test results. For a business process test, all the components are shown in the test results tree. You can select an individual component in the test results tree to see its results.

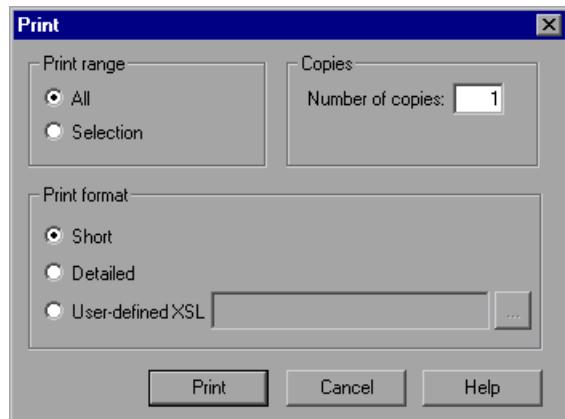
Printing Test Results

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 choose **File > Print**. The Print dialog box opens.



- 2 Select a **Print range** option:

- **All**—Prints the results for the entire test or component.
- **Selection**—Prints test results information for the selected branch in the test 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 test results tree. This option is only available if you selected **All** in step 2.
- **Detailed**—Prints all available information for each item in the test results tree.
- **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 606.

Note: The **Print format** options are available only for test results created with QuickTest version 8.0 and later.

- 5 Click **OK** to print the selected test results information to your default Windows printer.

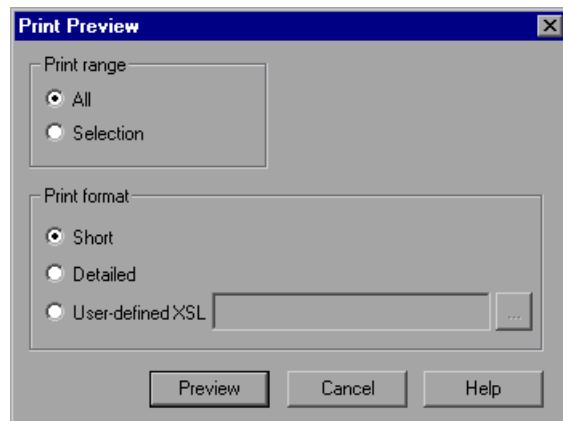
Previewing Test Results

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 Choose **File > Print Preview**. The Print Preview dialog box opens.



2 Select a **Print range** option:

- **All**—Previews the test results for the entire test or component.
- **Selection**—Previews test results information for the selected branch in the test results tree.

3 Select a **Print format** option:

- **Short**—Previews a summary line (when available) for each item in the test results tree. This option is only available if you selected **All** in step 2.
- **Detailed**—Previews all available information for each item in the test results tree.
- **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 606.

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**.

Viewing Checkpoint Results

By adding checkpoints to your tests or components, 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 or component, 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 or component 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 or component in the Test Results window. For more information, see “Viewing the Results of a Run Session” on page 546.
- 2** In the left pane of the Test Results window, expand the branches of the test 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 right part of the Test Results window displays information about 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 tab of the Options dialog box. For more information, see “Setting Run Testing Options” on page 625.

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 559
- “Analyzing Table and Database Checkpoint Results” on page 560
- “Analyzing Bitmap Checkpoint Results” on page 562
- “Analyzing Text or Text Area Checkpoint Results” on page 563
- “Analyzing XML Checkpoint Results” on page 564
- “Analyzing Accessibility Checkpoint Results” on page 573

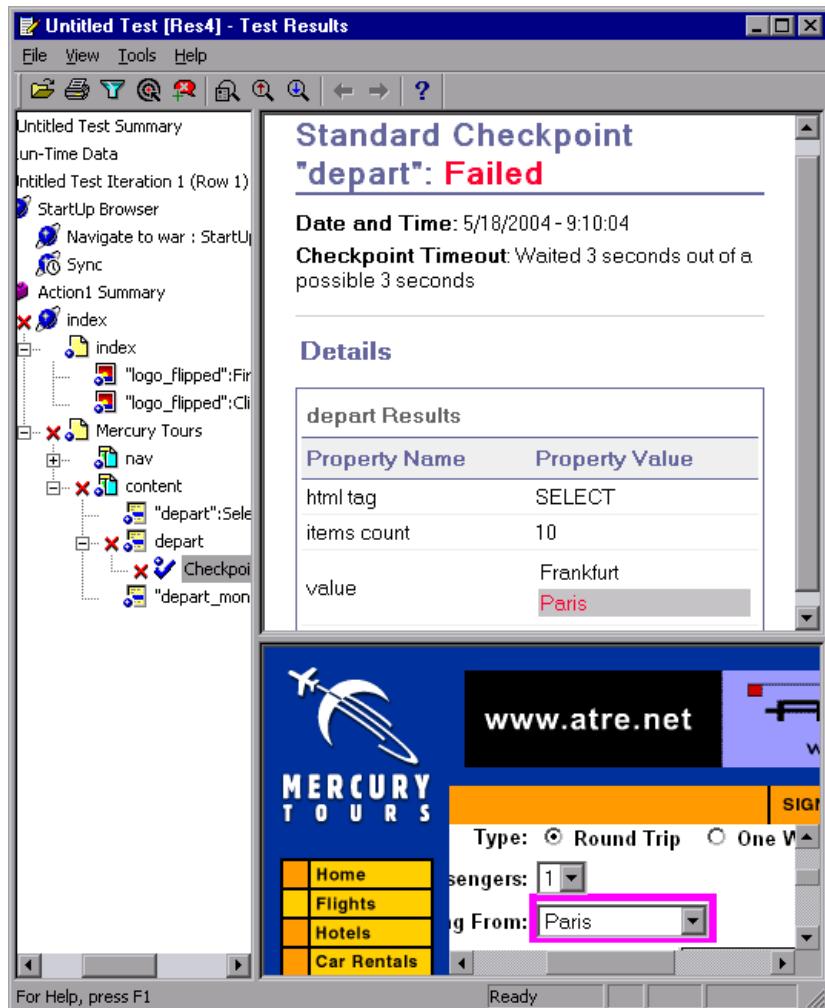
- 3** Choose **File > Exit** to close the Test Results window.

For more information on checkpoints, see Chapter 6, “Understanding Checkpoints.”

Analyzing Standard Checkpoint Results

By adding standard checkpoints to your tests or components, 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" on page 123.

You can view detailed results of the standard checkpoint in the Test Results window.



The top right pane 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 right 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 **Frankfurt**, but the actual value is **Paris**.

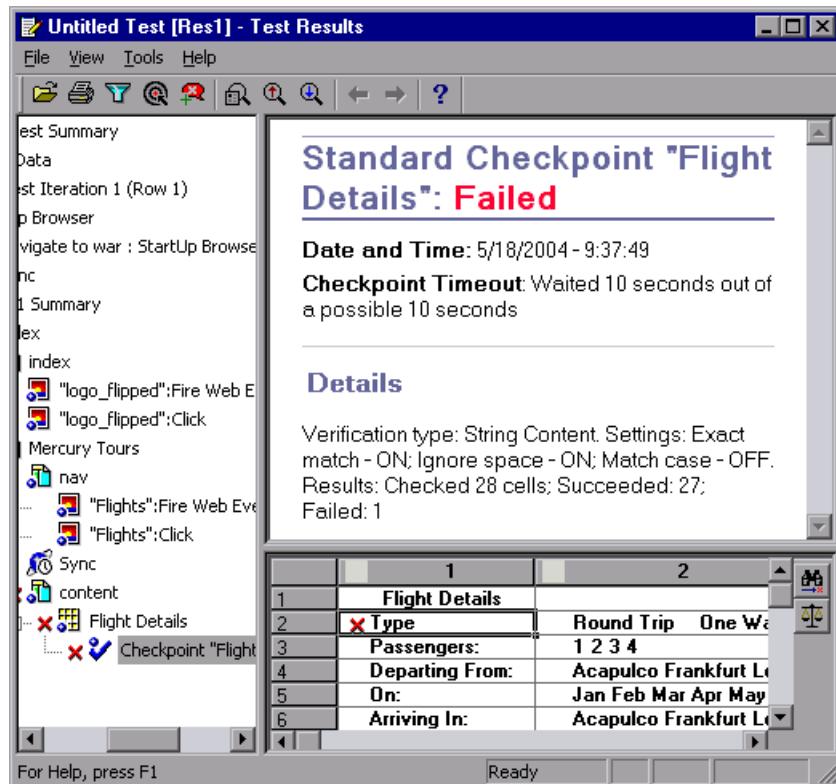
Analyzing Table and Database Checkpoint Results

By adding table checkpoints to your tests or components, 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 or components, 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 or component, 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 8, “Checking Tables and Databases.”

You can view detailed results of the table or database checkpoint in the Test Results window.



The top right pane 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.

The bottom right pane 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 **X** icon.



You can click the **Next Mismatch** button in the bottom right pane to highlight the next table cell or database record that failed the checkpoint.



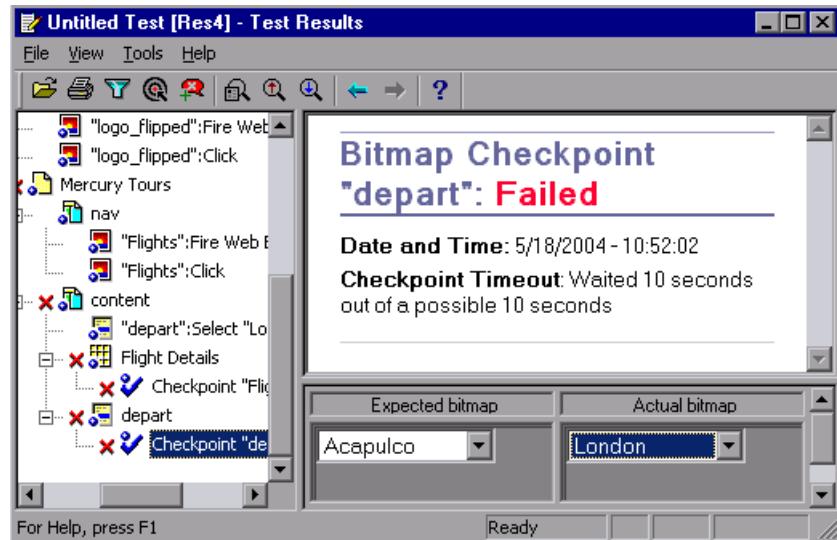
You can click the **Compare Values** button in the bottom right pane to display the expected and actual values of the selected table cell or database record.

Analyzing Bitmap Checkpoint Results

By adding bitmap checkpoints to your tests or components, you can check the appearance of elements in your application by matching captured bitmaps. When you run your test or component, 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 bitmap checkpoints, see Chapter 10, “Checking Bitmaps.”

You can view detailed results of the bitmap checkpoint in the Test Results window.



The top right pane 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).

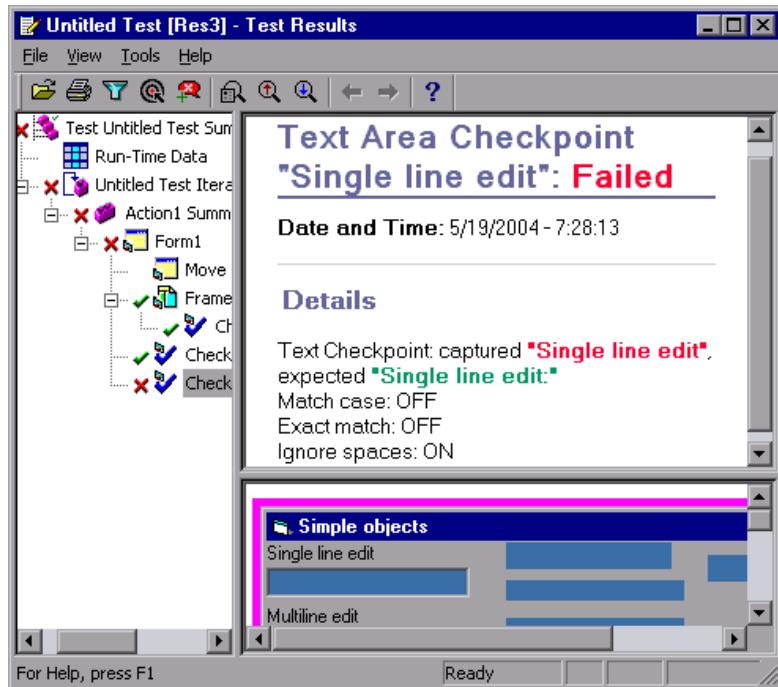
The bottom right pane shows the expected and actual bitmaps that were compared during the run session.

Analyzing Text or Text Area Checkpoint Results

By adding text or text area checkpoints to your tests or components, you can check that a text string is displayed in the appropriate place in your application. When you run your test or component, 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 9, "Checking Text."

You can view detailed results of the text or text area checkpoint in the Test Results window.

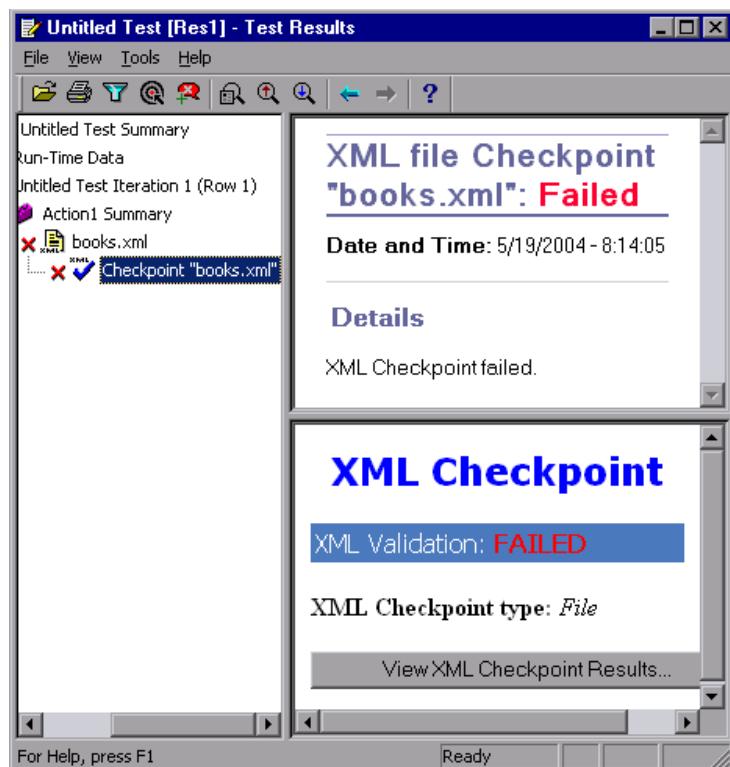


The top right pane displays the checkpoint step results, including its status (**Passed** or **Failed**), the date and time the checkpoint was run and the portion of the checkpoint timeout interval that was used (if any). It also shows the expected text and actual text that was checked, and the verification settings you specified for the checkpoint.

Analyzing XML Checkpoint Results

By adding XML checkpoints to your tests or components, you can verify that the data and structure in your XML documents or files has not changed unexpectedly. When you run your test or component, 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 11, “Checking XML.”

You can view summary results of the XML checkpoint in the Test Results window.



The top right pane displays the checkpoint step results.

The bottom right pane shows the details of the schema validation (if applicable) and a summary of the checkpoint results. If the schema validation failed, the reason(s) for the failure is 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 right pane. 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. If you want to view the detailed results of the checkpoint even when the checkpoint passes, choose **Tools > Options** and select the **Run** tab. In the **Save step screen capture to test results** option, select **always**.

Understanding the XML Checkpoint Results Window

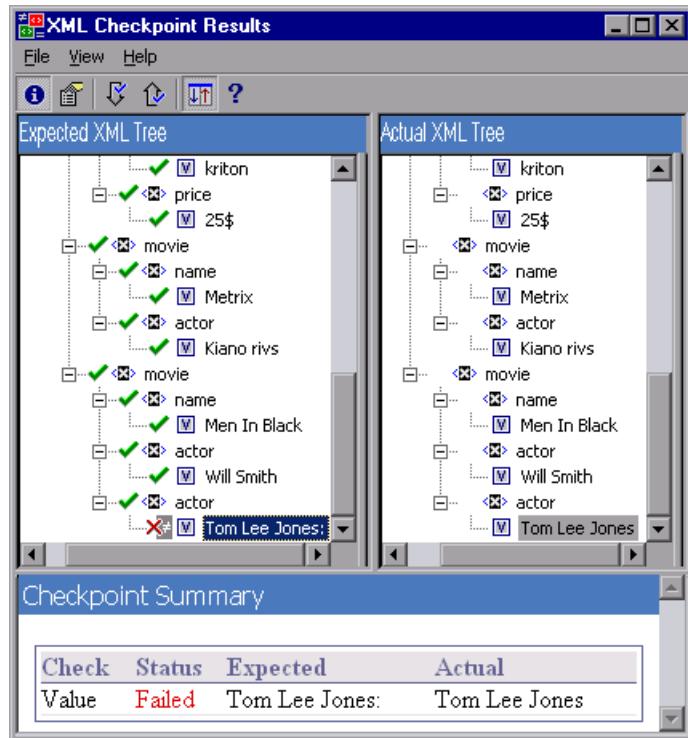
When you click the **View XML Checkpoint Results** button from the Test Results window, the XML Checkpoint Results window displays the XML file hierarchy.

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 following commands or toolbar buttons to navigate your XML checkpoint results:

- **View Checkpoint Summary**—Select an element in the XML Tree and click the **View Checkpoint Summary** button or choose **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 book element in an XML file.

The screenshot shows the 'XML Checkpoint Results' application window. It has two main panes: 'Expected XML Tree' on the left and 'Actual XML Tree' on the right. Both panes display an XML structure for an RSS feed with a 'Books' channel containing three 'book' elements. Each 'book' element has attributes like 'name', 'author', and 'price'. Below the trees is a 'Checkpoint Summary' table:

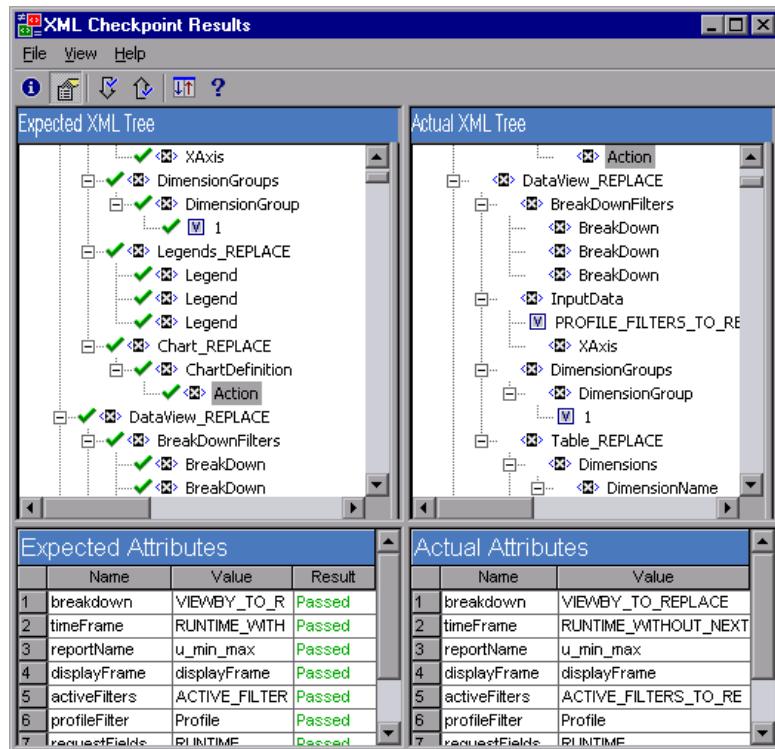
Check	Status	Expected	Actual
Attributes check	Passed	See attributes table for more information.	
Number of children of type <Any Child>	Passed	3	3



- **View Attribute Details**—In the XML Tree, select an element whose attributes were checked. Click the **View Attribute Details** button or choose **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**—Choose **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**—Choose **View > Find Previous Check** or click the **Find Previous Check** button to jump directly to the previous checked item in the XML Tree.



- **Scroll Trees Simultaneously**—Choose **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.
- **Help Topics**—Choose **Help > Help Topics** or click the **Help Topics** button to view help on the XML Checkpoint Results window.



Examining Sample XML Checkpoint Results

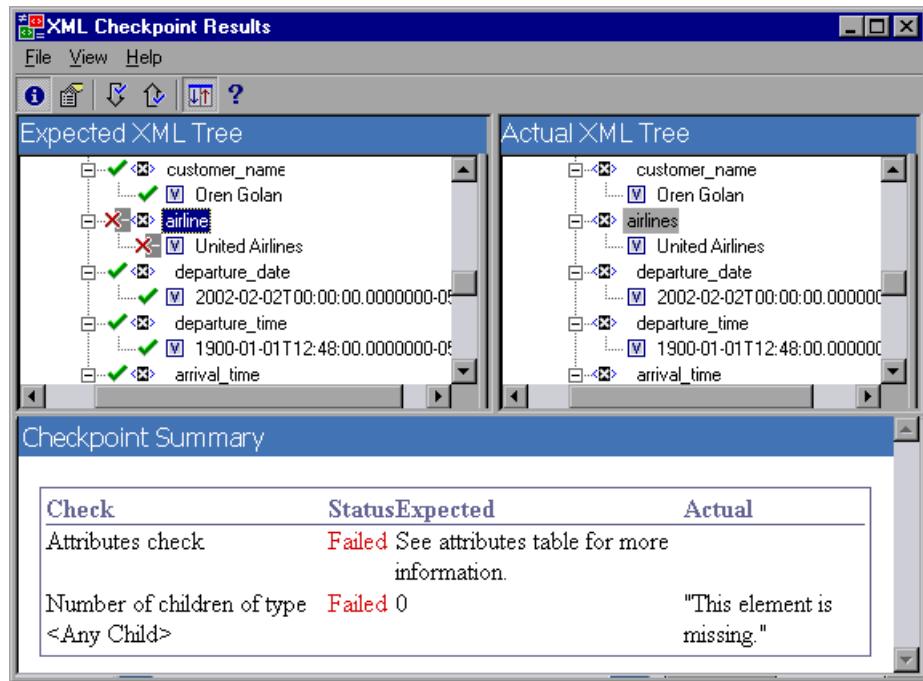
Below are four sample XML checkpoint scenarios. Each example describes the changes that occurred in the actual XML document, explains how you locate the cause of the problem in the XML checkpoint results, and displays the corresponding XML Checkpoint Results window.

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 choose **View > Checkpoint Summary** to view the Checkpoint Summary in the bottom pane of the XML Checkpoint Results window.

The text: This element is missing indicates that the airline element tag changed in your XML document.

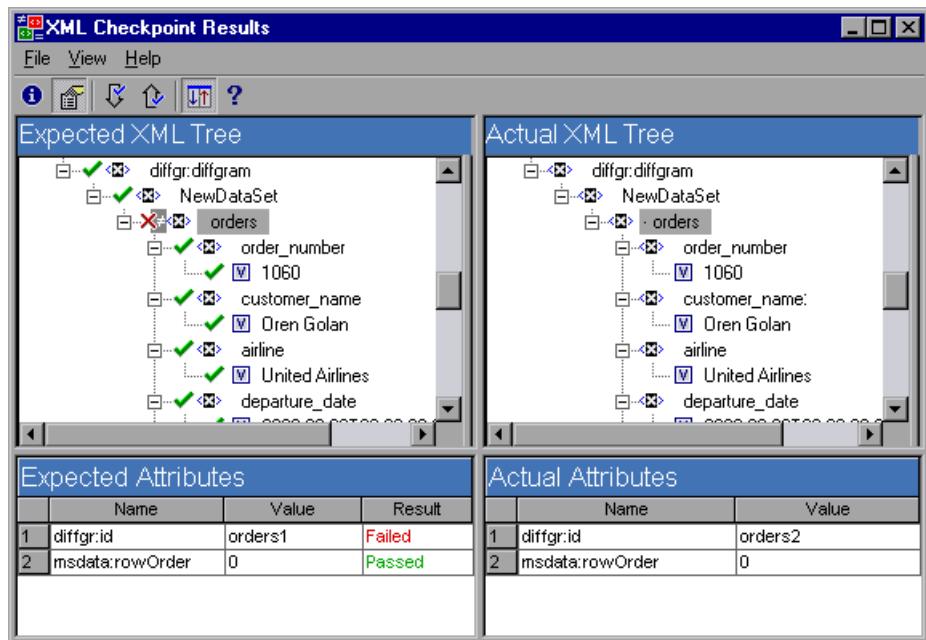


Scenario 2

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 choose **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.

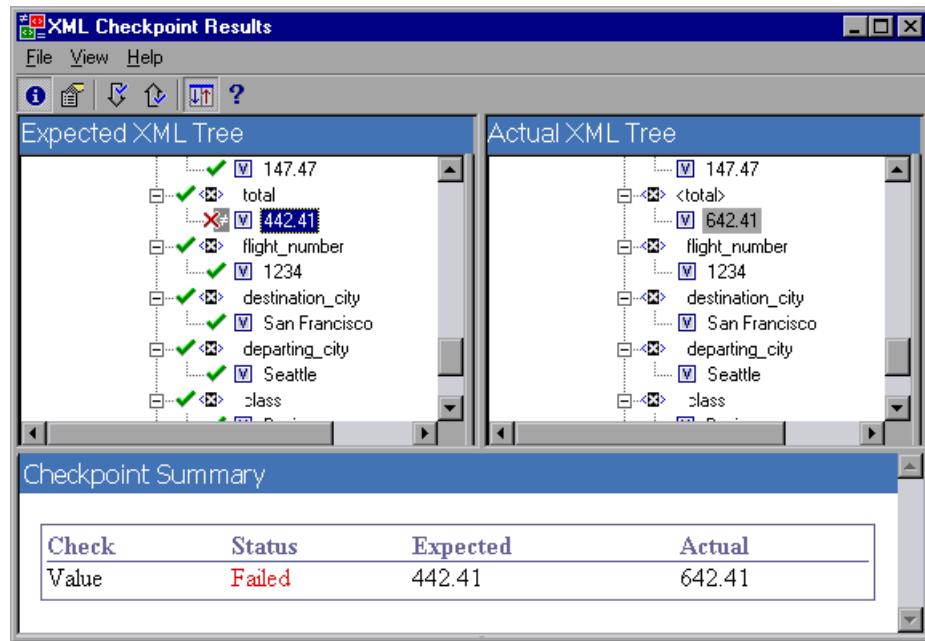


Scenario 3

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 choose **View > Checkpoint Summary** to view the Checkpoint Summary in the bottom pane of the XML Checkpoint Results window.

Using the Checkpoint Summary pane, you can compare the expected and actual values of the total element.



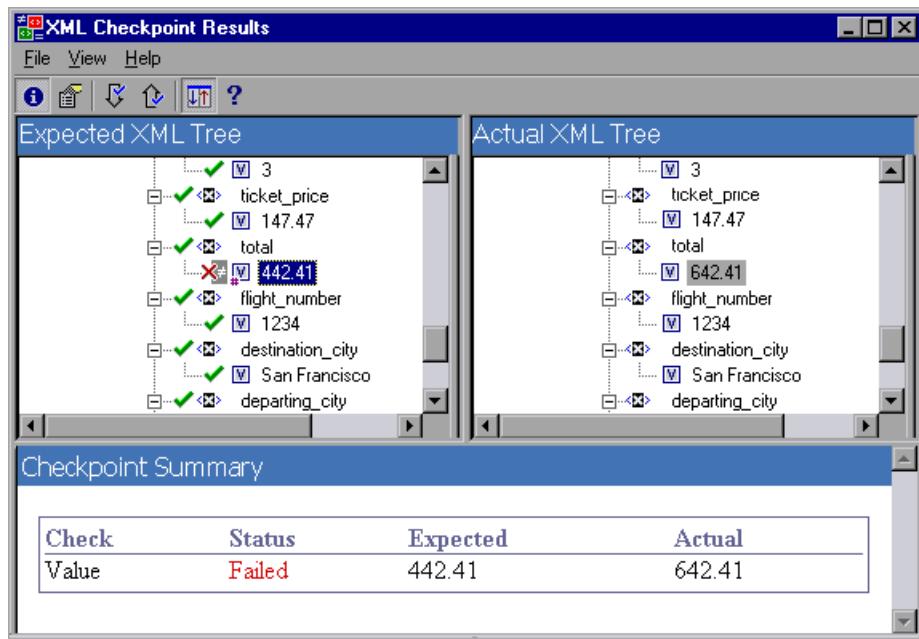
Scenario 4

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 choose **View > Checkpoint Summary** to view the Checkpoint Summary in the bottom pane of the XML Checkpoint Results window. Note that the procedure for analyzing the checkpoint results does not change even though the value was parameterized.

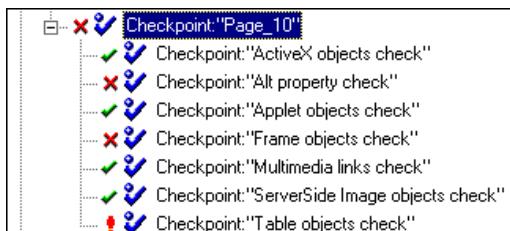
Using the Checkpoint Summary pane, you can compare the expected and actual values of the total element.



Analyzing Accessibility Checkpoint Results

When you include accessibility checkpoints in your test or component, the Test Results window displays the results of each accessibility option that you checked.

The test 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 test results tree for an accessibility checkpoint may look something like this:



The test results 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 Chapter 30, “Testing Web Objects.”

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	ActiveMovie1

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 or component fails and the test results 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 right pane 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), 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 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 or component fails and the test results details display a list that shows which objects are lacking the tag. For example:

Frame objects check			
Object Class	Object Tag	Object Name	Title Value
Frame	FRAME	f	Form With button input
Frame	FRAME	Subframe	[NONE]
Frame	FRAME	frame1	FORMS
Frame	FRAME	frame2	Form With button input
Frame	FRAME	frame3	Targets
Frame	FRAME	frame4	Base element
Page		Frame into Frame	Frame into Frame

The bottom right part 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 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 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:

ServerSide Image objects check	
Object Class	Object Name
Image	[Historical Congressional Documents]

Tables Check

Guideline 5 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:

Table objects check																								
Object Class	Object Name	Table Structure																						
WebTable	Table 1	<table border="1"><tr><td>TD</td><td>TD</td></tr><tr><td>TD</td><td>TD</td></tr><tr><td>TD</td><td>TD</td></tr><tr><td>TD</td><td>TD</td></tr><tr><td>TD</td><td>TD</td></tr><tr><td>TD</td><td>TD</td></tr><tr><td>TD</td><td>TD</td></tr><tr><td>TD</td><td>TD</td></tr><tr><td>TD</td><td>TD</td></tr><tr><td>TD</td><td></td></tr><tr><td>TD</td><td>TD</td></tr></table>	TD		TD	TD																		
TD	TD																							
TD	TD																							
TD	TD																							
TD	TD																							
TD	TD																							
TD	TD																							
TD	TD																							
TD	TD																							
TD	TD																							
TD																								
TD	TD																							

Viewing Parameterized Values and Output Value Results

You can view information about 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

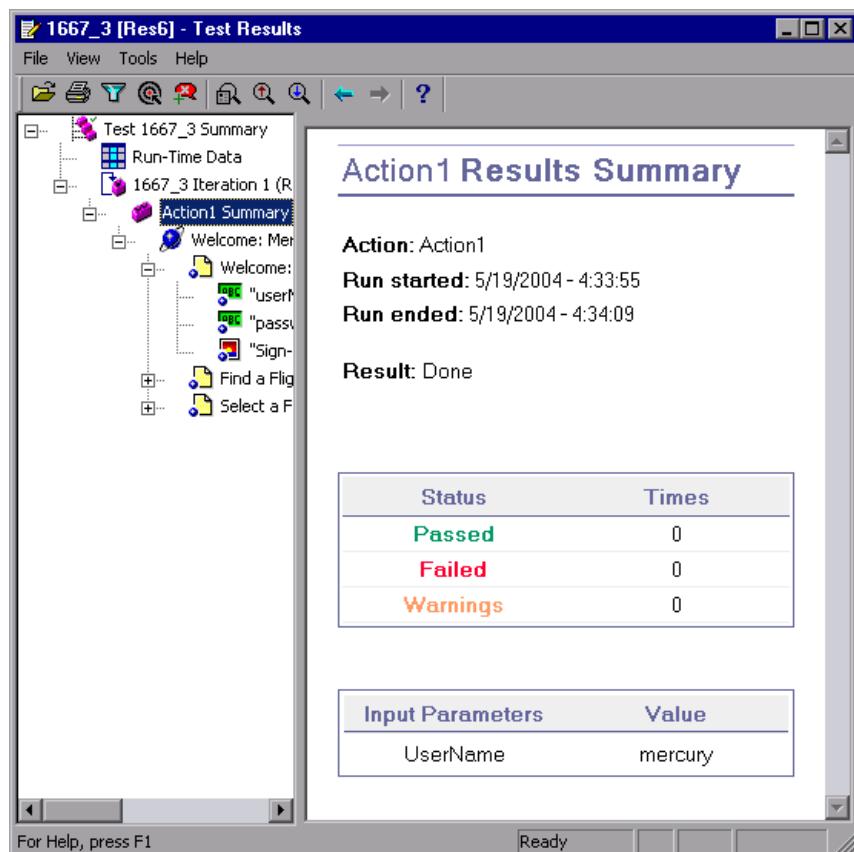
A *parameter* is a variable that is assigned a value from an external data source or generator. For more information on defining and using parameters in your tests and components, see Chapter 12, “Parameterizing Values.”

You can view the values for the parameters defined in your test or component in the Test Results window.

To view parameterized values:

- 1** Display the test results for your test or component in the Test Results window. For more information, see “Viewing the Results of a Run Session” on page 546.
- 2** In the left pane of the Test Results window, expand the branches of the test results tree and click the branch for the test, action, component, or step that contains parameterized values.

The name and value of the input parameters are displayed in the bottom 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 and components, see Chapter 12, “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. For more information on defining and storing output values, see Chapter 13, “Outputting Values.”

To view the results of an output value step :

- 1 Display the test results for your test or component in the Test Results window. For more information, see “Viewing the Results of a Run Session” on page 546.
- 2 In the left pane of the Test Results window, expand the branches of the test 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 screenshot shows the 'Test Results' window for a test named '1667_3'. The left pane displays a hierarchical tree of test steps. The 'fromPort' step under 'Find a Flight' is expanded, showing its properties and results. The right pane displays the 'Standard Output Value' for 'fromPort' with the status 'Done'. It also shows the date and time of the capture and a detailed table of the captured properties.

Name	Captured Value	Type	Name
value	Acapulco	Environment	Departure

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 13, “Outputting Values.”

For information on viewing the results of XML output value steps, see “Analyzing XML Output Value Results” on page 582.

Viewing the Run-Time Data Table

After running a test or component with Data Table parameters and/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. For more information on the run-time Data Table, see Chapter 18, “Working with Data Tables.”

To view the run-time Data Table:

- 1** Display the test results for your test or component in the Test Results window. For more information, see “Viewing the Results of a Run Session” on page 546.



- 2 Highlight **Run-Time Data** in the left pane of the Test Results window.

The screenshot shows the 'MercuryTours1 [Res1] - Test Results' window. On the left, the tree view shows 'Test MercuryTours1 Summary' expanded, with 'Run-Time Data' selected. Under 'Run-Time Data', three iterations are listed: 'MercuryTours1 Iteration 1' (green checkmark), 'MercuryTours1 Iteration 2' (red X), and 'MercuryTours1 Iteration 3' (red X). To the right is a data grid titled 'A3 | Seattle'. The grid has columns labeled 'departure', 'B', 'C', and 'D'. Row 1 contains 'New York'. Row 2 contains 'Portland'. Row 3 contains 'Seattle'. Rows 4 through 17 are empty. At the bottom of the window, there are tabs for 'Global' and 'Action1', and status indicators 'Ready' and 'For Help, press F1'.

	departure	B	C	D
1	New York			
2	Portland			
3	Seattle			
4				
5				
6				
7				
8				
9				
10				
11				
12				
13				
14				
15				
16				
17				

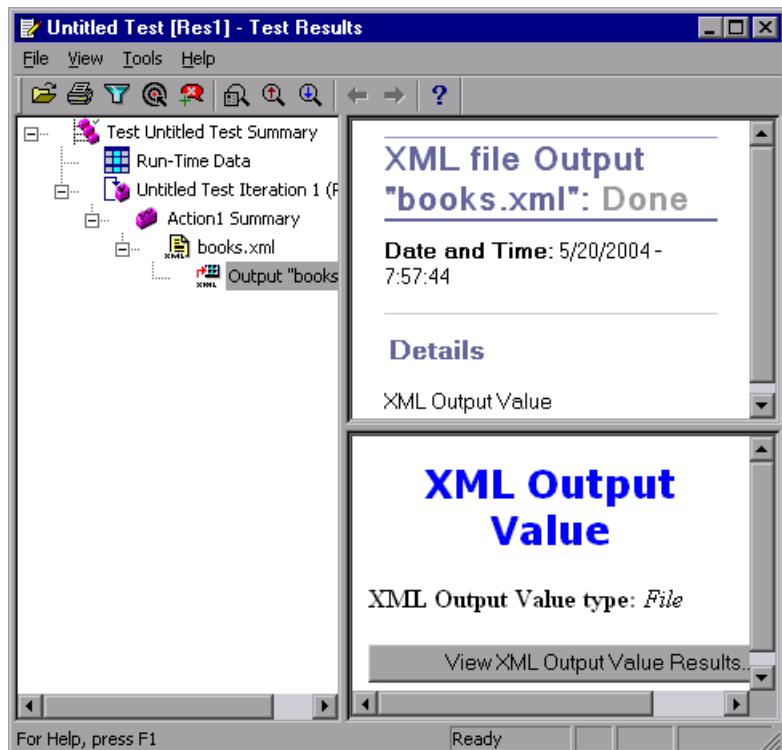
In the above example, the Run-Time Data Table contains the parameterized flight departure values.

For more information on the run-time Data Table, see Chapter 18, “Working with Data Tables.”

Analyzing XML Output Value Results

You can output element or attribute values to your test or component from XML documents used in your application. For more information on XML output values, see “Outputting XML Values” on page 280.

You can view summary results of the XML output value in the Test Results window.



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. If you want to have the option to view the detailed results of the output value after every run, choose **Tools > Options** and select the **Run** tab. In the **Save step screen capture to test results** option, select **always**.

For more information on XML output value results, see “Understanding the XML Output Value Results Window,” below.

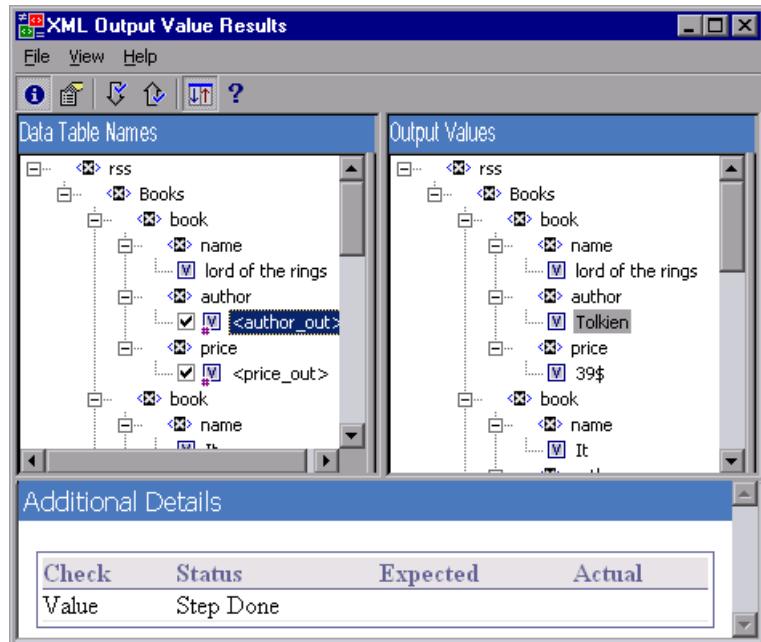
Understanding the XML Output Value Results Window

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.



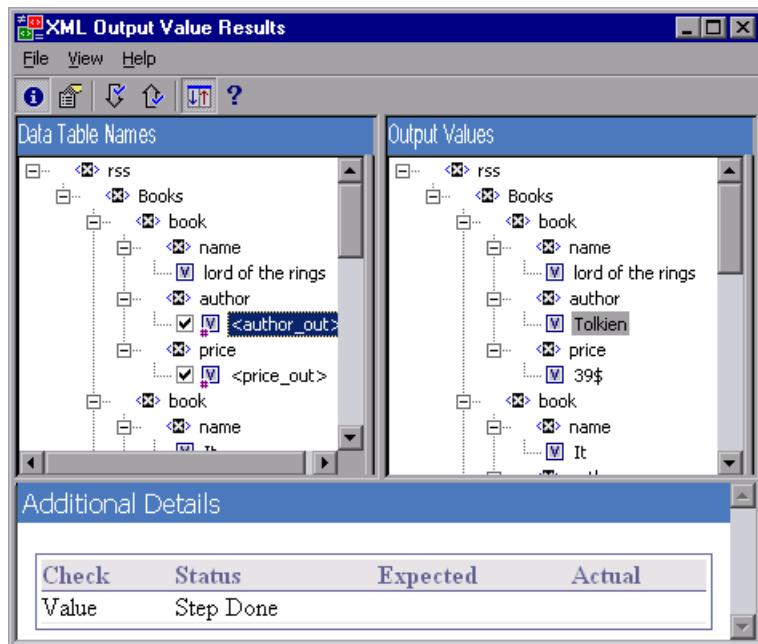
Navigating the XML Output Value Results Window

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 following commands or toolbar buttons to navigate your XML output value results:

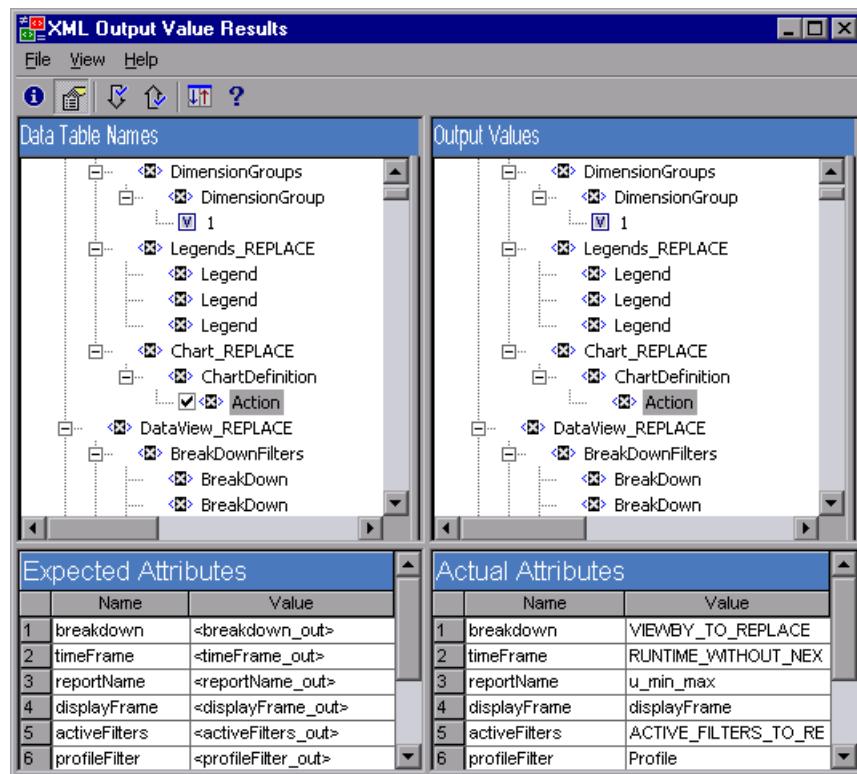


- **View Output Value Summary**—Select an element in the XML Tree and click the **View Output Value Summary** button or choose **View > Output Value Summary**. The Additional Details pane, which provides information regarding the output value for the selected element, attribute, or value, is displayed 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 choose **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**—Choose **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**—Choose **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.
- **Scroll Trees Simultaneously**—Choose **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**—Choose **Help > Help Topics** or click the **Help Topics** button to view help on the XML Output Value Results window.

Analyzing Smart Identification Information in the Test Results

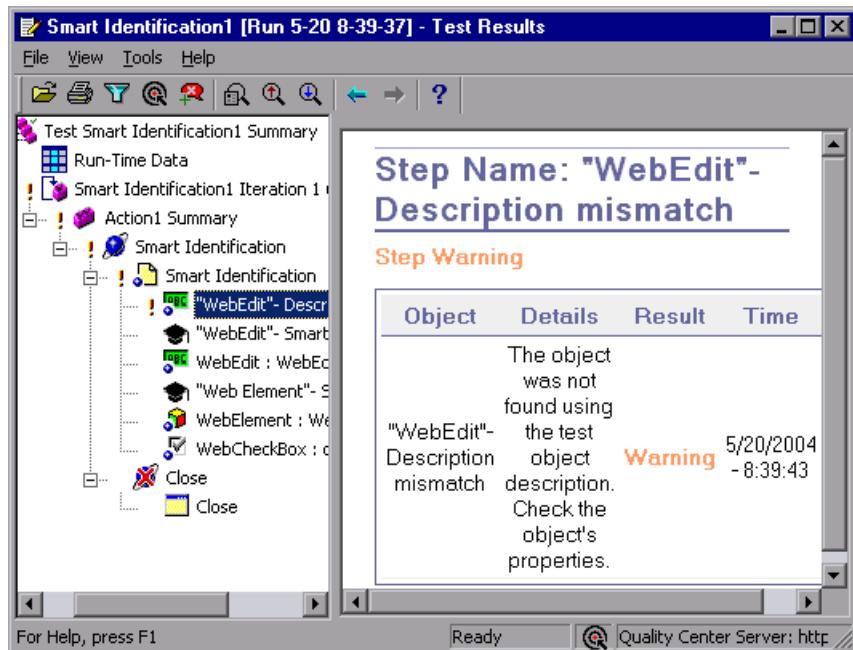
If the recorded description does not enable QuickTest to identify the specified object in a step, and a Smart Identification definition is defined (and enabled) for the object, then QuickTest tries to identify the object using the Smart Identification mechanism.

If QuickTest successfully uses Smart Identification to find an object after no object matches the recorded description, the test results receive a warning status and include the following information:

In the results tree:	In the results details:
A description mismatch icon for the missing object. For example:  "WebEdit"- Description mismatch	An indication that the object was not found.
A Smart Identification icon for the missing object. For example:  "WebEdit"- Smart Identification	An indication that the Smart Identification mechanism successfully found the object, and information about the properties used to find the object. You can use this information to modify the recorded test object description, so that QuickTest can find the object using the description in future run sessions.
The actual step performed.	Normal result details for the performed step.

For more information on the Smart Identification mechanism, see Chapter 33, “Configuring Object Identification.”

The image below shows the results for a test in which Smart Identification was used to identify the WebEdit object after one of the recorded description property values changed.

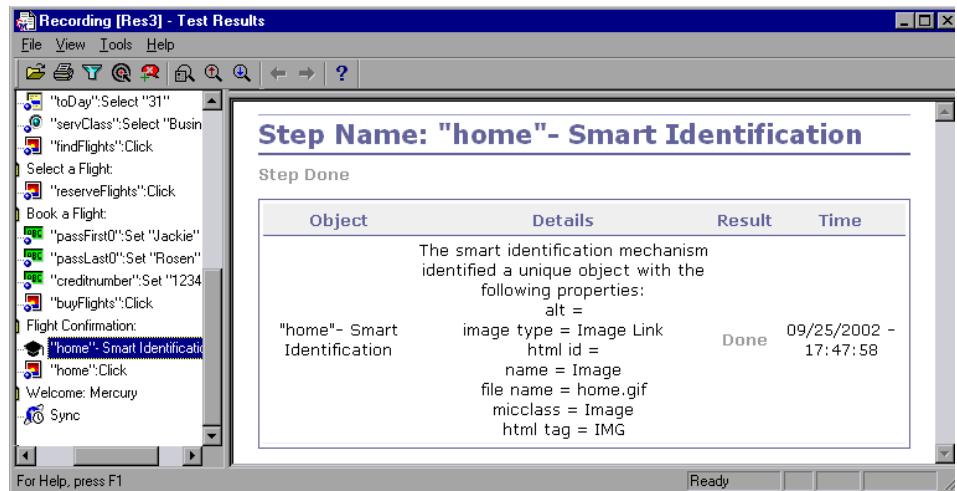


If QuickTest successfully uses Smart Identification to find an object after multiple objects are found that match the recorded description, QuickTest shows the Smart Identification information in the test results. The step still receives a passed status, because in most cases, if Smart Identification 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 results details:
A Smart Identification icon for the missing object. For example:  "home"- Smart Identification	An indication that the Smart Identification mechanism successfully found the object, and information about 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:  "home":Click	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 recorded description resulted in multiple matches.



If the Smart Identification mechanism cannot successfully identify the object, the test or component fails and a normal failed step is displayed in the test results.

Deleting Test 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) in order to directly delete results that meet criteria that you specify.

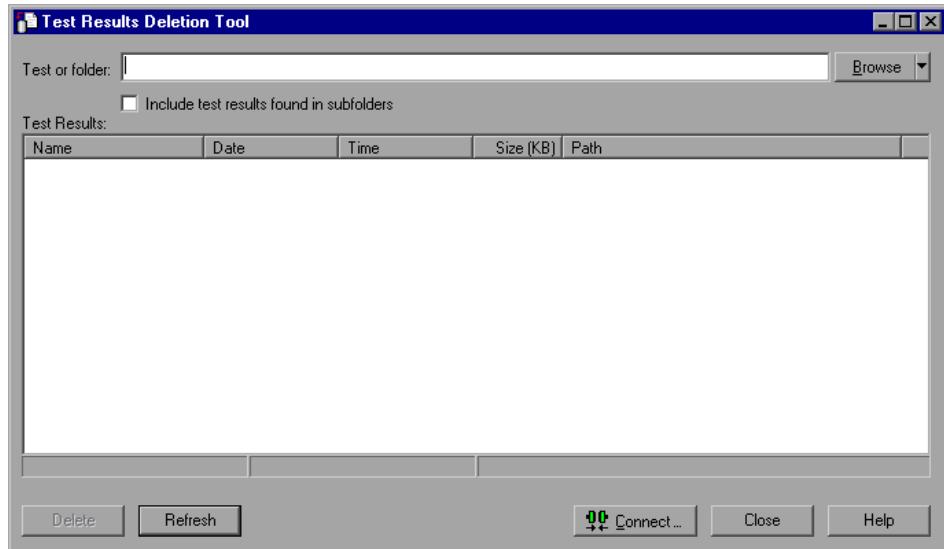
Deleting Results Using the Test Results Deletion Tool

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 easily identify the results you want to delete.

To delete test results using the Test Results Deletion Tool:

- 1 Choose **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 folder or specific test from which you want to delete test results. You can specify a full file system path or a full Quality Center path.

You can also browse to a test or folder as follows:

- To navigate to a specific test, click the **Browse** button or click the arrow to the right of the **Browse** button and select **Tests**.
- To navigate to a specific folder, click the arrow to the right of the **Browse** button and select **Folders**.

Note: To delete test results from a Quality Center database, click **Connect** to connect to Quality Center before browsing or entering the test path. Specify the Quality Center test path in the format [Quality Center] Subject\<folder name>\<test name>. For more information, see “Connecting to and Disconnecting from Quality Center” on page 946.

- 3** Select **Include test results found in subfolders** if you want to view all tests 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. For more information, see “Command Line Options,” below.

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, 2002.

```
TestResultsDeletionTool.exe -Silent -Test "C:\tests\test1" -FromDate "11/1/2002"
```

-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 name(s) of the result file(s) to be deleted. Only results with the specified name(s) are deleted.

You can use regular expressions to specify criteria for the result file(s) you want to delete. For more information about regular expressions and regular expression syntax, see “Understanding and Using Regular Expressions” on page 290.

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 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 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 deletes all results in the **F:\Tests\Keep\webobjects** test.

```
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"
```

Note: The **-Test** option can be combined with the **-Recursive** option to delete all test results in the specified 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, 2002.

```
TestResultsDeletionTool.exe -Silent -Test "C:\tests\test1" -UntilDate "11/1/2002"
```

-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 or component. 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 about working with Quality Center and QuickTest, see Chapter 40, “Working with Quality Center.” For additional information about Quality Center, refer to the *Mercury Quality Center User’s 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 Choose **Tools > Quality Center Connection** or click the **Quality Center Connection** button to connect to a Quality Center project. For additional information, see “Connecting to Quality Center from the Test Results Window,” below.



Note: If you do not connect to a Quality Center project before proceeding to the next step, QuickTest prompts you to connect before continuing.

- 2 Choose **Tools > Add Defect** or click the **Add Defect** button to open the Add Defect dialog box in the specified Quality Center project. The Add Defect dialog box opens.
- 3 You can modify the **Defect Information** if required. Basic information about the test or component and any checkpoints (if applicable) is included in the description:

```
Operating system : Windows 2000  
Test path : C:\Program Files\Mercury Interactive\QuickTest Professional\Tests\Tutorial\Recording on PREDATOR  
The CheckPoint 'roundtrip' Failed
```

- 4 Click **Submit** to add the defect information to the Quality Center project.
- 5 Click **Close** to close the Add Defect dialog box.

Connecting to Quality Center from the Test Results Window

To manually submit defects to Quality Center from the Test Results window, you must be connected to Quality Center.

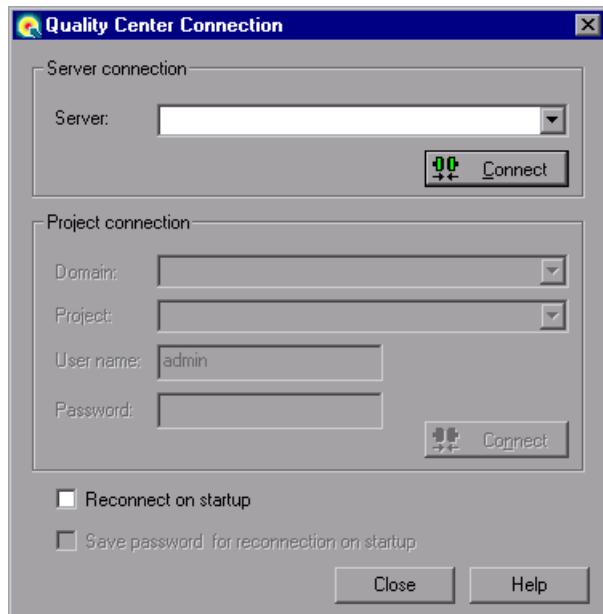
The connection process has two stages. First, you connect QuickTest to a local or remote Quality Center Web server. This server handles the connections between QuickTest and the Quality Center project.

Next, you choose the project in which you want to report the defects.

Note that Quality Center projects are password-protected, so you must provide a user name and a password.

To connect QuickTest to Quality Center:

- 1 Choose **Tools > Quality Center Connection** or click the **Quality Center Connection** button. The Quality Center Connection dialog box opens.



- 2 In the **Server** box, type the URL address of the Web server where Quality Center is installed.

Note: You can choose a Web server accessible via a Local Area Network (LAN) or a Wide Area Network (WAN).

- 3 Click **Connect**.

Once the connection to the server is established, the server's name is displayed in read-only format in the Server box.

- 4 If you are connecting to a project in TestDirector 7.6 or later, or Quality Center 8.0 or later, in the **Domain** box, select the domain which contains the TestDirector or Quality Center project.

If you are connecting to a project in TestDirector 7.2, skip this step.

- 5 In the **Project** box, select the desired project with which you want to work.
- 6 In the **User name** box, type a user name for opening the selected project.
- 7 In the **Password** box, type the password.
- 8 Click **Connect** to connect QuickTest to the selected project.

Once the connection to the selected project is established, the project's name is displayed in read-only format in the Project box.

- 9 To automatically reconnect to the Quality Center server and the selected project the next time you open QuickTest or the Test Results viewer, select the **Reconnect on startup** check box.
- 10 If the **Reconnect on startup** check box is selected, then the **Save password for reconnection on startup** check box is enabled. To save your password for reconnection on startup, select the **Save password for reconnection on startup** check box.

If you do not save your password, you will be prompted to enter it when QuickTest connects to Quality Center on startup.

- 11 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.

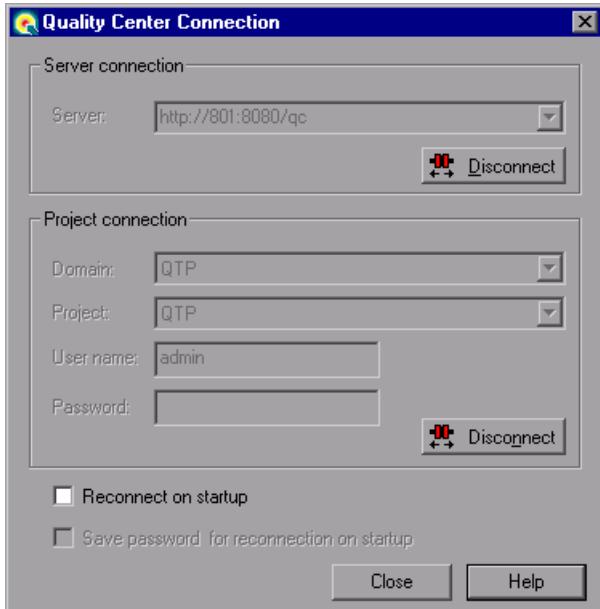


Tip: You can open the Quality Center Connection dialog box by double-clicking the **Quality Center** icon in the status bar.

You can disconnect from a Quality Center project and/or server. Note that if you disconnect QuickTest from a Quality Center server without first disconnecting from a project, QuickTest's connection to that project database is automatically disconnected.

To disconnect QuickTest from Quality Center:

- 1** Choose **Tools > Quality Center Connection** or click the **Quality Center Connection** button. The Quality Center Connection dialog box opens.



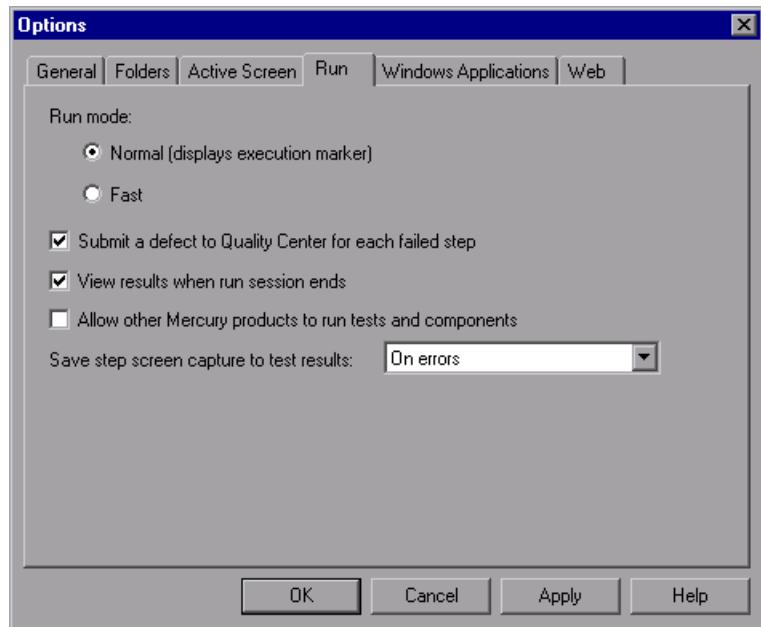
- 2** To disconnect QuickTest from the selected project, in the **Project connection** area, click **Disconnect**.
- 3** To disconnect QuickTest from the selected Web server, in the **Server connection** area, click **Disconnect**.
- 4** Click **Close** to close the Quality Center Connection dialog box.

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 (**Tools > Quality Center Connection**) for each failed step in your test or component.

To automatically submit defects to Quality Center:

-  1 Choose **Tools > Options** or click the **Options** button. The Options dialog box opens.
- 2 Click the Run tab.



- 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  
XML file Checkpoint "books.xml" failed  
Test name: checkpoint  
Test location: C:\Program Files\Mercury Interactive\QuickTest Professional\Tests\checkpoint  
Action name: Action1  
Operating system : Windows 2000  
Host: GEM
```

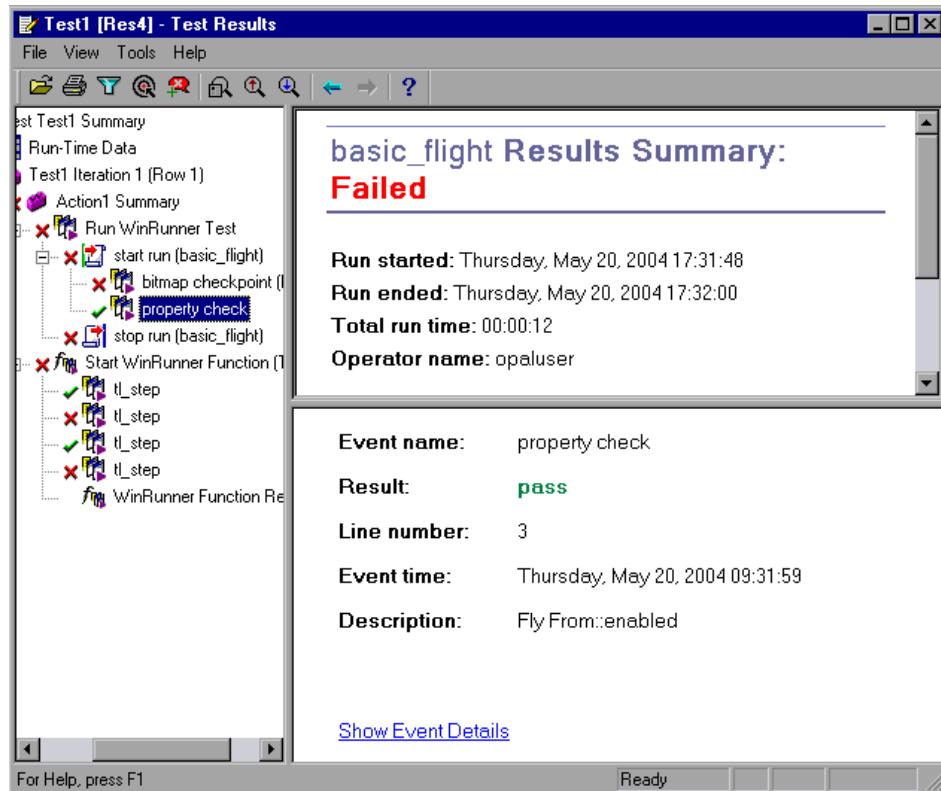
Viewing WinRunner Test Steps in the Test Results

If your QuickTest test includes a call to a WinRunner test and WinRunner 7.6 or later is installed on your computer, you can view detailed results of the WinRunner steps within your QuickTest Test Results window.

Note: You can view the WinRunner test steps of a result created using QuickTest Professional 8.0 only after you have installed patch **WR76P44 - Support WR/QTP integration** on WinRunner 7.6. This patch is available in the patch database on the Mercury Interactive Customer Support site (<http://support.mercury.com>).

Tip: If you have WinRunner version 7.5 installed on your computer, you can view basic information about the WinRunner test run in the QuickTest Test Results window. You can also open the WinRunner Test Results window for additional details.

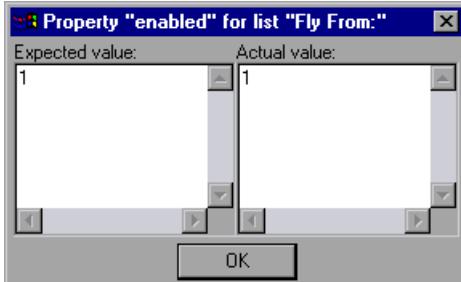
The left pane of the QuickTest test results include a node for each WinRunner event that would normally be included in the WinRunner results. When you select a node corresponding to a WinRunner test event or function call, the right pane displays a summary of the called WinRunner test or function and 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	<p>Displays additional information about the selected step followed by a link to the WinRunner details for the step.</p> <p>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:</p>  <p>Note: You must have WinRunner 7.6 or later installed on your computer to view WinRunner details for a selected step.</p>

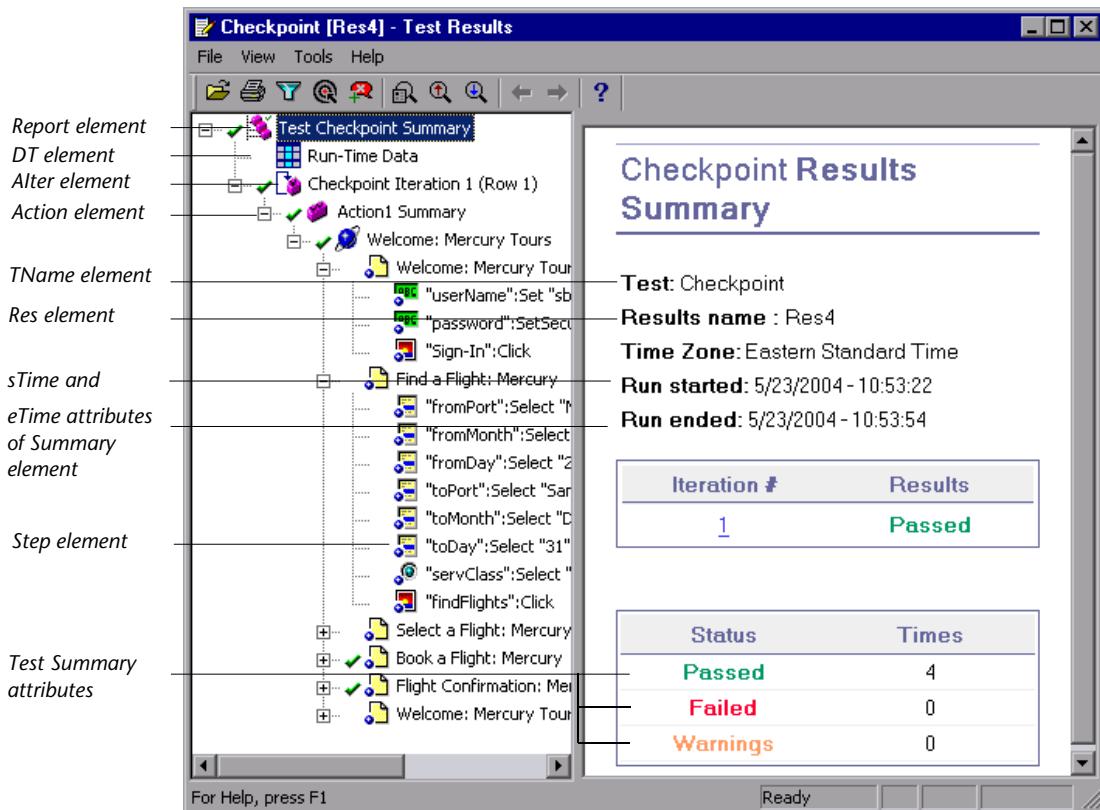
For more information on running WinRunner tests and functions from QuickTest, see Chapter 39, “Working with WinRunner.”

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 about 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 of the Test Results window.

Each node in the test 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 or when displaying test results in your own customized results viewer).

The diagram below shows the correlation between some of the elements in the **.xml** file and the items they represent in the test results.



XSL provides you with the tools to describe exactly which test result information to display and exactly where and how to display/print 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 about 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 when you select the **Short** option in the Print dialog box.
- **PDetails.xsl**—Specifies the content of the test results report printed when you select the **Detailed** option in the Print dialog box.
- **PSelection.xsl**—Specifies the content of the test results report printed when you select the **Selection** option in the Print dialog box.
- **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 555.

For information about the structure of the XML schema, and a description of the elements and attributes you can use to customize the test results reports, refer to the XML Report Help (located in <**QuickTest Professional installation folder**>\help\XMLReport.chm).

Part V

Configuring QuickTest

24

Setting Global Testing Options

You can control how QuickTest records and runs tests and components by setting global testing options.

This chapter describes:

- About Setting Global Testing Options
- Using the Options Dialog Box
- Setting General Testing Options
- Setting Folder Testing Options
- Setting Active Screen Options
- Setting Run Testing Options
- Setting Windows Application Testing Options
- Setting Web Testing Options

About Setting Global Testing Options

Global testing options affect how you record and run tests and components, as well as the general appearance of QuickTest. For example, you can choose not to display the Welcome screen when QuickTest starts, or set timing-related settings used by QuickTest when running a test or component. The values you set remain in effect for all tests and components and for subsequent testing sessions.

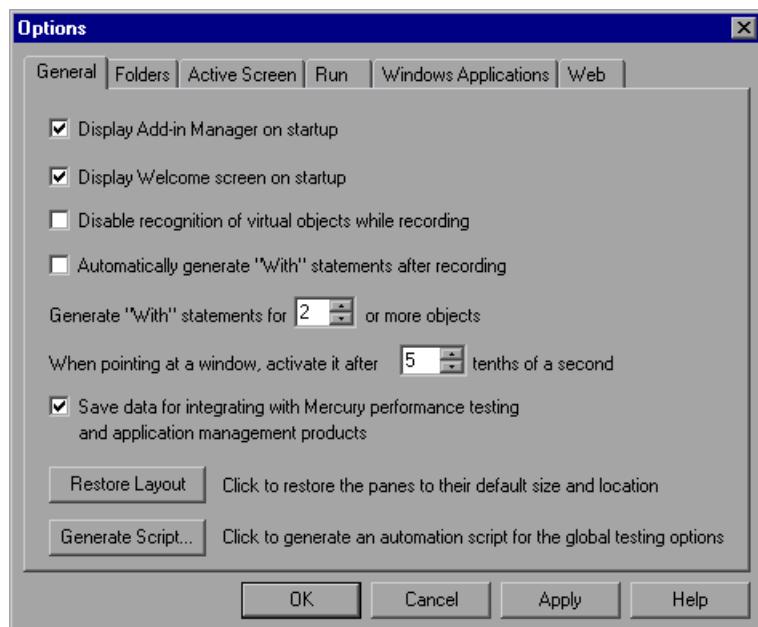
You can also set testing options that affect only the test or component currently open in QuickTest. For more information, see Chapter 25, “Setting Options for Individual Tests or Components.”

Using the Options Dialog Box

You can use the Options dialog box to modify your testing options. The values you set remain in effect for all subsequent record and run sessions.

To set global testing options:

- 1 Choose **Tools > Options** or click the **Options** toolbar button. The Options dialog box opens. It is divided by subject into several tabbed pages.



Note: The Web tab shown above is displayed only if the Web Add-in is installed and loaded.

- 2 Select the required tab and set the options as necessary. See the table below for more information on the available options in each tab.
- 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.

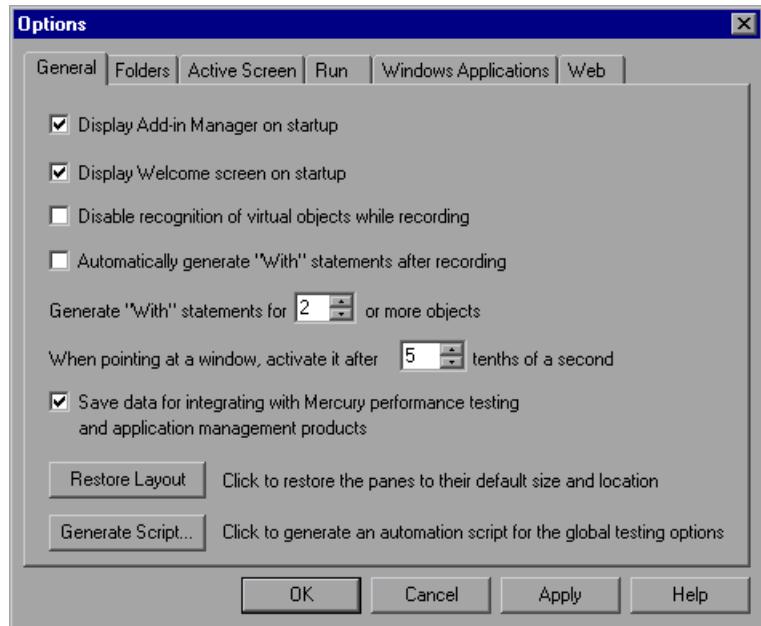
The Options dialog box can contain the following tabbed pages:

Tab Heading	Contains:
General	Options for general test settings. For more information, see “Setting General Testing Options” on page 614.
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 616.
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 618.
Run	Options for running tests. For more information, see “Setting Run Testing Options” on page 625.
Windows Applications	Options for configuring how QuickTest records and runs tests for the following Windows applications: <ul style="list-style-type: none"> • Standard Windows applications • .NET Windows Forms • Visual Basic • ActiveX For more information, see “Setting Windows Application Testing Options” on page 628.
Web (displayed only if the Web Add-in is installed and loaded)	Options for configuring recording and run session behavior in the Web environment. For more information, see “Setting Web Testing Options” on page 639.

The Options dialog box may contain additional tabs for any external add-ins that are currently installed and loaded.

Setting General Testing Options

The General tab options affect the general appearance of QuickTest and other general testing options.



The General tab 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 “Loading QuickTest Add-ins” on page 732.
Display Welcome screen on startup	Determines whether the Welcome screen is displayed when starting QuickTest.
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 16, “Learning Virtual Objects.”

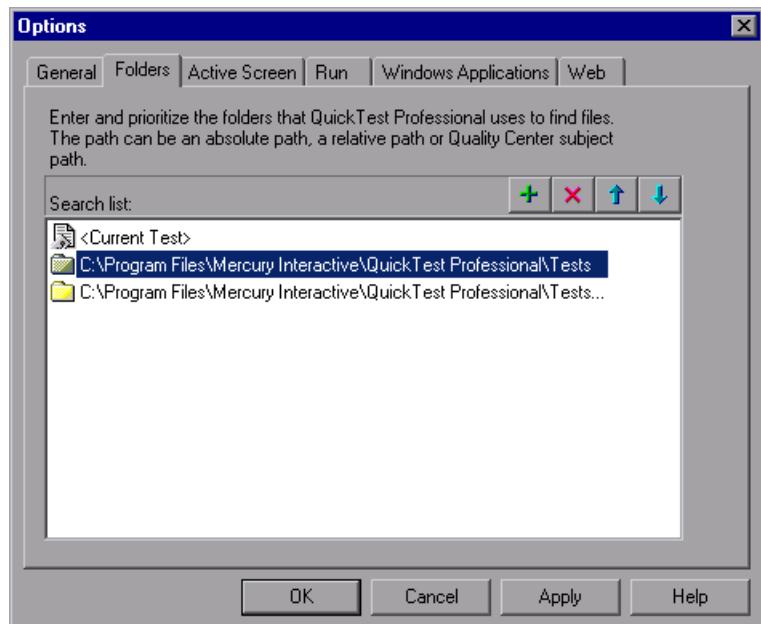
Option	Description
Automatically generate "With" statements after recording	Instructs QuickTest to automatically generate With statements when you record. For more information, see “Generating “With” Statements for Your Test or Component” on page 491.
Generate “With” statements for __ or more objects	<p>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.</p> <p>Default = 2.</p> <p>For more information, see “Generating “With” Statements for Your Test or Component” on page 491.</p>
When pointing at a window, activate it after __ tenths of a second	<p>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 on).</p> <p>Default = 5.</p>
Save data for integrating with Mercury performance testing and application management products	<p>Instructs QuickTest to save the relevant data to enable Mercury performance testing and application management products to use QuickTest tests.</p> <p>Note: The data is saved with each individual test.</p>
Restore Layout	Restores the layout of the QuickTest window so that it displays the panes in their default sizes and positions.
Generate Script	Generates an automation script containing the current global testing options. For more information, see “Automating QuickTest Operations” on page 919, or refer to the <i>QuickTest Automation Object Model Reference</i> (Help > QuickTest Automation Object Model Reference).

Setting Folder Testing Options

The Folders tab enables you to enter 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 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 tab.

Note: The current test or component is listed in the Search list by default. It cannot be deleted.

The order in which the folders are displayed in the search list determines the order in which QuickTest searches for the specified test, component, action, or file.



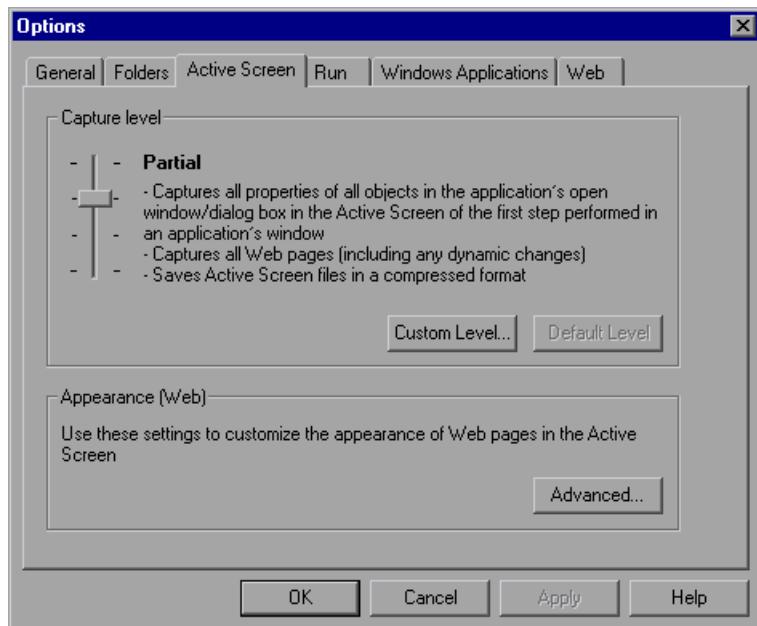
The Folders tab includes the following options:

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, component, 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.
	<p>Adds a new folder to the search list.</p> <p>Tip: 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.</p> <p>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.</p> <p>Note that QuickTest searches Quality Center project folders only when you are connected to the corresponding Quality Center project.</p>
	Deletes the selected folder from the search list.
	Moves the selected folder up in the list.
	Moves the selected folder down in the list.

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 tab. For more information, refer to the *QuickTest Professional Object Model Reference*.

Setting Active Screen Options

The Active Screen tab enables you to specify which information QuickTest saves and displays in the Active Screen while recording and running tests and components.



Tip: The more information saved in the Active Screen, the easier it is to edit the test or component 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 in Visual Basic, ActiveX, and .NET Windows Forms environments.

You can increase or decrease the amount of information saved in your test or component after it is recorded. For more information, see “Can I increase or decrease Active Screen information after I finish recording a test?” on page 1008.

Note: When you are recording on an MDI (Multiple Document Interface) application, the Active Screen does not capture information for non-active child frames.

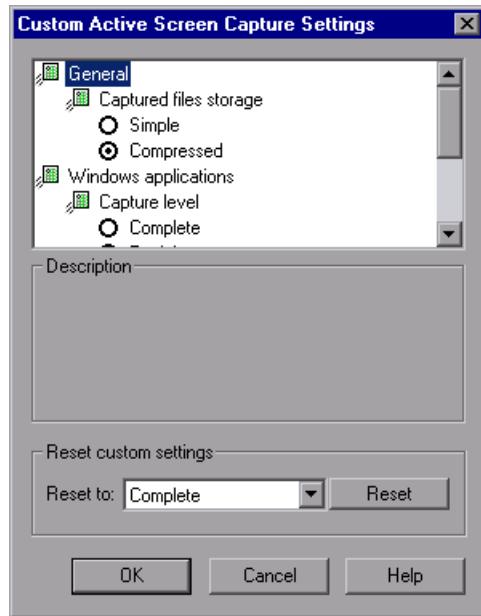
The Active Screen tab includes the following options:

Option	Description
Capture level	<p>Specifies the objects for which QuickTest stores data in the Active Screen. Use the slider to select one of the following options:</p> <ul style="list-style-type: none">• 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 “Custom Active Screen Capture Settings” on page 620.

Option	Description
Default Level	Returns the capture level settings to the predefined default level (Partial).
Advanced	Enables you to define the appearance of Web pages in the Active Screen, see “Web Page Appearance” on page 623.

Custom Active Screen Capture Settings

The Custom Active Screen Capture Settings dialog box enables you to customize how QuickTest captures and saves Active Screen information.



Note: The Custom Active Screen Capture Settings dialog box may also contain options applicable to any QuickTest external add-ins installed on your computer. For information regarding these options, refer to the documentation provided with the specific add-in.

General Options

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.

Windows Applications Options

You can specify which properties are captured for each object in a Windows application when it is captured for the Active Screen.

- **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.

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 96.

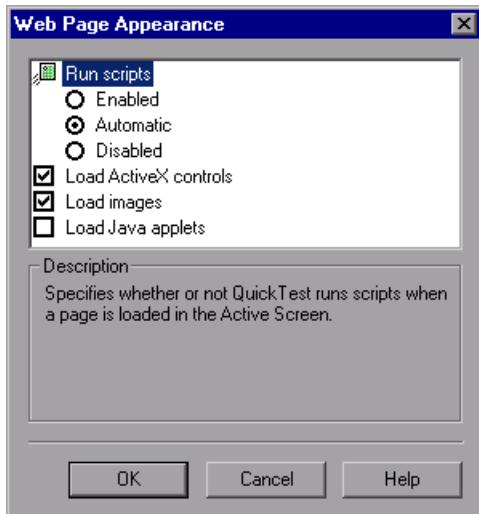
- **Capture original HTML source**—Captures the HTML source of Web pages as they appear originally, before any scripts are run. Deselecting this option instructs QuickTest to capture the HTML source of Web pages after any dynamic changes have been made to the HTML source (for example, by scripts running automatically when the page is loaded).

Reset Custom Settings

You can 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 **Reset**. For more information on the available capture levels, see “Windows Applications Options” on page 621.

Web Page Appearance

The Web Page Appearance dialog box enables you to modify how QuickTest displays captured Web pages in the Active Screen.



The Web Page Appearance dialog box contains the following options:

- **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:

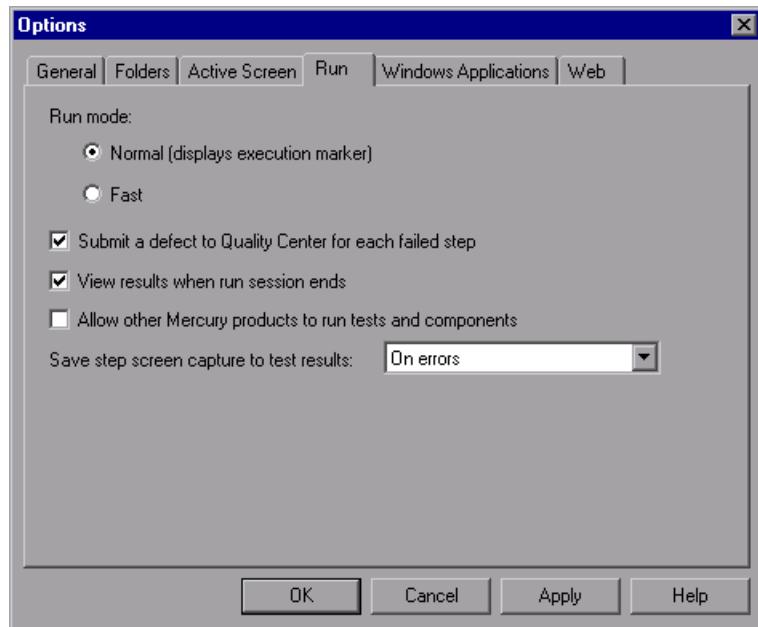
QuickTest loads ActiveX controls or Java applets to the Active Screen in view-only mode. You cannot perform operations or retrieve additional information on the loaded ActiveX or Java objects.

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.

Setting Run Testing Options

The Run tab options affect how QuickTest runs tests and components and displays test results.



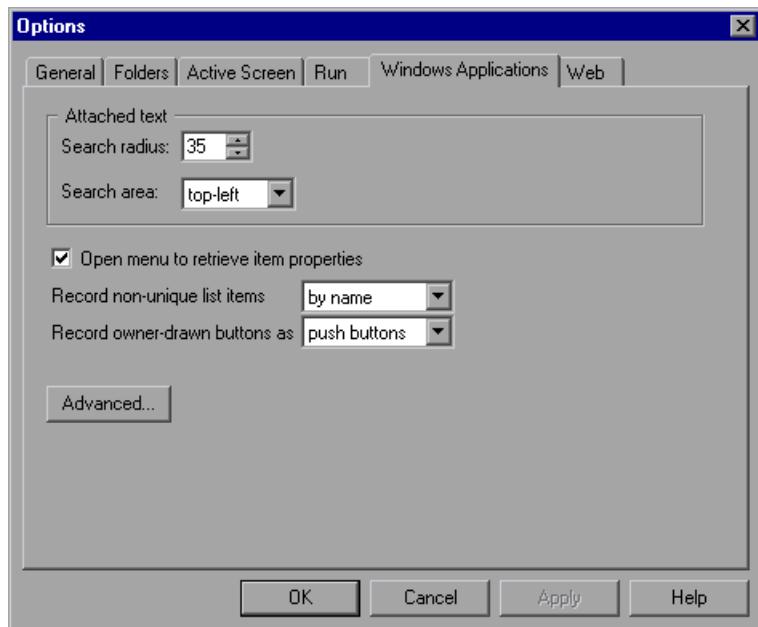
The Run tab includes the following options:

Option	Description
Run mode	<p>Instructs QuickTest how to run your test or component:</p> <p>Normal (displays execution marker)—Runs your test or component 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. This option requires more system resources.</p> <p>Note: You must have Microsoft Script Debugger installed to enable this mode. For more information, refer to the <i>QuickTest Installation Guide</i>.</p> <p>Fast—Runs your test or component 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.</p>
Submit a defect to Quality Center for each failed step	<p>Instructs QuickTest to automatically submit a defect to Quality Center for each failed step in your test or component. 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 602.</p>
View results when run session ends	<p>Instructs QuickTest to display the results automatically following the run session.</p>
Allow other Mercury products to run tests and components	<p>Enables other Mercury products such as Quality Center and Test Batch Runner to run QuickTest tests and components. This option is not required to enable WinRunner to run QuickTest tests and components.</p>

Option	Description
Save step screen capture to results	<p>Instructs QuickTest when to capture and save images of the application during the run session to display them in the results.</p> <p>Choose an option from the list:</p> <ul style="list-style-type: none">• Always—Captures images of each step, whether the step fails or passes.• On errors—Captures images only if the step fails.• On errors and warnings—Captures images only if the step returns a failed or warning status.• Never—Never captures images.

Setting Windows Application Testing Options

The Windows Applications tab options allow you to configure how QuickTest records and runs tests and components for Standard Windows, ActiveX, .NET Windows Forms, and Visual Basic applications.



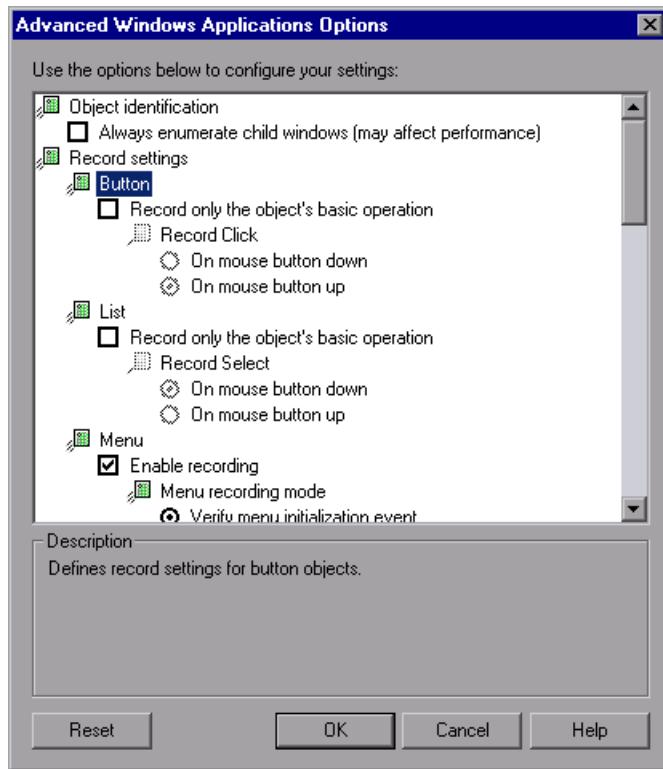
The Windows Applications tab includes the following options:

Option	Description
Attached text	<p>Enables you to specify the search criteria that QuickTest uses to retrieve an object's attached text. An object's attached text is the closest static text within a specified radius from a specified point. The retrieved attached text is saved in the object's corresponding text or attached text test object property.</p> <p>Note: Sometimes the static text that you believe to be closest to an object is not actually the closest static text. You may need to use trial and error to make sure that the attached text is the static text object of your choice.</p> <p>Search radius—Indicates the maximum distance, in pixels, that QuickTest searches for attached text.</p> <p>Search area—Indicates the point on an object from which QuickTest searches for the object's attached text.</p> <p>Choose an option from the list:</p> <ul style="list-style-type: none"> • Top-Left—top-left corner • Top—midpoint of the two top corners • Top-Right—top-right corner • Right—midpoint of the two right corners • Bottom-Right—bottom-right corner • Bottom—midpoint of the two bottom corners • Bottom-Left—bottom-left corner • Left—midpoint of the two left corners
Open menu to retrieve item properties	<p>Instructs QuickTest to open menu objects before retrieving menu item properties during a run session.</p> <p>Note: Selecting this option may slow the run, but it can be useful if menu item properties change upon opening the menu.</p> <p>This option, selected by default, sets the default behavior for all menu objects. You can use the ExpandMenu method to set this behavior for a specified menu object. For more information, refer to the <i>QuickTest Professional Object Model Reference</i>.</p>

Option	Description
Record non-unique list items	<p>Determines what QuickTest records when more than one item (in a list or tree) has an identical name.</p> <ul style="list-style-type: none"> • by name—Records the item's name. When the test or component runs, QuickTest finds and selects the first instance of the name, no matter which item was chosen during recording. Select this option if the all items with the same name have identical properties. • by index—Records the item's index number. Select this option if items with the same name do not necessarily have identical properties.
Record owner-drawn buttons as	<p>Instructs QuickTest how to identify and record custom-made buttons in the application.</p> <p>Choose an option from the list:</p> <ul style="list-style-type: none"> • push buttons • check boxes • radio buttons • objects <p>Note: Choosing objects records each owner-drawn button either as a WinObject or as a virtual object. For the latter, you must also define the virtual object. For more information, see “Defining a Virtual Object” on page 333.</p>
Advanced	<p>Opens the Advanced Windows Applications Options dialog box, where you can customize record and run options for your Windows applications. For more information, see “Advanced Windows Applications Options” on page 631.</p>

Advanced Windows Applications Options

The Advanced Windows Applications Options dialog box enables you to modify how QuickTest records and runs tests or components on Windows-based applications, such as ActiveX or Visual Basic. You can click the **Reset** button at any time to reset all options to their default settings.



Description
Defines record settings for button objects.

Reset

OK

Cancel

Help

Object Identification Options

You can specify the method QuickTest uses to identify objects when running a test or component.

The Advanced Windows Applications Options dialog box includes the following **Object identification** options:

Option	Description
Always enumerate child windows (may affect performance)	Instructs QuickTest to enumerate all child windows when recording and running a test or component. This option is cleared by default and should be used only when an object cannot otherwise be identified, because it may significantly affect performance. For more information, see “Advanced Information” on page 637.

Record Settings Options

You can specify how QuickTest treats certain objects when recording a test or component.

The Advanced Windows Applications Options dialog box includes the following **Record settings** options:

Category	Option
Button	Defines record settings for button objects: <ul style="list-style-type: none">• Record only the object’s basic operation—Enables simplified recording on the button. Using this mode may improve recognition of user operations in non-standard cases. This option is cleared by default and should be used only when the default recording method does not meet your needs. For more information, see “Advanced Information” on page 637.• Record Click—Specifies whether the Click operation should be recorded when the mouse button is pressed (On mouse button down) or when the mouse button is released (On mouse button up). This option is only enabled when Record only the object’s basic operation is selected. Default = On mouse button up.

Category	Option
List	<p>Defines record settings for Windows-based list objects (for example, WinList, WinListView, and VbList):</p> <ul style="list-style-type: none"> • Record only the object's basic operation—Enables simplified recording on the list. Using this mode may improve recognition of user operations in non-standard cases. This option is cleared by default and should be used only when the default recording method does not meet your needs. For more information, see “Advanced Information” on page 637. • Record Select—Specifies whether the Select operation should be recorded when the mouse button is pressed (On mouse button down) or when the mouse button is released (On mouse button up). This option is only enabled when Record only the object's basic operation is selected. Default = On mouse button up.
Menu	<p>Defines record settings for menu objects:</p> <ul style="list-style-type: none"> • Enable recording—Specifies whether QuickTest records operations on menu controls. For example, you may want QuickTest to ignore the actual process of selecting a menu to open another window. This option is selected by default. • Menu recording mode—Specifies whether QuickTest verifies or ignores menu initialization events before recording operations on menu controls. This option is only enabled when Enable recording is selected. Default = Verify menu initialization event. <p>For more information, see “Advanced Information” on page 637.</p>

Category	Option
Object	<p>Defines record settings for objects recognized as WinObject test objects:</p> <ul style="list-style-type: none"> • Record only the object's basic operation—Enables simplified recording on the WinObject test object. Using this mode may improve recognition of user operations in non-standard cases. This option is cleared by default and should be used only when the default recording method does not meet your needs. For more information, see “Advanced Information” on page 637. • Record Click—Specifies whether the Click operation should be recorded when the mouse button is pressed (On mouse button down) or when the mouse button is released (On mouse button up). This option is only enabled when Record only the object's basic operation is selected. Default = On mouse button down.
Tab	<p>Defines record settings for tab objects:</p> <ul style="list-style-type: none"> • Record only the object's basic operation—Enables simplified recording on the tab. Using this mode may improve recognition of user operations in non-standard cases. This option is cleared by default and should be used only when the default recording method does not meet your needs. For more information, see “Advanced Information” on page 637. • Record Select—Specifies whether the Select operation should be recorded when the mouse button is pressed (On mouse button down) or when the mouse button is released (On mouse button up). This option is only enabled when Record only the object's basic operation is selected. Default = On mouse button up.

Category	Option
Toolbar	<p>Defines record settings for toolbar objects:</p> <ul style="list-style-type: none"> • Record only the object's basic operation—Enables simplified recording on the toolbar. Using this mode may improve recognition of user operations in non-standard cases. This option is cleared by default and should be used only when the default recording method does not meet your needs. For more information, see “Advanced Information” on page 637. • Record Press—Specifies whether the Press operation should be recorded when the mouse button is pressed (On mouse button down) or when the mouse button is released (On mouse button up). This option is only enabled when Record only the object's basic operation is selected. Default = On mouse button up.
Tree view	<p>Defines record settings for tree view objects:</p> <ul style="list-style-type: none"> • Record only the object's basic operation—Enables simplified recording on the tree view. Using this mode may improve recognition of user operations in non-standard cases. This option is cleared by default and should be used only when the default recording method does not meet your needs. For more information, see “Advanced Information” on page 637. • Record Select—Specifies whether the Select operation should be recorded when the mouse button is pressed (On mouse button down) or when the mouse button is released (On mouse button up). This option is only enabled when Record only the object's basic operation is selected. Default = On mouse button up. • Record tree items—Specifies whether tree items are recorded By name or By virtual index. Default = By name.

Category	Option
Window	<p>Defines record settings for window objects:</p> <ul style="list-style-type: none"> • Record only the object's basic operation—Enables simplified recording on the window. Using this mode may improve recognition of user operations in non-standard cases. This option is cleared by default and should be used only when the default recording method does not meet your needs. For more information, see “Advanced Information” on page 637. • Record Click—Specifies whether the Click operation should be recorded when the mouse button is pressed (On mouse button down) or when the mouse button is released (On mouse button up). This option is only enabled when Record only the object's basic operation is selected. Default = On mouse button up.
Keyboard	<p>Defines record settings for operations performed on the keyboard:</p> <ul style="list-style-type: none"> • Keyboard state detection—Specifies which API QuickTest should use to detect the keyboard state. Default = Standard. For more information, see “Advanced Information” on page 637.
Utility object	<p>Defines record settings for utility objects:</p> <ul style="list-style-type: none"> • Record SystemUtil.Run commands—Specifies whether QuickTest records SystemUtil.Run commands when you open an application during a recording session. This option is selected by default. For more information on the SystemUtil.Run method, refer to the <i>QuickTest Professional Object Model Reference</i>.

Run Settings Options

You can specify how QuickTest treats certain objects when running a test or component.

The Advanced Windows Applications Options dialog box includes the following **Run settings** options:

Option	Description
Edit Box	<p>Defines run settings for Edit objects:</p> <ul style="list-style-type: none"> • Click Edit box before inserting text—Specifies whether QuickTest performs a Click operation to set the focus in an edit box before inserting text in it while running a test or component. This option is cleared by default. • Use keyboard events to perform Set operations—When selected, instructs QuickTest to simulate keyboard events when performing Set operations on edit boxes during a run session. When cleared, instructs QuickTest to use API or Window messages for edit box Set operations. This option is cleared by default.

Advanced Information

The following information is intended for users with expertise in the Win32 API and the Windows messages model. It expands on the information provided for some of the Advanced Windows Applications options in the previous section.

Always enumerate child windows

If QuickTest does not correctly record an object in your application, you can select this option to force QuickTest to enumerate all windows in the system. This means that even when QuickTest looks for a window without WS_CHILD style, it enumerates all windows in the system and not only the top-level windows.

You should select this option if there is a window in your application that does not have a WS_CHILD style but does have a parent (not an owner) window.

Record only the object's basic operation

In general, QuickTest records operations on Windows objects based on Windows messages sent by the application. QuickTest recognizes the sequence of Windows messages sent to a specific application window by the system, and uses a smart algorithm to determine which operation to record.

In rare cases (where a non-standard message sequence is used), the smart algorithm may record unwanted operations. Select this option if you want to record only the object's basic operation when the selected event occurs. When you select this option, you can also select when to record the operation. If you select **On mouse button down**, QuickTest records the operation performed when a WM_LBUTTONDOWN message is detected; if you select **On mouse button up**, QuickTest records the operation performed when a WM_LBUTTONUP message is detected.

Keyboard state detection

If QuickTest does not correctly record keyboard key combinations (for example, CTRL+Y, or ALT+CTRL+HOME), you can try changing the default setting for this option. Following is a brief explanation of each of the options:

- **Standard**—Uses the GetKeyboardState API to detect the keyboard state. For more information, see <http://msdn.microsoft.com/library/en-us/winui/winui/windowsuserinterface/userinput/keyboardinput/keyboardinputreference/keyboardinputfunctions/getkeyboardstate.asp>.
- **Alternate synchronous**—Uses the GetKeyState API to detect the keyboard state. For more information, see <http://msdn.microsoft.com/library/en-us/winui/winui/windowsuserinterface/userinput/keyboardinput/keyboardinputreference/keyboardinputfunctions/getkeystate.asp>.
- **Alternate asynchronous**—Uses the GetAsyncKeyState API to detect the keyboard state. For more information, see <http://msdn.microsoft.com/library/en-us/winui/winui/windowsuserinterface/userinput/keyboardinput/keyboardinputreference/keyboardinputfunctions/getasynckeystate.asp>.

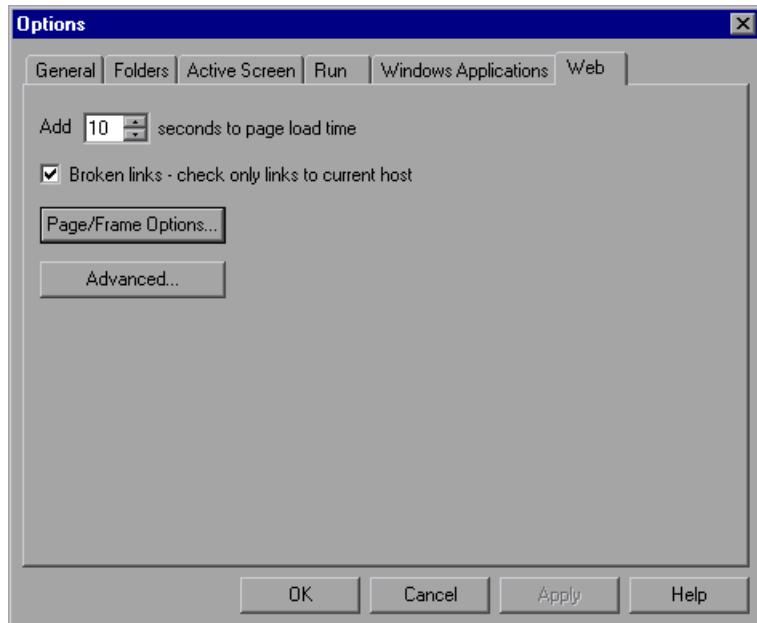
Menu recording mode

In most applications, Windows sends a WM_CONTEXTMENU message, WM_ENTERMENULOOP message, WM_INITMENU message, WM_INITMENUPOPUP message, or other initialization message when a user opens a menu. Windows then sends a WM_MENUSELECT message when a user selects a menu item.

The **Verify menu initialization** event option instructs QuickTest to record menu operations only after detecting a menu initialization message. If QuickTest does not correctly record menu operations, or if your application does not send initialization messages before sending WM_MENUSELECT messages, try using the **Ignore menu initialization** event option. This instructs QuickTest to always record menu operations.

Setting Web Testing Options

The Web tab options determine QuickTest's behavior when recording and running tests or components on Web sites.

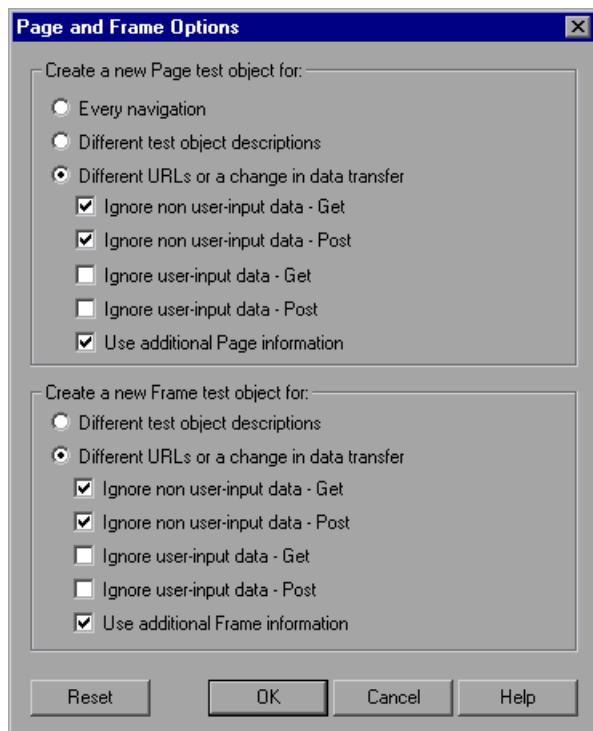


The Web tab includes the following options:

Option	Description
Add __ seconds to page load time	<p>Instructs QuickTest to add a specified number of seconds to the page load time property specified in each Page checkpoint.</p> <p>Note: This option is a safeguard that prevents page checkpoints from failing in the event that the amount of time it takes for a page to load during the run is longer than the amount of time it took during the record session.</p>
Broken links - check only links to current host	<p>Instructs QuickTest to check only for broken links that are targeted to your current host.</p>
Page/Frame Options	<p>Opens the Page and Frame Options dialog box, where you can customize how QuickTest records Page and Frame test objects. For more information, see “Page and Frame Options” on page 641.</p>
Advanced	<p>Opens the Advanced Web Options dialog box, where you can customize record and run options for your Web sites. For more information, see “Advanced Web Options” on page 644.</p>

Page and Frame Options

The Page and Frame Options dialog box enables you to modify how QuickTest records Page and Frame objects. You can click the **Reset** button at any time to reset all options to their default settings.



Page Options

The **Create a new Page test object for** options instruct QuickTest when to create a new Page object in the object repository while recording.

Note: These options only affect how Page test objects are created; Frame test objects are created according to the Frame options you select. For more information, see “Frame Options” on page 643.

The following Page options are available:

- **Every navigation**—Creates a new Page object every time you click a hyperlink on a Web page.
 - **Different test object descriptions**—Creates a new Page object for pages with different test object descriptions, according to the properties defined for the Page test object.
-

Note: The default test object description for Page objects includes only the test object class. If you select this option, it is highly recommended that you define object identification properties that uniquely identify different Page objects. You should also ensure that the properties you define remain constant over time, otherwise future runs may fail.

- **Different URLs or a change in data transfer**—(Default). Creates a new Page object only when the page URL changes, or if the URL stays the same and data that is transferred to the server changes, according to the data types and transfer methods you select (below).
-

Notes: Clear this option to instruct QuickTest to create a new Page test object for every navigation. (QuickTest version 5.6 and earlier worked this way automatically.)

- **Ignore non user-input data - Get**—Instructs QuickTest to ignore non user-input data if the **Get** method is used to transfer data to the server.

For example, suppose a user enters data on a Web page, and the data is then inserted as a hidden field using the **Get** method. The user clicks **Submit** (to send the data to the server). The new Web page is different, according to the hidden field data. However, QuickTest does not create a new Page test object.

- **Ignore non user-input data - Post**—Instructs QuickTest to ignore non user-input data if the **Post** method is used to transfer data to the server.

For example, suppose a user enters data on a Web page, and the data is then inserted as a hidden field using the **Post** method. The user clicks **Submit** (to send the data to the server). The new Web page is different, according to the hidden field data. However, QuickTest does not create a new Page test object.

- **Ignore user-input data - Get**—Instructs QuickTest to ignore user-input data if the **Get** method is used to transfer data to the server.

For example, suppose a user enters data in a form on a Web page and clicks **Submit** (to send the data to the server) using the **Get** method. The new Web page is different according to the data filled in by the user. However, QuickTest does not create a new Page test object.

- **Ignore user-input data - Post**—Instructs QuickTest to ignore user-input data if the **Post** method is used to transfer data to the server.

For example, suppose a user enters data in a form on a Web page and clicks **Submit** (to send the data to the server) using the **Post** method. The new Web page is different according to the data filled in by the user. However, QuickTest does not create a new Page test object.

- **Use additional Page information**—Instructs QuickTest to use additional properties of the test object to identify an existing Page test object.

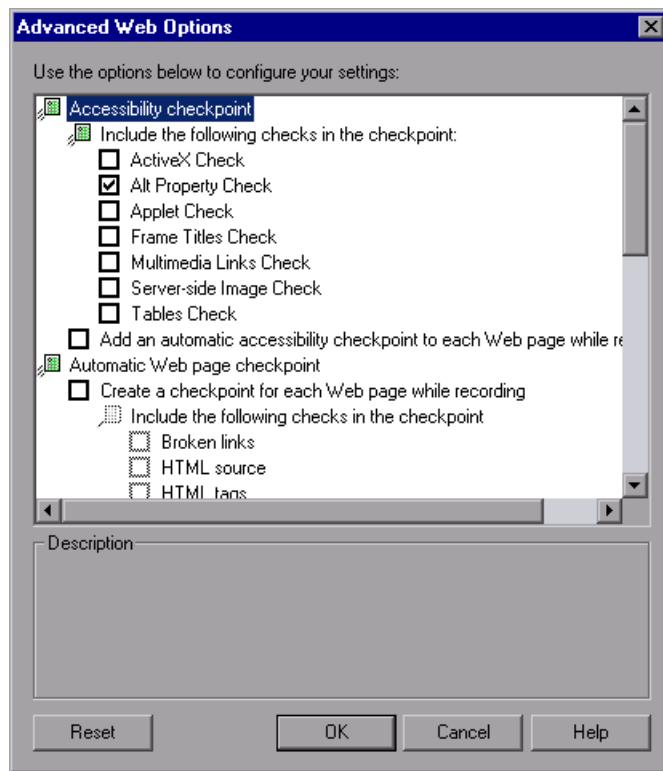
Tip: Select this option to instruct QuickTest to recognize existing pages when the **Back** and **Forward** navigation buttons are used.

Frame Options

The **Create a new Frame test object for** options instruct QuickTest when to create a new Frame object in the object repository while recording. The Frame options are similar to the Page options (except that the **Every navigation** option is not available). For more information, see “Page Options” on page 641.

Advanced Web Options

The Advanced Web Options dialog box enables you to modify how QuickTest records and runs tests and components on Web sites. You can click the **Reset** button at any time to reset all options to their default settings.



Accessibility Checkpoint Options

You can add accessibility checkpoints to check that Web pages and frames conform to the W3C Web Content Accessibility Guidelines.

Note: All accessibility checkpoints in a test or component use the options that are selected in this dialog box during the run session.

The Advanced Web Options dialog box includes the following **Accessibility checkpoint** options:

Option	Description
Include the following checks in the checkpoint	<p>Instructs QuickTest to check the selected accessibility elements for all accessibility checkpoints. Choose from the following:</p> <ul style="list-style-type: none">• ActiveX Check—Checks whether the page or frame contains ActiveX objects. If it does, QuickTest sends a warning and displays a list of the objects in the Test Results.• Alt Property Check—Checks that the <code><alt></code> attribute exists for all relevant objects (such as images). If one or more objects lack the required attribute, the test or component fails and QuickTest displays a list of the objects with the missing attribute in the Test Results. This option is selected by default.• Applet Check—Checks whether the page or frame contains Java objects. If it does, QuickTest sends a warning and displays a list of the objects in the Test Results.• Frame Titles Check—Checks that the page and all frames in the page have titles. If one or more frames (or the page) lack the required title, the test or component fails and QuickTest displays a list of the frames that lack titles in the Test Results.

Option	Description
Include the following checks in the checkpoint (continued)	<ul style="list-style-type: none"> • Multimedia Links Check—Checks whether the page or frame contains links to multimedia objects. If it does, QuickTest sends a warning and displays a list of the links in the Test Results. • Server-side Image Check—Checks whether the page or frame contains Server-side images. If it does, QuickTest sends a warning and displays a list of the images in the Test Results. • Tables Check—Checks whether the page or frame contains tables. If it does, QuickTest sends a warning and displays the table format and the tags used in each cell in the Test Results. <p>For more information, see “Checking Web Content Accessibility” on page 760.</p>
Add an automatic accessibility checkpoint to each Web page while recording	Instructs QuickTest to automatically add an accessibility checkpoint to each Web page while recording, using the checks selected in the option above.

Automatic Web Page Checkpoint Options

You can check that expected and actual page properties are identical. The Advanced Web Options dialog box includes the following automatic Page checkpoint options:

Option	Description
Create a checkpoint for each Web page while recording	Instructs QuickTest to automatically add a Page checkpoint for each Web page navigated during the recording process. Note: If you are testing a Web page with dynamic content, using automatic Page checkpoints may cause the test or component to fail as these checkpoints assume that the page content is static between record and run sessions.

Option	Description
Create a checkpoint for each Web page while recording (continued)	<p>All automatic Page checkpoints include the checks that you select from among the following options:</p> <ul style="list-style-type: none">• Broken links—Displays the number of broken links contained in the page during the run session. <p>Note: If the Broken links - check only links to current host option is selected (see “Setting Web Testing Options” on page 639), this number includes only those broken links that are targeted to the current host.</p> <ul style="list-style-type: none">• HTML source—Checks that the expected source code is identical to the source code during the run session.• HTML tags—Checks that the expected HTML tags in the source code are identical to those in the run session.• Image source—Checks that the expected source paths of the images are identical to the sources in the run session.• Links URL—Checks that the expected URL addresses for the links are identical to the URL addresses in the source code during the run session.• Load time—Checks that the expected time it takes for the page to load during the run session is less than or equal to the amount of time it took during the record session PLUS the amount of time specified in the Add seconds to page load time option (see “Setting Web Testing Options” on page 639).• Number of images—Checks that the expected number of images is identical to the number displayed in the run session.• Number of links—Checks that the expected number of links is identical to the number displayed in the run session.

Option	Description
Ignore automatic checkpoints while running tests or components	Instructs QuickTest to ignore the automatically added Page checkpoints while running your test or component.

Record Settings

You can set the preferences for recording Web objects. The Advanced Web Options dialog box includes the following Record options:

Option	Description
Enable Web support for Microsoft Windows Explorer	<p>When selected, QuickTest treats relevant objects in Microsoft Windows Explorer as Web objects. When cleared, QuickTest does not record events on Web pages displayed in Microsoft Windows Explorer.</p> <p>Note: After modifying this setting, for the change to take effect, you must close all instances of Microsoft Windows Explorer (confirm that all <code>explorer.exe</code> processes are closed in the Windows Task Manager or restart the computer) and then restart QuickTest.</p>
Record coordinates	Records the actual coordinates relative to the object for each operation.
Record MouseDown and MouseUp as Click	Records a Click method for MouseUp and MouseDown events.
Record Navigate for all navigation operations	Records a Navigate statement each time a Frame URL changes.
Use standard Windows mouse events	<p>Instructs QuickTest to use standard Windows mouse events instead of browser events for the following events:</p> <ul style="list-style-type: none"> • OnClick • OnMouseDown • OnMouseUp <p>Note: Use this option only if the events are not properly recorded using browser events.</p>

If QuickTest does not record Web events in a way that matches your needs, you can also configure the events you want to record for each type of Web object. For example, if you want to record events, such as a mouseover that opens a sub-menu, you may need to modify the your Web event configuration to recognize such events. For more information, see Chapter 35, “Configuring Web Event Recording.”

Run Settings

You can set the preferences for working with Web objects during a run session. The Advanced Web Options dialog box includes the following Run options:

Option	Description
Browser cleanup	Closes all open browsers when the current test or component closes. When this option is selected, all currently open browsers are closed when the current test or component closes, regardless of whether the browsers were opened before or after QuickTest.
Run only click	Runs Click events using the MouseDown event, the MouseUp event, and the Click event, or using only the Click event.
Replay type	Configures how to run mouse operations according to the selected option: <ul style="list-style-type: none">• Event—Runs mouse operations using browser events.• Mouse—Runs mouse operations using the mouse.
Run using source index	Uses the source index property for better performance.

Option	Description
Enable browser resize on record/run	<p>If this option is selected and you resize the browser during a recording session, QuickTest resizes the browser to this size when a subsequent run session begins. At the end of a run session, the browser returns to its default size.</p> <p>Note: To use this option, select the Open the following browser option in the Record and Run Settings dialog box before recording.</p> <p>When this option is cleared, QuickTest does not change the browser size when a run session begins.</p>

25

Setting Options for Individual Tests or Components

You can control how QuickTest records and runs different tests and components by setting specific testing options for any individual test or component.

This chapter describes:

- About Setting Options for Individual Tests or Components
- Using the Test Settings Dialog Box
- Using the Business Component Settings Dialog Box
- Defining Properties for Your Test
- Defining Properties for Your Component
- Defining Run Settings for Your Test
- Defining a Snapshot for Your Component
- Defining Application Settings for Your Component
- Defining Resource Settings for Your Test
- Defining Resource Settings for Your Component
- Defining Parameters for Your Test or Component
- Defining Environment Settings for Your Test or Component
- Defining Web Settings for Your Test or Component
- Defining Recovery Scenario Settings for Your Test or Component

About Setting Options for Individual Tests or Components

You can set testing options that affect how you record and run a specific test or component. 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 or the component.

The Test Settings and Business Component Settings dialog boxes are very similar, containing many options that are common to both tests and components. However, there are some important differences between them.

 Certain options and tabbed pages are available only when you are working with tests. For example:

- The Test Settings dialog box includes **Set as Default** options that enable you to define the current settings as the defaults for new tests. The default settings for new components are based on the settings defined in the Quality Center project's component template. For more information on default settings for components, see "Working with Component Templates" on page 987.
- Run session preference options are available only when working with tests. The Business Component Settings dialog box does not have a Run tab. For more information about the run settings for components, see "Running Components" on page 993.

 Certain options and tabbed pages are available only when you are working with components. For example:

- The options in the Applications tab enable you to specify the Windows-based applications on which the component can record.
 - You can use the Snapshot tab to capture an image and save it with the component for display in the business process test in Quality Center.
-

Note: When you are working with components, only the settings in the Business Component Settings dialog box apply. Settings that were made previously in the Test Settings dialog box, including those that were set as default settings, do not apply to components.

You can set testing options for specific parts of the test or component during the run session. For more information, see Chapter 28, “Setting Testing Options During the Run Session.”

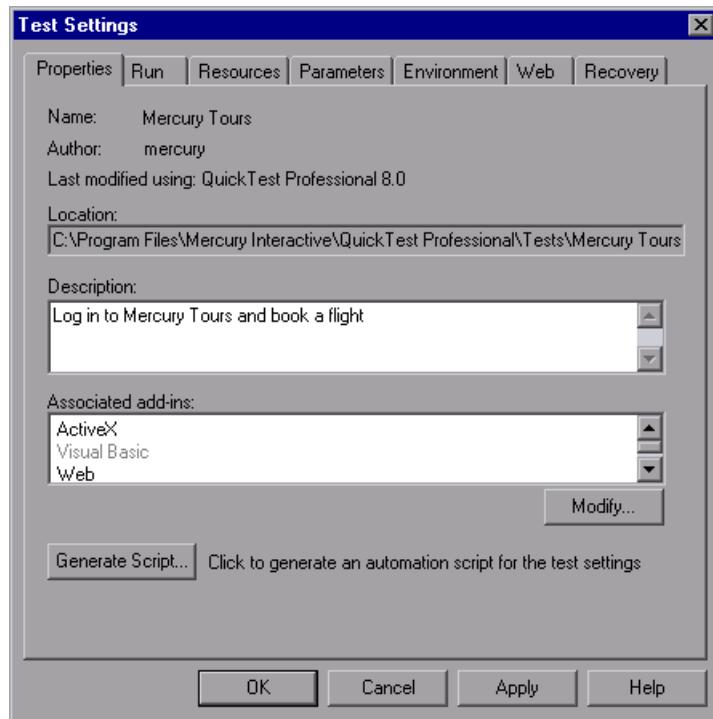
You can also set testing options that affect all tests and components. For more information, see Chapter 24, “Setting Global Testing Options.”

Using the Test Settings Dialog Box

 Before you record or run a test, you can use the Test Settings dialog box to modify your testing options for the specific test.

To set testing options for an individual test:

-  1 Choose **Test > Settings** or click the **Test Settings** toolbar button. The Test Settings dialog box opens. It is divided by subject into tabbed pages.



- 2 Select the required tab and set the options as necessary. See the table below for more information on the available options in each tab.
- 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 Test Settings dialog box contains the following tabbed pages:

Tab Heading	Tab Contents
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 659.
Run	Options for setting the run session preferences. For more information, see “Defining Run Settings for Your Test” on page 665.
Resources	Options for specifying resources you want to associate with your test, such as function libraries stored in VBScript library files. For more information, see “Defining Resource Settings for Your Test” on page 674.
Parameters	Options for specifying input and output parameters for your test. For more information, see “Defining Parameters for Your Test or Component” on page 682.
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 or Component” on page 685.
Web (displayed only if the Web Add-in is installed and loaded)	Options for setting how the test records and runs on a Web browser. For more information, see “Defining Web Settings for Your Test or Component” on page 692.
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 or Component” on page 694.

In addition to these tabs, the Test Settings dialog box may contain other tabs corresponding to any external add-ins that are loaded. For more information on external add-ins, refer to the relevant QuickTest add-in documentation.

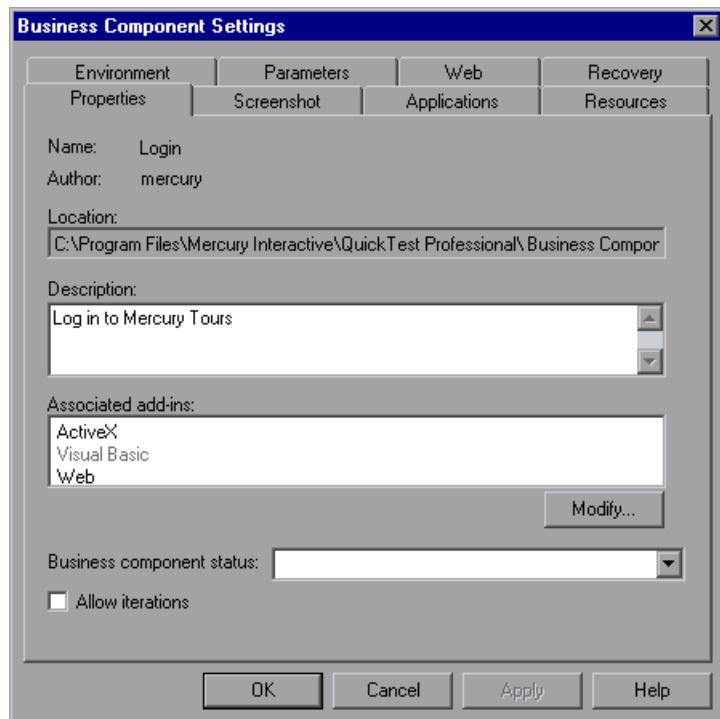
Using the Business Component Settings Dialog Box

 Before you record or run a component, you can use the Business Component Settings dialog box to modify your testing options for the specific component.

When you open a new component, the Business Component Settings dialog box displays the currently defined default settings. Many of these settings are based on the defaults defined in the Quality Center project's component template. For more information on component template settings, see "Working with Component Templates" on page 987.

To set testing options for an individual component:

- 1 Choose **Component > Settings** or click the **Business Component Settings** toolbar button. The Business Component Settings dialog box opens. It is divided by subject into tabbed pages.



- 2 Select the required tab and set the options as necessary. See the table below for more information on the available options in each tab.
- 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 Business Component Settings dialog box contains the following tabbed pages:

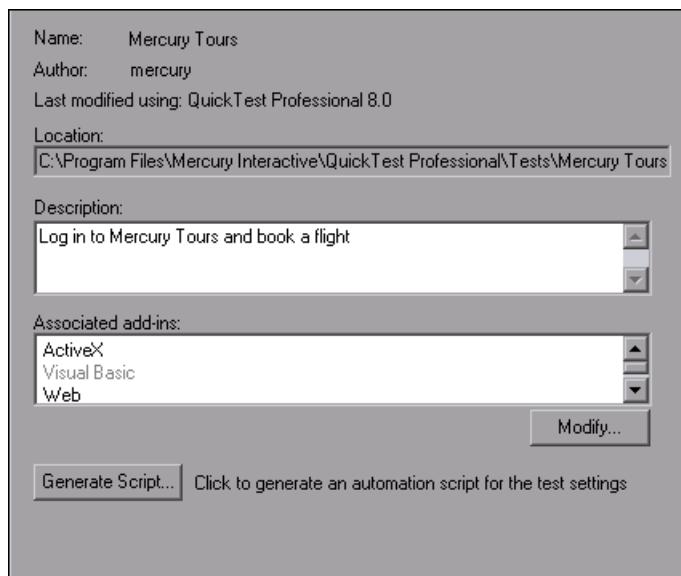
Tab Heading	Tab Contents
Properties	Options for setting the properties of the component, for example, its description and associated add-ins. For more information, see “Defining Properties for Your Component” on page 660.
Snapshot	Options for capturing a snapshot image to be saved with the component for display in Quality Center. For more information, see “Defining a Snapshot for Your Component” on page 669.
Applications	Options for specifying the Windows-based applications on which the component can record. For more information, see “Defining Application Settings for Your Component” on page 670.
Resources	Options for specifying resources you want to associate with your component, such as the location of the shared object repository to use with your component. For more information, see “Defining Resource Settings for Your Component” on page 680.
Parameters	Options for specifying input and output parameters for your component. For more information, see “Defining Parameters for Your Test or Component” on page 682.
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 or Component” on page 685.

Tab Heading	Tab Contents
Web (displayed only if the Web Add-in is installed and loaded)	Options for setting how the component records and runs on a Web browser. For more information, see “Defining Web Settings for Your Test or Component” on page 692.
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 or Component” on page 694.

In addition to these tabs, the Business Component Settings dialog box may contain other tabs corresponding to any external add-ins that are loaded. For more information on external add-ins, refer to the relevant QuickTest add-in documentation.

Defining Properties for Your Test

You can use the Properties tab of the Test Settings dialog box 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.



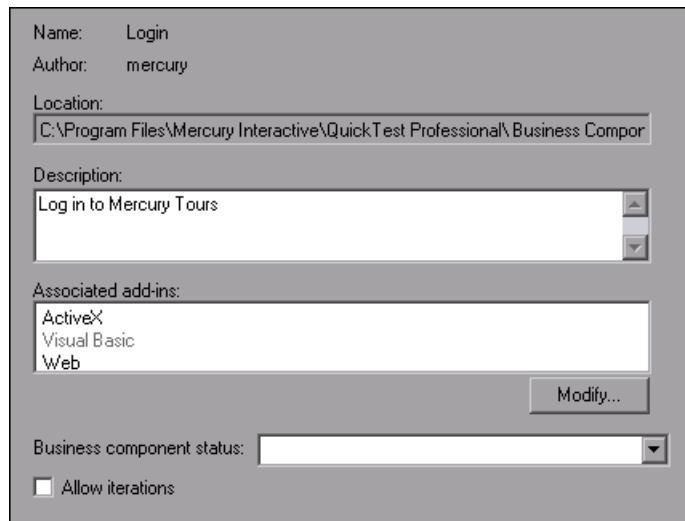
The Properties tab of the Test Settings dialog box includes the following items:

Option	Description
Name	Indicates the name of the test.
Author	Indicates the user name of the person who created the test.
Last modified using	Indicates the version of QuickTest last used to modify the test.
Location	Indicates the path and filename of the test.
Description	Enables you to specify a description for your test.

Option	Description
Associated add-ins	Displays the add-ins associated with the test. For more information, see “Associating Add-ins with Your Test or Component” on page 662.
Modify	Enables you to select the add-ins to associate with your test. For more information, see “Modifying Associated Add-Ins” on page 663.
Generate Script	Generates an automation script containing the current test settings. For more information, see “Automating QuickTest Operations” on page 919, or refer to the <i>QuickTest Automation Object Model Reference</i> (Help > QuickTest Automation Object Model Reference).

Defining Properties for Your Component

You can use the Properties tab of the Business Component Settings dialog box to view and define general information about your component, including the add-ins associated with it. You can also set the status of the component, and you can choose whether Quality Center can specify iterations for the component.



The Properties tab of the Business Component Settings dialog box includes the following items:

Option	Description
Name	Indicates the name of the component.
Author	Indicates the user name of the person who created the component.
Location	Indicates the path and filename of the component.
Description	Indicates the description specified for your component.
Associated add-ins	Displays the add-ins associated with the component. For more information, see “Associating Add-ins with Your Test or Component” on page 662.
Modify	Enables you to select the add-ins to associate with your component. For more information, see “Modifying Associated Add-Ins” on page 663.
Status	Specifies the status of the component. You can change the status of the component by selecting a different option from the list. For more information about status options, see “Defining the Status for a Component”, below.
Allow iterations	Controls whether iterations can be specified for this component in a Quality Center business process test. Component iterations are defined in Quality Center, using the Components tab. However, some components may not be designed for multiple iterations, if the last step in the component does not end at the same point in the application as the first step. In such cases, you can clear the Allow iterations check box to prevent iterations from being specified for the component in Quality Center.

Defining the Status for a Component

The following component status options are provided when Quality Center is installed with Business Process Testing support:

- **Error**—The component contains errors that need to be fixed, for example, due to a change in the application. When a business process test contains a component with this status, the status of the entire business process test is also **Error**.
- **Maintenance**—The component is currently being developed and tested and is not yet ready to run, or it was previously implemented and is now being modified to adapt it for changes that have been made in the application.
- **Ready**—The component is fully implemented and ready to be run. It answers the requirements specified for it and has been tested according to the criteria defined for your specific system.
- **Under Development**—The component is currently under development. This status is initially assigned to:
 - New components created in the Business Components module of Quality Center with Business Process Testing support.
 - Component requests dragged into the component tree in Quality Center with Business Process Testing support.

Associating Add-ins with Your Test or Component

When you open QuickTest, you select the add-ins to load from the Add-in Manager dialog box. You can record on 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. When you create a new component, the default associated add-ins are those defined in the component template. For more information on default settings for components, see “Working with Component Templates” on page 987.

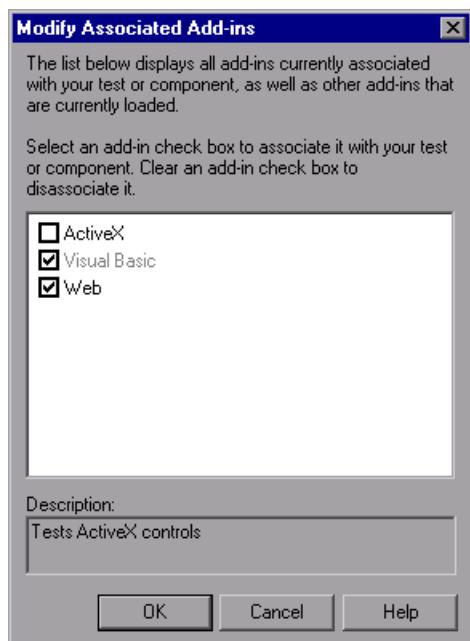
Choosing to associate an add-in with your test or component instructs QuickTest to check that the associated add-in is loaded each time you open that test or component.

When you open a test or component, 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 or component. 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 or component.

Quality Center uses the associated add-ins list to determine which add-ins to load when it opens QuickTest. For more information on working with Quality Center, see Chapter 40, “Working with Quality Center.”

Modifying Associated Add-Ins

You can associate or disassociate add-ins with your test or component in the Modify Associated Add-ins dialog box.



This dialog box lists all the add-ins currently associated with your test or component, as well as any other add-ins that are currently loaded in QuickTest. Add-ins that are associated with your test or component but not currently loaded are shown dimmed.

You can select the check boxes for add-ins that you want to associate with your test or component, or clear the check boxes for add-ins that you do not want to associate with your test or component.

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 or component, 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 tab of the Options dialog box.

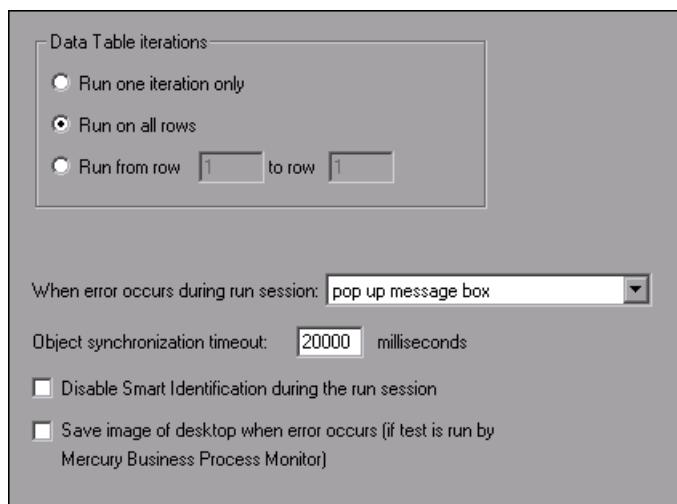
See Chapter 24, “Setting Global Testing Options” for more information on the Options dialog box. For more information on the Add-in Manager, see Chapter 29, “Working with QuickTest Add-Ins.”

You can also retrieve this list and load add-ins accordingly using an automation script. For more information on working with automation scripts, refer to the *QuickTest Automation Object Model Reference*.

Defining Run Settings for Your Test

When you run a test, QuickTest performs the steps you recorded on your application or Web site.

You can use the Run tab in the Test Settings dialog box 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 220.

You can use the Run tab to instruct QuickTest to run iterations on a test only for certain lines in the Global tab in the Data Table.

Note: The Run tab 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 about action run properties, see “Setting the Run Properties for an Action” on page 377.

The Run tab includes the following options:

Option	Description
Data Table iterations	<p>Specifies the iterations for the test. Choose an option:</p> <ul style="list-style-type: none">• 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	<p>Specifies how QuickTest responds to an error during the run session. For more information, see “Specifying the Response to an Error” on page 667.</p>
Object synchronization timeout	<p>Sets the maximum time (in milliseconds) that QuickTest waits for an object to load before running a step in the test.</p> <p>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 “Defining Web Settings for Your Test or Component” on page 692.</p>

Option	Description
Disable Smart Identification during the run session	<p>Instructs QuickTest not to use the Smart Identification mechanism during the run session.</p> <p>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.</p>
Save image of desktop when error occurs (if test is run by Mercury Business Process Monitor)	<p>This option is applicable only to tests that are run by the Business Process Monitor component of Mercury Application Management (formerly Topaz ActiveAgent). 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 Mercury Business Process Monitor. The image is saved in Application Management. The Business Process Monitor forwards the run results to the Application Management servers.</p>

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 or component when an error occurs.

QuickTest first performs any recovery scenarios associated with the test or component, 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 or Component” on page 694.

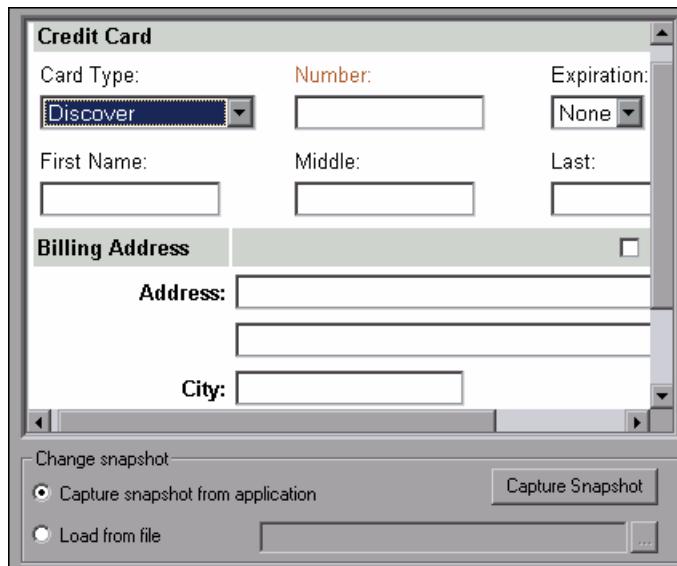
Note: This option replaces the global option found in QuickTest versions 6.0 and earlier. When you open a test created in QuickTest 6.0 and earlier, the **pop up message box** option is automatically selected by default.

To use a different setting for a large number of tests, you can use a QuickTest automation script to set this value. To easily access the automation script line that controls this option, you can use the **Generate Script** button in the Properties tab of the Test Settings dialog box.

For more information, see “Automating QuickTest Operations” on page 919, or refer to the *QuickTest Automation Object Model Reference* (**Help > QuickTest Automation Object Model Reference**).

Defining a Snapshot for Your Component

The Snapshot tab of the Business Component Settings dialog box enables you to capture an image and save it with the component. This image is displayed in the business process test in Quality Center and provides a visual indication of the main purpose of the component.



Note: The snapshot image can also be captured and saved with the component from the Components tab in Quality Center when installed with Business Process Testing support.

For more information about business process testing, see Chapter 41, "Working with Business Process Testing."

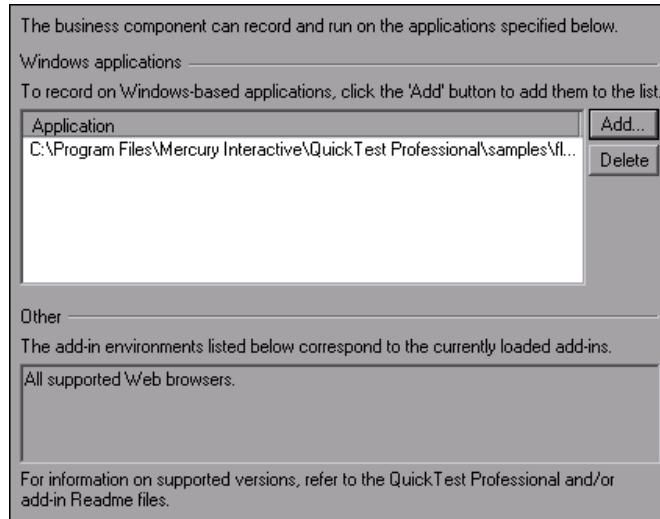
The Snapshot tab contains the following options:

Option	Description
Capture snapshot from application	Enables you to define the image to be captured by clicking the Capture Snapshot button. You can then drag the crosshairs pointer to select the area to be captured. When you release the mouse button, the captured area is displayed in the Snapshot pane.
Load from file	Specifies the .png or .bmp file containing the required image. You can enter the path and filename or use the browse button to locate the file.

When you click **Apply** or **OK**, the image is saved with the component and is displayed in the business process tests containing this component in Quality Center.

Defining Application Settings for Your Component

In the Applications tab of the Business Component Settings dialog box, you can specify the Windows-based applications on which the component can record. You can record steps only on the specified applications.



You can also view other environments on which the component can currently record (based on the currently loaded add-ins).

You can use the Applications tab to set or modify your application preferences in the following scenarios:

- You have already recorded one or more steps in the component and you want to modify the settings before you continue recording.
- You want to run the component on a different application than the one you previously used.

If you are recording a new component and have not yet set your application settings in the Applications tab of the Business Component Settings dialog box, the Applications dialog box opens with the same options as in the Applications tab. For more information on recording components, see “Recording Components” on page 991.

Note: To record on an application, you must open it manually. There are no settings available for automatically opening applications for components.

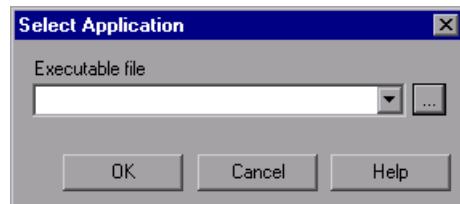
The following options are available in the Applications tab:

Option	Description
Application	<p>Lists the applications on which to record the component.</p> <p>If you do not want to record on any Windows application, leave the Application area blank. (This is the default setting.)</p> <p>You can manually add standard Windows steps to your component and then run them, even if the Application area is left blank (or does not list the application for which you want to add a step).</p> <p>Note: If you define values for one or more APP_ENV environment variables, those values override the values specified in the Application area during a run session. For more information, see “Using Environment Variables to Specify Windows-based Applications” on page 672.</p>

Option	Description
Add	Adds an application to the application list. You can add up to ten applications. For more information, see "Specifying an Application for a Component," below.
Delete	Removes the selected application from the Application area.

Specifying an Application for a Component

In the Select Application dialog box you can specify the executable file on which you want QuickTest to record your component.



You can enter the path and filename of the executable file in the **Executable file** box, or click the browse button to locate the file.

Using Environment Variables to Specify Windows-based Applications

You can use special, predefined environment variables to specify the applications you want to record on for your component. This can be useful if you want to test how your application works in different environments. For example, you may want to test that similar or localized versions of your application work properly.

When you define an environment variable for one or more applications before recording, the environment variable value overrides the value specified in the Applications dialog box and tab.

You can define the environment variables as internal user-defined variables, or you can add them to an external environment variable file and set your component to load environment variables from that file.

To use application environment variables for your component:

- 1** Define an environment variable for each application executable file you want to set using the appropriate variable name.

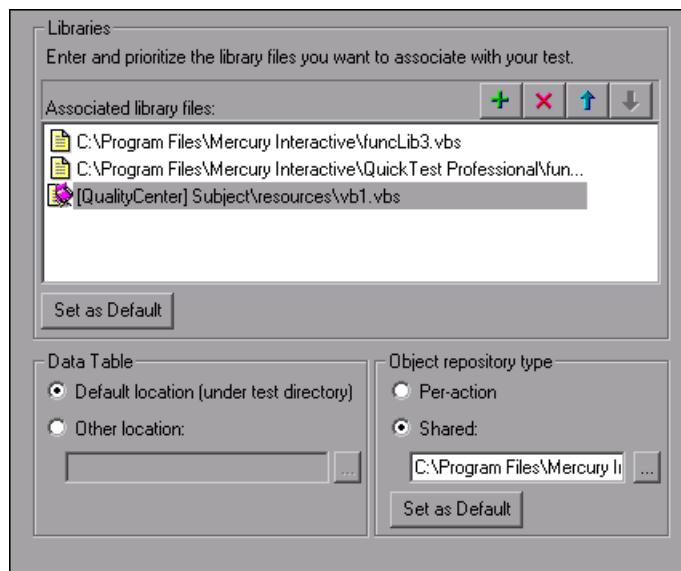
Option	Variable Names	Description
Application	1_APP_ENV, 2_APP_ENV, ... 10_APP_ENV	The executable files (up to ten application names) on which QuickTest records operations.

For more information on how to define a user-defined environment variable and how to create environment variable files, see “Using Environment Variable Parameters” on page 222.

- 2** Run the component. QuickTest uses the environment variables to determine on which application(s) you can record.

Defining Resource Settings for Your Test

You can use the Resources tab of the Test Settings dialog box to associate specific files with your test, such as VBScript library files and Data Table files, and to specify the object repository mode and file to use for your test. You can also set the currently associated library files and object repository settings as the default settings for all new tests.



The Resources tab in the Test Settings dialog box includes the following option areas:

Option Area	Description
Libraries	Displays the list of library files associated with your test. You can add, delete, and prioritize the files. You can also set the default library files for new tests. For more information, see “Specifying Associated Library Files” on page 677.
Data Table	<p>Specifies the location of the Data Table to be used in your test:</p> <ul style="list-style-type: none">• 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. <p>For more information on choosing a Data Table location, see “Editing the Data Table” on page 400.</p> <p>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 408.</p>
Object repository type	<p>Specifies the object repository mode and file to use for your test, and enables you to set the default object repository mode and/or file for new tests.</p> <p>Note: These options are enabled only if the test does not call any external actions and does not contain any steps or objects.</p> <p>For more information, see “Specifying Object Repository Settings”, below.</p>

Specifying Object Repository Settings

The **Object repository type** option area in the Test Settings dialog box includes the following options:

- **Per-action**—Instructs QuickTest to use the object repository per-action mode for your test. A repository file is stored with each action.
- **Shared**—Instructs QuickTest to use the shared object repository mode and the specified object repository file for your test.

You can enter an existing shared object repository file as an absolute or relative path. If you enter a relative path, QuickTest searches for the file in the current test's directory, and then in the folders listed in the Folders tab of the Options dialog box. For more information, see “Setting Folder Testing Options” on page 616.

Note: If your object repository file is stored in the file system and you want other users or Mercury products to be able to run this test on other computers, you should 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. For more information, see “Setting Folder Testing Options” on page 616.

To create a new object repository file, enter the complete file system or Quality Center path of the new file. For more information, see “Using Shared Object Repository Files with Quality Center” on page 828.

- **Set as Default**—Sets the currently specified object repository mode and/or file to be the default for new tests. The **Set as Default** button is enabled when the setting for this test is different than the default for all tests.

For more information about object repository modes, see Chapter 34, “Choosing the Object Repository Mode.”

Specifying Associated Library Files

The **Associated library files** pane of the Resources tab indicates the list of library files associated with your test or component. QuickTest searches these files for the VBScript functions, subroutines, etc. specified in the test or component.

The order of the library files in the list determines the order in which QuickTest searches for a function or subroutine that is called from a step in your test or component. 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 Library Files” on page 908.

You can enter an associated library file as a relative path. During the run session, QuickTest searches for the file in the directory for the current test, and then in the folders listed in the Folders tab of the Options dialog box. For more information, see “Setting Folder Testing Options” on page 616.

Note:  When working with tests, if your library files are stored in the file system and you want other users or Mercury products to be able to run this test on other computers, you should 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. For more information, see “Setting Folder Testing Options” on page 616.

You can add, delete and prioritize the library files associated with your test or component using the library file control buttons:

Option	Description
	<p>Associates a library file with the test or component. You can enter the absolute or relative path and filename of the library file, or use the browse button to locate the required file.</p> <p>You can associate files located in Quality Center project folders. For more information, see “Associating Library Files in Quality Center Project Folders”, below.</p>
	Removes an associated library file from the list.
	Assigns a higher priority to the selected library file.
	Assigns a lower priority to the selected library file.
	<p>Sets the current list of library files as the default list to be associated with new tests.</p> <p>Note: The Set as Default option is available for tests only. It is enabled when the setting for this test is different than the default for all tests. The default library file settings for components are specified using the component template. For more information, see “Working with Component Templates” on page 987.</p>

Associating Library Files 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.

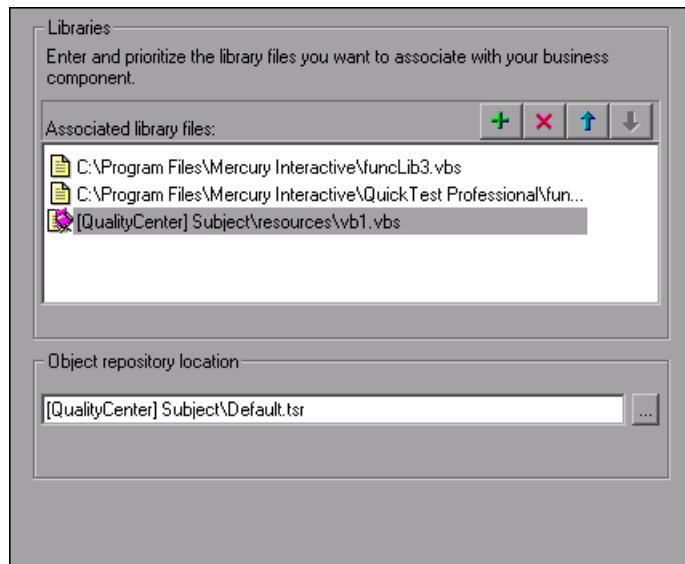
When not connected to Quality Center, you can add a file located in a Quality Center project folder by holding the SHIFT key and clicking the  button. QuickTest adds [QualityCenter], and you can enter the path. You can also type the entire Quality Center path manually. If you do, you must add a space after [QualityCenter]. For example: [QualityCenter] Subject\Tests.

Note:  When running a test, QuickTest uses associated library files from Quality Center project folders only when you are connected to the corresponding Quality Center project.

For more information about working with Quality Center projects, see Chapter 40, “Working with Quality Center.”

Defining Resource Settings for Your Component

You can use the Resources tab of the Business Component Settings dialog box to associate specific files with your component, such as VBScript library files. You can also specify the location of the shared object repository file to use for your component.



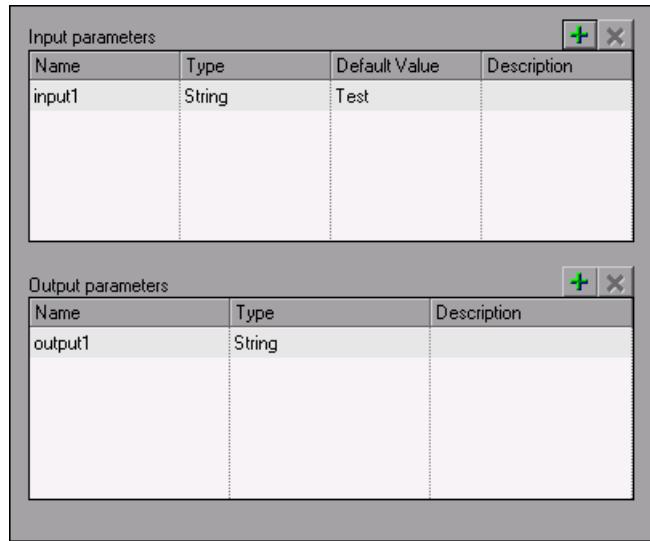
The Resources tab in the Business Component Settings dialog box includes the following option areas:

Option Area	Description
Libraries	Displays the list of library files currently associated with your component and enables you to add, delete, and prioritize the files. You can specify Quality Center files as associated library files. For more information, see “Specifying Associated Library Files” on page 677.
Object repository location	<p>Specifies the location of the shared object repository file to use with your component. Components use shared object repository files stored in Quality Center.</p> <p>You can enter the path and filename or use the browse button to locate the file. You can enter an existing shared object repository file as an absolute or relative path. If you enter a relative path, QuickTest searches for the file in the current test’s directory, and then in the folders listed in the Folders tab of the Options dialog box. For more information, see “Setting Folder Testing Options” on page 616.</p> <p>To create a new object repository file, enter the complete path of the new file.</p> <p>If you have already saved a shared object repository file as an attachment in your Quality Center project, you can enter a Quality Center path. For more information, see “Using Shared Object Repository Files with Quality Center” on page 828.</p>

Defining Parameters for Your Test or Component

In the Parameters tab, you can define input parameters that pass values into your test or component and output parameters that pass values from your test or component to external sources. You can also use the Parameters tab to modify or delete existing test or component parameters.

Test and Component parameters are similar to Action parameters. For information on Action parameters, see “Setting Action Parameters” on page 370.



The Parameters tab contains two parameter lists:

- **Input parameters**—Specifies the parameters that the test or component can receive values from the source that runs or calls it.
- **Output parameters**—Specifies the parameters that the test or component 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.

Note: For components, the input and output parameter lists can also be modified in the Quality Center Components module. For more information, refer to the *Business Process Testing User's Guide*.

You can add and remove input and output parameters for your test or component using the parameter control buttons:

Option	Description
	<p>Adds a parameter to the appropriate parameter list. Enter a name for the new parameter and select the parameter type. You can enter a description for the parameter, for example, the purpose of the parameter in the test or component.</p> <p>If you are defining an input parameter, a default value for the specified parameter type is automatically entered. You can modify enter a default value for the parameter in the Default Value column. For more information, see “Defining Default Values for Input Parameters”, below.</p> <p>You define test or component parameters in the same way you define action parameters. For information on defining parameters and parameter types, see “Setting Action Parameters” on page 370.</p>
	Removes the selected parameter from the test or component.

Defining Default Values for Input Parameters

When a test or component 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 or Component	QuickTest	Input Parameters tab of the Run dialog box. For more information, see “Running Your Entire Test or Component” on page 512.
Test	Quality Center	Test Run Properties dialog box (Test Lab module). For more information, refer to the <i>Quality Center User’s Guide</i> .
Component	Quality Center	Component Iterations dialog box (Test Plan module). For more information, refer to the <i>Business Process Testing User’s Guide</i> .

If, when a test or component 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 tab of the Test Settings or Business Components 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 or Component 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 “Using Action Parameters” on page 364.

 Once you have defined component parameters, you can use them directly in the steps in your component by selecting input component parameters in the Parameter Options or Value Configuration Options dialog box or by selecting output component parameters in the Output Options or the Storage Location Options dialog box. For more information, see “Using Test, Action, and Component Input Parameters” on page 211 and “Storing Values in Test, Action or Component Parameters” on page 252.

Alternatively, you can enter the parameter name in the Expert View using the **Parameter** utility object, in the format: `Parameter("ParameterName")`. For more information, see “Using Action, or Component Parameters in Steps in the Expert View” on page 214.

Defining Environment Settings for Your Test or Component

The Environment tab of the Test Settings or Business Component Settings dialog box 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 or component.

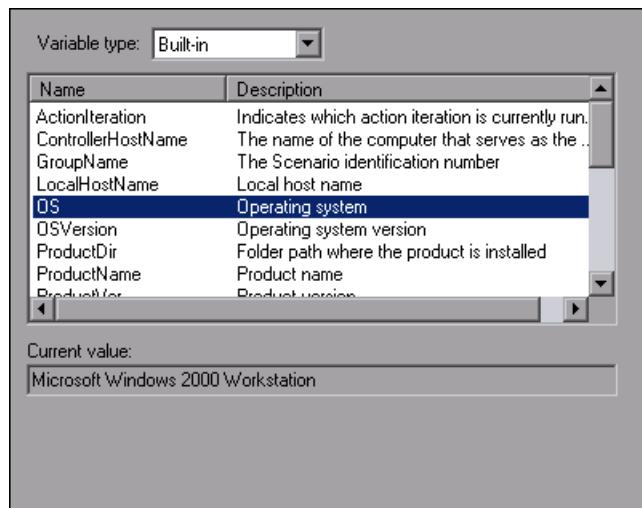
For more information about environment variables and environment parameters, see “Using Environment Variable Parameters” on page 222.

The Environment tab 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 tab lists the built-in environment variables defined by QuickTest Professional.

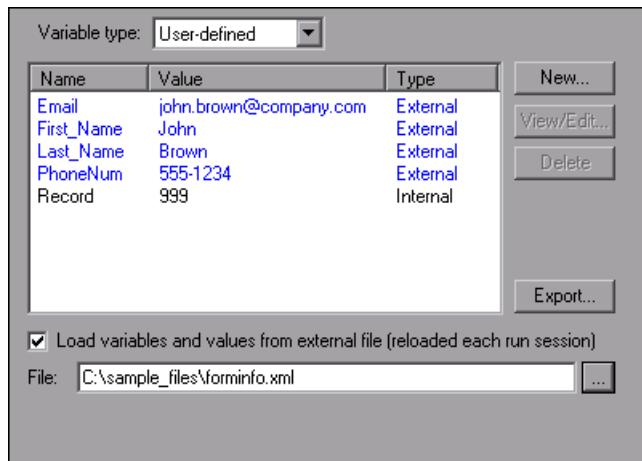


The following information is displayed for built-in environment variables:

- **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 tab lists the user-defined environment variables available for the test or component.



Note: Variables from an external environment variables file are displayed in blue. Internal environment variables are displayed in black.

The Environment tab 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 or component in which they are defined.

The Environment tab provides the following options for user-defined environment variables:

Option	Description
New	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.
View/Edit	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 689.
Delete	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 on the Test Settings or Business Component Settings dialog box.
Export	Exports your user-defined environment variables to an external .XML file for use with other tests or components. You can then use the exported environment variable file with any test or component. For more information, see “Exporting and Loading User-Defined Environment Variables” on page 691.
Load variables and values from external file (reloaded each run session)	Loads the variables saved in the .XML file that you specify for use with your test or component. For more information, see “Exporting and Loading User-Defined Environment Variables” on page 691.

Adding User-Defined Environment Variables

You can add internal user-defined environment variables in the Environment tab of the Test Settings or Business Component Settings dialog box. Internal environment variables are available only to the test or component in which they are defined.

To add internal user-defined environment variables:

- 1 In the **Variable type** box of the Environment tab, select **User-defined**.
- 2 Click the **New** button. The Add New Environment Parameter dialog box opens.



- 3 In the **Details** area, enter a definition for the variable:
 - **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 tab of the Test Settings or Business Component Settings dialog box.

Viewing and Modifying User-Defined Environment Variables

You can edit the values of internal user-defined environment variables in the Environment tab of the Test Settings or Business Component 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:

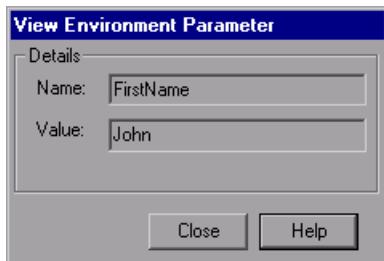
- 1 In the Environment tab of the Test Settings or Business Component Settings dialog box, double-click the internal variable, or select it and click the **View/Edit** button. The Edit Environment Parameter dialog box opens.



- 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 Windows Clipboard, select the value text, right-click, and choose **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 tab of the Test Settings or Business Component Settings dialog box.

To view an external user-defined environment variable:

- 1 In the Environment tab of the Test Settings or Business Component Settings dialog box, double-click the external variable you want to view, or select it and click the **View/Edit** 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 Windows Clipboard, select the value text, right-click and choose **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 or components. You can then use the exported environment variables with any test or component, by loading them from the file as external user-defined environment variables. Their values are loaded each time the test or component runs.

To export user-defined environment variables:

- 1 In the Environment tab of the Test Settings or Business Component 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 Choose the folder in which you want to save the file. If QuickTest is currently connected to Quality Center, you can click the **Quality Center** button to save the file in Quality Center, or you can click the **File System** button to save the file in the file system.
- 3 Type a name for the file in the **File name** box.
- 4 Click **Save** to save your file.

To load variables from an external user-defined environment variable file:

- 1 In the Environment tab of the Test Settings or Business Component Settings dialog box, select **Load variables and values from external file (reloaded each run session)**.
- 2 In the **File** box, enter the file name or click the browse button to find the external user-defined variable file. If QuickTest is currently connected to Quality Center, you can click the **Quality Center** button in the Open dialog box to find the file in Quality Center, or you can click the **File System** button to find the file in the file system.

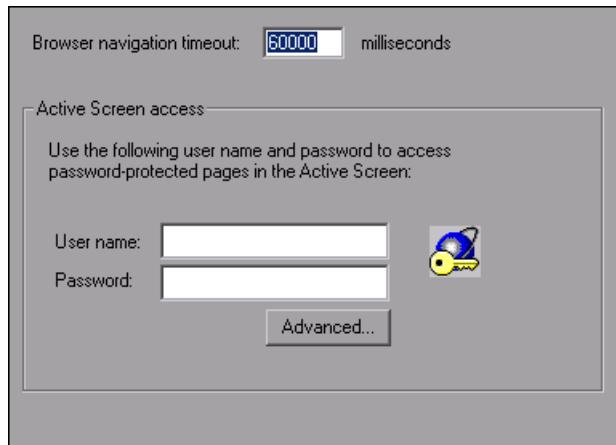
The environment variables loaded from the selected file are displayed in blue in the Environment tab of the Test Settings or Business Component Settings dialog box.

Note: You can enter a relative path for the environment variable file. QuickTest searches for the file in the folders listed in the Folders tab of the Options dialog box. For more information, see “Setting Folder Testing Options” on page 616.

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 222.

Defining Web Settings for Your Test or Component

The Web tab provides options for recording and running tests or components on Web sites. You can set how long to wait for browser navigations and you can specify the Active Screen access information to use with password-protected resources in the captured Active Screen page.



Note: The Web tab is available only if the Web Add-in is installed and loaded.

The Web tab includes the following options:

Option	Description
Browser navigation timeout	Sets the maximum time (in milliseconds) that QuickTest waits for a Web page to load before running a step in the test or component.
User name	<p>The user name for password-protected resources that use a standard authentication mechanism.</p> <p>For more information, see “Using the Standard Authentication Mechanism” on page 765.</p>
Password	<p>The password for password-protected resources that use a standard authentication mechanism.</p> <p>For more information, see “Using the Standard Authentication Mechanism” on page 765.</p>
Advanced	<p>Opens the Advanced Authentication dialog box, which enables you to manually log in to your Web site in order to enable access to password-protected resources that use an advanced authentication mechanism.</p> <p>For more information, see “Using the Advanced Authentication Mechanism” on page 767.</p>

Tip: In addition to the options in this tab, you can also configure the events you want to record for each type of Web object. For example, if you want to record events, such as a mouseover that opens a sub-menu, you may need to modify your Web event configuration to recognize such events. For more information, see Chapter 35, “Configuring Web Event Recording.”

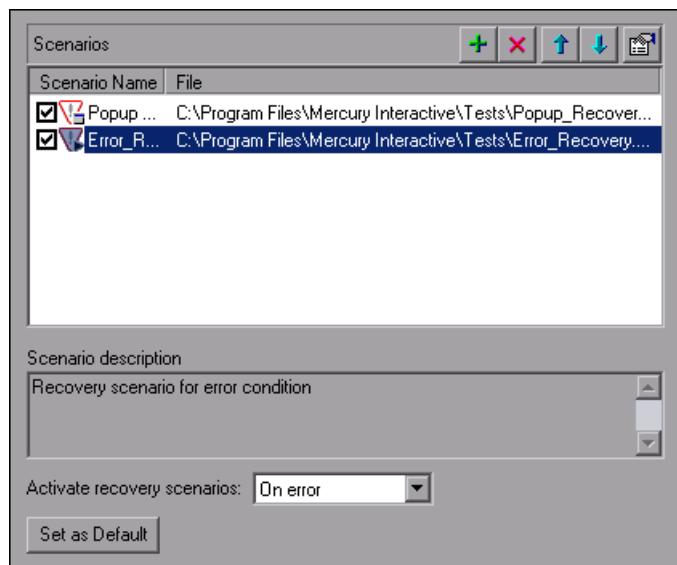
Defining Recovery Scenario Settings for Your Test or Component

The Recovery tab displays a list of all recovery scenarios associated with the current test or component. It also enables you to associate additional recovery scenarios with the test or component, remove scenarios from the test or component, 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 or component.

 If you are working with tests, you can also specify that the current list of scenarios be used as the default for all new tests.

For more information about recovery scenarios, see Chapter 19, “Defining and Using Recovery Scenarios.”



The Recovery tab includes the following option areas:

Option Area	Description
Scenarios	Displays the name and recovery file path for each recovery scenario associated with your test or component. 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 “Specifying Associated Recovery Scenarios”, below.
Scenario description	Displays the textual description of the scenario selected in the Scenarios box.
Activate recovery scenarios	<p>Instructs QuickTest to check whether to run the associated scenarios as follows:</p> <ul style="list-style-type: none"> • 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. <p>Note: Choosing On every step may result in slower performance during the run session.</p>
Set as Default 	<p>Sets the current list of recovery scenario files as the default list to be associated with new tests.</p> <p>Note: The Set as Default option is available for tests only. It is enabled when the setting for this test is different than the default for all tests. The default recovery scenario settings for components are specified in the component template. For more information, see “Working with Component Templates” on page 987.</p>

Note:  When working with tests, if your recovery files are stored in the file system and you want other users or Mercury 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. For more information, see “Setting Folder Testing Options” on page 616.

Specifying Associated Recovery Scenarios

You can select or clear the check box next to each scenario in order to enable or disable it for the current test or component.

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 or component.

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 or component does not run successfully.
	Indicates that the recovery scenario is triggered when a specified application fails during the run session.

Icon	Description
	Indicates that the recovery scenario is no longer available for the test or component. This may be 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 tab use an environment variable (**%ProductDir%**) in the file path. This enables QuickTest to locate these recovery scenarios when tests or components associated with them are run on different computers or by different Mercury 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 or component 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 or component. For more information, see “Understanding the Recovery Scenario Manager Dialog Box” on page 424.
	Removes the selected recovery scenario from the test or component.
	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 452.

26

Setting Record and Run Options



You can control how QuickTest starts recording and running tests in specific environments by setting the record and run options.

Note: This chapter describes how to define record and run settings for tests. The applications to use with components are defined differently, and are described in “Recording Components” on page 991.

This chapter describes:

- About Setting Record and Run Options
- Using the Record and Run Settings Dialog Box
- Setting Web Record and Run Options
- Setting Windows Applications Record and Run Options
- Using Environment Variables to Specify the Application Details for Your Test

About Setting Record and Run Options

You can set options that affect how you start recording and running tests for different environments. For example, you can choose to have QuickTest open specific Windows applications when you start recording or running tests in the standard Windows environment, or QuickTest can open a particular Web browser and URL you start recording and running tests on a Web site. You can set your record and run options in the Record and Run Settings dialog box, or you can set the options using environment variables.

Using the Record and Run Settings Dialog Box

Before you record or run a test on a Web or Windows application, you can use the Record and Run Settings dialog box to instruct QuickTest which applications to open when you begin to record or run your test. For Windows applications, you also specify the applications on which you want to record. Note that you can instruct QuickTest to open and record on applications from more than one environment.

Notes:

You can set the record and run settings for some add-in environments using the corresponding tab (displayed only when the add-in is installed and loaded). For other add-in environments, such as terminal emulators, you may use the Windows Applications tab. For more information on setting the record and run settings for a specific add-in, refer to the relevant QuickTest add-in documentation.

You can choose not to set record and run options at all but, in this case, you may need to open the application after you open QuickTest to ensure support for that application. For more information, refer to the Working with Supported Environments section of this guide, or refer to your add-in documentation.

The Record and Run Settings dialog box opens automatically each time you begin recording a new test (unless you open the dialog box and set your preferences manually before you begin recording). QuickTest uses the same settings for additional record sessions on the same test, and when you run the test, unless you open the Record and Run Settings dialog box manually to modify the settings.

Note: The setting of the Active Screen capture level (**Tools > Options > Active Screen** tab) can significantly affect the recording time for your test and the functionality of the Active Screen while editing your test. Confirm that the level selected answers your testing needs. For more information, see “Setting Active Screen Options” on page 618.

The Record and Run Settings dialog box can contain the following tabbed pages:

Tab Heading	Subject
Web	<p>Options for testing Web sites.</p> <p>Note: The Web tab is available only when Web support is installed and loaded.</p> <p>For more information, see “Setting Web Record and Run Options” on page 703.</p>
Windows Applications	<p>Options for testing standard Windows applications.</p> <p>Note: This tab is always available and applies to all Standard Windows, Visual Basic, and ActiveX. (Appropriate add-ins must also be loaded.)</p> <p>For more information, see “Setting Windows Applications Record and Run Options” on page 705.</p>

In addition to these tabs, the Record and Run Settings dialog box may contain other tabs corresponding to any external add-ins that are loaded. For more information on external add-ins, refer to the relevant QuickTest add-in documentation.

Note: If you define environment variables to specify the record and run application details, those values override the values in the Record and Run dialog box. For more information, see “Using Environment Variables to Specify the Application Details for Your Test” on page 709.

To set record and run options:



- 1** Click the **Record** button or choose **Test > Record**. If you are recording for the first time in a test and have not yet set your recording preferences (by opening the dialog box manually), the Record and Run Settings dialog box opens. It is divided by environment into several tabbed pages.
- 2** To choose an environment, click a tab.
- 3** Set the required options, as described in the following sections.
- 4** To apply your changes and keep the Record and Run Settings dialog box open, click **Apply**.
- 5** When you are finished, click **OK** to save your changes and start recording.

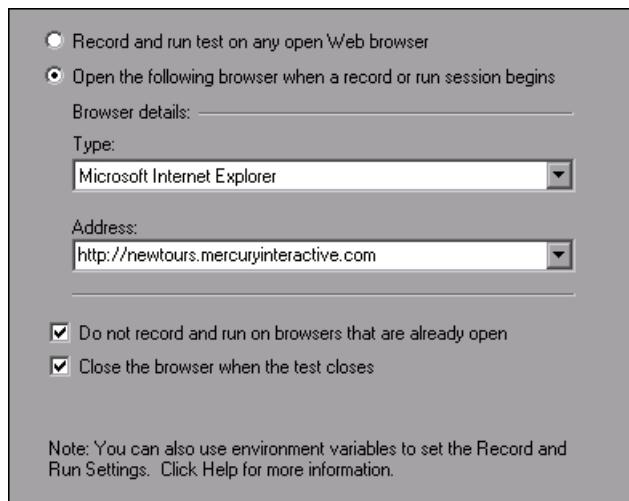
Note: Once you have set the record and run settings for a test, the Record and Run settings dialog box will not open the next time you record operations in that test. However, you can choose **Test > Record and Run Settings** to open the Record and Run Settings dialog box. Use this option to set or modify 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 or browser than the one you previously set in the Record and Run Settings dialog box.

If you change the record and run settings for additional recording sessions, confirm that you return the settings to match the needs of the first step in your test before you run it.

Setting Web Record and Run Options

The Web tab defines your browser preferences for recording and running your test.



Note: The Web tab is available only when the corresponding Web add-in is installed and loaded.

The Web tab includes the following options:

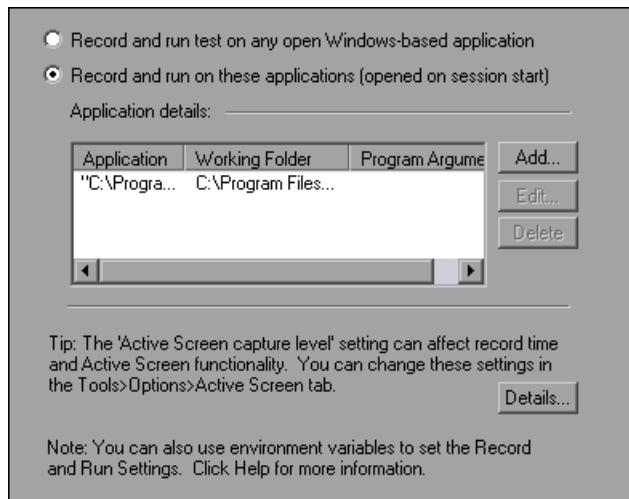
Option	Description
Record and run test on any open Web browser	<p>Instructs QuickTest to use any open Web browser to record and run the test.</p> <p>Note: You must open your Web browser after you open QuickTest.</p>
Open the following browser when a record or run session begins	<p>Instructs QuickTest to open a new browser session to record and run the test using the specified URL address.</p> <p>Note: This option can be used with supported versions of Microsoft Internet Explorer and Netscape Navigator. For other browser types, use the Record and run tests on any open Web browser option. For information on supported browser versions, see “Working with Web Browsers” on page 742 and refer to the <i>QuickTest Professional Readme</i> file.</p>
Browser details	<p>Instructs QuickTest regarding the use of the browser during test recording and test runs.</p> <ul style="list-style-type: none"> • Type—Instructs QuickTest to open the specified browser type (i.e. Internet Explorer, Netscape, etc.). Note that the Type list displays only those browsers that are currently installed on your computer. • Address—Instructs QuickTest to open the browser to the specified URL. <p>Note: If you define a value for the browser type or URL environment variables, that value overrides the value in the Type or Address box of the Web tab during a run session. For more information, see “Using Environment Variables to Specify the Application Details for Your Test” on page 709.</p>
Do not record and run on browsers that are already open	<p>Instructs QuickTest not to record or run tests on any browsers that are already open prior to the start of the record or run session.</p>
Close the browser when the test closes	<p>Instructs QuickTest to close the browser window specified in the Address box when the test closes.</p>

Note to AOL users and users of applications with embedded Web browser controls:

To record and run tests on AOL or an application with embedded Web browser controls, select **Record and run tests on any open Web browser** in the Record and Run Settings dialog box, make sure AOL or the application is opened after QuickTest, and start recording. For more information on browser settings, see “Recording a Test or Component” on page 88. For additional information on working with the AOL browser, refer to the *QuickTest Professional Readme* file.

Setting Windows Applications Record and Run Options

The Windows Application tab defines your preferences for recording and running tests on Windows applications including Standard Windows, Visual Basic, and ActiveX applications.



The record and run options in this tab have a slightly different significance than the corresponding options in the other tabs.

Selecting **Record and run test on any open Windows-based application** records all operations performed on any Windows-based application that is open while recording your test (including on e-mail applications, file management applications, etc.).

Selecting **Record and run on these applications (opened on session start)** restricts recording operations to the Windows applications specified in the list. Additionally, QuickTest opens these applications for you at the beginning of a record or run session.

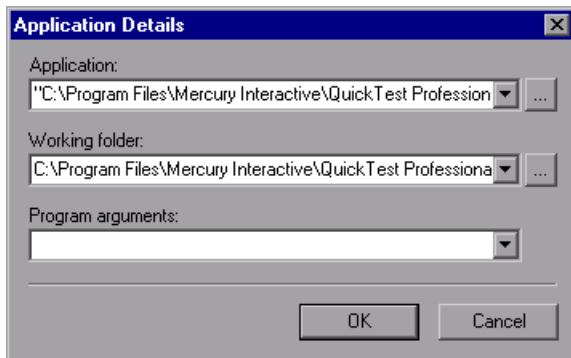
The Windows Applications tab includes the following options:

Option	Description
Record and run test on any open Windows-based application	<p>Instructs QuickTest to use any open Windows-based application to record and run the test.</p> <p>Note: Make sure that all the applications on which you want to record are currently closed. QuickTest can record on the applications that you open manually only <i>after</i> you select this option and click OK. Instances of these applications that are already open when the Record and Run Settings dialog box opens may be ignored or may not be recorded correctly.</p>
Record and run on these applications (opened on session start)	<p>Instructs QuickTest to restrict its recording on Windows applications to the specified applications and to open these applications when record or run sessions begin.</p> <p>If you do not want to record on any standard Windows application, select this option and leave the Application details area blank (this is the default setting).</p> <p>When working with standard Windows applications only, you can manually add steps to your test and then run them, even if you select this option and leave the Application details area blank (or if the list does not contain the application for which you want to add a step).</p> <p>Note: Make sure that all the applications listed in the Application details area are currently closed. QuickTest can record only on the instances of the specified applications that are opened <i>after</i> you select this option and click OK. Instances of these applications that are already open when the Record and Run Settings dialog box opens may be ignored or may not be recorded correctly.</p>

Option	Description
Application details	Lists the details of the applications on which to record and run the test.
Add	<p>Opens the Application Details dialog box to enable you to add an application to the application list. You can add up to ten applications.</p> <p>For more information, see “Adding or Editing Application Details”, below.</p>
Edit	<p>Opens the Application Details dialog box to enable you to edit the application details for the selected application. For more information, see “Adding or Editing Application Details”, below.</p>
Delete	Deletes the selected application from the Applications details list.

Adding or Editing Application Details

When you click **Add** or **Edit** in the Windows Applications tab of the Record and Run dialog box, the Application Details dialog box opens.



You can add an application to the application list displayed in the Windows Applications tab, and you can edit an existing application in the list. You can specify up to ten applications in the list.

You can specify the following details for the application in the Application Details dialog box:

Option	Description
Application	<p>Instructs QuickTest to record on and open the specified executable file.</p> <p>You can enter the executable file as a relative path. During the run session, QuickTest searches for the file in the directory for the current test, and then in the folders listed in the Folders tab of the Options dialog box. For more information, see “Setting Folder Testing Options” on page 616.</p> <p>Note: The Application box should contain only the file name and path for the application. If you want to add command line arguments, use the Program arguments box.</p> <p>Tip: You can specify a document or other file associated with the application, for example, c:\tmp\A.txt. QuickTest opens the specified file in the associated application (Notepad in this example). If you use this option, QuickTest ignores the contents of the Program arguments box.</p>
Working folder	Instructs QuickTest to open the application from the specified folder.
Program arguments	Optional. Instructs QuickTest to open the application using the specified command line arguments.

Using Environment Variables to Specify the Application Details for Your Test

You can use special, predefined environment variables to specify the applications you want to use for your test. This can be useful if you want to test how your application works in different environments. For example, you may want to test that your Web application works properly on identical or similar Web sites with different Web addresses.

When you define an environment variable for one (or more) of the application detail options and you select the option to open an application when record and run sessions begin for a particular environment (the lower radio button in each tab of the Record and Run Settings dialog box), the environment variable value overrides the value specified in the Record and Run Settings dialog box.

Note: If you select the option to Record and Run on any application from a particular environment (the upper radio button in each tab of the Record and Run Settings dialog box), QuickTest ignores the application details environment variables.

You can define the environment variables as internal user-defined variables, or you can add them to an external environment variable file and set your test to load environment variables from that file.

You can set your Record and Run settings manually while recording your test and then define the environment variables or load the environment variable file only when you are ready to run the test (as described in the procedure below).

Alternatively, you can define environment variables before you record your test. In this case, QuickTest uses these values to determine which applications to open when you begin recording—assuming that the option to open an application when starting record and run sessions for the particular environment is selected. (This option is the lower radio button in each tab of the Record and Run Settings dialog box.)

To use application details environment variables for your test:

- 1** Set your Record and Run Setting preferences normally to record your test.
-

Note: If you already have environment variables set for one or more application details, and you select the option to open an application when the record session begins, the environment variable values override the record settings you enter in the dialog box.

- 2** Record and edit your test normally.
- 3** If you did not define environment variables prior to recording your test, define an environment variable for each application detail you want to set using the appropriate variable name.

For a list of available application details environment variables, see “Defining Application Details Environment Variables,” below.

For more information on how to define a user-defined environment variable and how to create environment variable files, see “Using Environment Variable Parameters” on page 222.
- 4** Before running the test, confirm that the lower radio button is selected in the tab(s) corresponding to the environment(s) for which you want to use environment variables.
- 5** Run the test. QuickTest uses the environment values to determine which application(s) to open at the beginning of the run session.

Defining Application Details Environment Variables

In order to use environment variables to specify the applications you want to use for your test run, you must use the appropriate variable names as specified below.

Use the variable name listed in the table below to define the Web browser and URL to open:

Option	Variable Name	Description
Type	BROWSER_ENV	The browser type to open. Possible values: IE, NS, NS6
Address	URL_ENV	The Web address to display in the browser.

Use the variable name listed in the table below to define the details for Windows applications on which you want to record and run test:

Option	Variable Names	Description
Application	1_APP_ENV, 2_APP_ENV, ... 10_APP_ENV	The executable files on which QuickTest records operations and opens when record and run sessions begin. You can specify up to ten application names.
Working Folder	1_DIR_ENV 2_DIR_ENV ... 10_DIR_ENV	The folder to which the corresponding executable file refers (for up to ten corresponding folder paths).

27

Customizing the Expert View

You can customize the way your test or component is displayed when you work in the Expert View.

This chapter describes:

- About Customizing the Expert View
- Customizing Expert View Behavior
- Customizing Script Element Appearance
- Personalizing Editing Commands

About Customizing the Expert View

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

The Editor Options dialog box enables you to change the way scripts are displayed in the Expert View. You can also change the font style and size of text in your scripts, and change the color of different script 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 let you move the cursor, delete characters, 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.

Note: You can modify the way your script is printed using options in the Print dialog box. For more information, see “Printing a Test or Component” on page 99.

For more information on using the Expert View, see Chapter 36, “Working with the Expert View.”

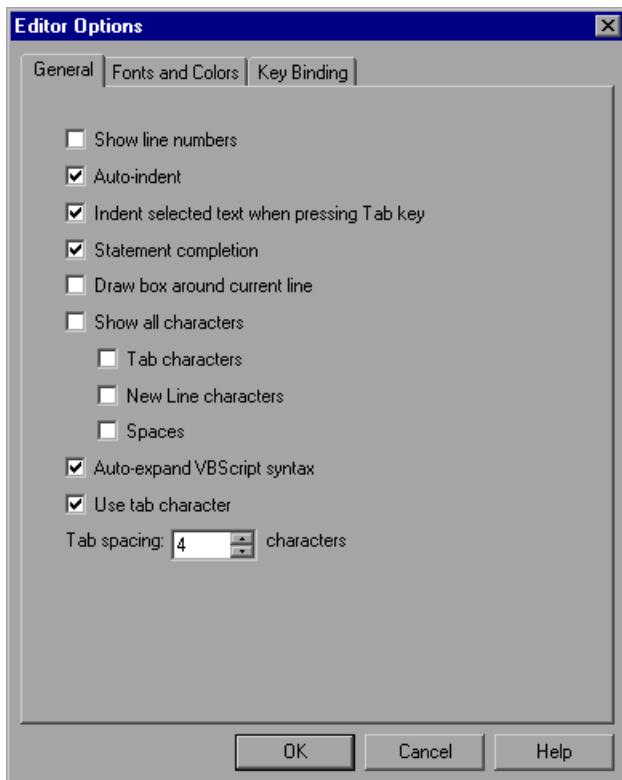
Customizing Expert View Behavior

You can customize how scripts are displayed in the Expert View. 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 36, “Working with the Expert View.”

To customize Expert View behavior:

- 1** Choose **Tools > Editor Options**. The Editor Options dialog box opens.

- 2 Click the General tab.



- 3 Choose from the following options:

Options	Description
Show line numbers	Displays a line number to the left of each line in the script.
Auto-indent	Causes lines following an indented line to automatically begin at the same point as the previous line. You can click the HOME key on your keyboard to move the cursor back to the left margin.

Options	Description
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	<p>When this option is selected, if you type:</p> <ul style="list-style-type: none"> • a dot after a test object—QuickTest displays a list of available test objects and methods that you can add after the object you typed. • an open parenthesis (after an object—QuickTest displays a list of all test objects of this type in the object repository. If there is only one object of this type in the object repository, QuickTest automatically enters its name in quotes after the open parenthesis. • a method—QuickTest displays the syntax for the method, including its specific mandatory and optional arguments. • the Object property—if the object data is currently available in the Active screen or the open application, QuickTest displays native methods and properties of any run-time object in your application. <p>For more information on using the statement completion (IntelliSense) feature, see “Generating Statements in the Expert View” on page 858.</p>
Draw box around current line	Displays a box around the line of the test or component in which the cursor is currently located.
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.

Options	Description
Auto-expand VBScript syntax	<p>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.</p> <p>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:</p> <pre>If Then End If</pre>
Use tab character	<p>Inserts a TAB character when the TAB key on the keyboard is used. When this option is not enabled, the appropriate number of space characters is inserted, when you press the TAB key.</p>

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

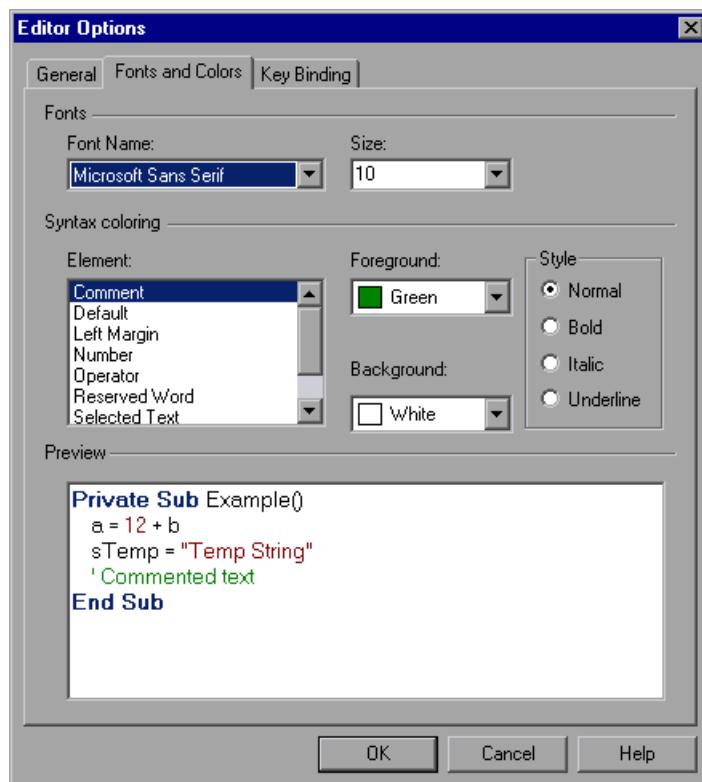
Customizing Script Element Appearance

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

To set font and color preferences for script elements:

- 1 Choose **Tools > Editor 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 script 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 component may not be correctly displayed in the Expert View. However, the test or component 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 a script 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.

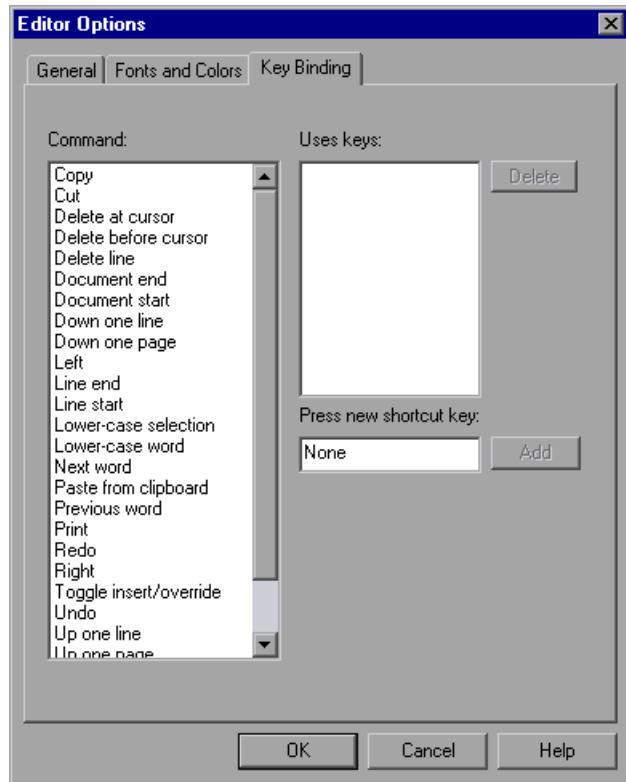
Personalizing Editing Commands

You can personalize the default keyboard shortcuts you use for editing scripts. 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 “Executing Commands Using Shortcut Keys” on page 22.

To personalize editing commands:

- 1** Choose **Tools > Editor 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 key(s) 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 is already defined for another command, a message to this effect is displayed below the shortcut key box.

- 6 Repeat steps 3 - 5 for any additional commands as required.
- 7 If you want to delete a key sequence from the list, select the command in the **Command** list, then highlight the key(s) in the **Uses keys** list, and click **Delete**.
- 8 Click **OK** to apply the changes and close the dialog box.

28

Setting Testing Options During the Run Session

You can control how QuickTest records and runs tests or components by setting and retrieving testing options during a run session.

This chapter describes:

- About Setting Testing Options During the Run Session
- Setting Testing Options
- Retrieving Testing Options
- Controlling the Test Run
- Adding and Removing Run-Time Settings

About Setting Testing Options During the Run Session

QuickTest testing options affect how you record and run tests and components. For example, you can set the maximum time that QuickTest allows for finding an object in a page.

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 36, “Working with the Expert View.”

By retrieving and setting testing options using the **Setting** object, you can control how QuickTest runs a test or component.

You can also set many testing options using the Options dialog box (global testing options), the Test Settings dialog box (test-specific settings), and the Business Component Settings dialog box (component-specific settings). For more information, see Chapter 24, “Setting Global Testing Options” and Chapter 25, “Setting Options for Individual Tests or Components.”

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, refer to the **Utility** section of the *QuickTest Professional Object Model Reference*.

Note: You can also control QuickTest options as well as most other QuickTest operations from an external application using automation programs. For more information, see “Automating QuickTest Operations” on page 919, or refer to the *QuickTest Automation Object Model Reference* (**Help > QuickTest Automation Object Model Reference**).

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 are per-test settings.

Using the **Setting** object with a global testing option changes a testing option globally, and this change is reflected in the Options dialog box.

For example, if you execute the following statement:

```
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** check box in the Advanced Web Options dialog box (Choose **Tools > Options > Web** tab, and click **Advanced**).

Using the **Setting** object to set per-test options is also reflected in the Test Settings dialog box. 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 726.

For example, if you execute 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 tab 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 724.

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")
```

Note: When using a **Setting.Add** statement, an error occurs if you try to add an existing key value. To avoid this error you should use a **Setting.Exists** statement first. For more details about all the **Setting** methods, refer to the *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")
```


Part VI

Working with Supported Environments

Working with QuickTest Add-Ins

QuickTest Professional is provided with several core add-ins, including Web, ActiveX, and Visual Basic. You can purchase additional add-ins separately.

When you work with these add-ins, you can use special checkpoints, methods, and properties to create the best possible test or component for your application.

This chapter describes:

- About Working with QuickTest Add-ins
- Loading QuickTest Add-ins
- Tips for Working with QuickTest Add-ins

About Working with QuickTest Add-ins

You can choose to install the core add-ins when you install QuickTest Professional, or you can run the installation again at another time to install the QuickTest add-ins. Add-ins that are installed separately from the QuickTest Professional core installation are referred to as *external* add-ins. You can install external QuickTest add-ins at any time after you install QuickTest Professional.

When QuickTest opens, you can choose which of the installed add-ins you want to load using the Add-In Manager dialog box.

If you choose to install and load an add-in, QuickTest recognizes the objects in your application when you record on the corresponding environment and enables you to work with the appropriate properties, methods, checkpoints, and other specialized options.

External add-ins require a separate seat or concurrent license code. For seat licenses, you install the QuickTest Professional Add-in License using the Add-in Manager dialog box. For concurrent licenses, you install the QuickTest Professional Add-in License on the Mercury Functional Testing Concurrent License Server computer.

Note: The QuickTest Professional Add-in License is valid for all QuickTest Professional external add-ins. This means that after you have installed it for one external add-in, you do not need to install it again if you install additional external add-ins on the same computer.

For more information on installing add-ins, refer to the *QuickTest Professional Installation Guide*.

Loading QuickTest Add-ins

If you have installed QuickTest add-ins, you can specify which add-ins to load at the beginning of each QuickTest session.

When you start QuickTest, the Add-in Manager dialog box opens. It displays a list of all installed add-ins for QuickTest. You can select which add-ins to load for the current session of QuickTest.



If the add-in license has not yet been installed, any installed external add-ins are displayed as (**not licensed**) in the Add-in Manager dialog box. If the words (**not licensed**) are not displayed, then the add-in license has already been installed, and you do not need to install it again.

If you have an old version of a QuickTest add-in installed, the Add-in Manager displays the word **outdated** next to the add-in name. You cannot load an outdated add-in.

Tip: You can use most QuickTest Professional external 6.5.x add-ins with either a new installation or upgraded installation of QuickTest Professional 8.0 and the relevant Add-in Upgrade patch (provided on the QuickTest Professional CD-ROM). The installation procedures to follow are different for Web-based external 6.5.x add-ins (SAP solutions, .NET, PeopleSoft) and external 6.5.x add-ins that are not Web-based (Terminal Emulator, Java, Oracle), and also depend on whether you are installing QuickTest Professional 8.0 from scratch or upgrading from QuickTest Professional 6.5. For information and instructions, refer to the *QuickTest Professional 8.0 Installation Guide*.

Note: Note that the QuickTest Professional Multimedia Add-in is not supported in QuickTest Professional 8.0. Therefore, if you are upgrading from an earlier version of QuickTest Professional, you will see this outdated add-in in the Add-in Manager dialog box.

The Add-in Manager includes the following options:

Option	Description
Add-ins list	Lists the installed add-ins that are available to load with QuickTest. When you click OK, QuickTest loads the selected add-ins. By default, the ActiveX and Web add-ins are selected the first time QuickTest is started. Each time you start QuickTest, the currently selected add-ins are loaded.
Description	Describes the add-in.

Option	Description
Show on startup	<p>Displays the Add-in Manager dialog box each time you open QuickTest.</p> <p>When this check box is cleared, QuickTest opens without displaying the Add-in Manager. To display it again, choose Tools > Options > General and select Display Add-in Manager on startup.</p> <p>For information on working with the Options dialog box, see Chapter 24, “Setting Global Testing Options.”</p>
Add-in License (displayed only when a QuickTest seat license is installed on your computer.)	<p>Opens the QuickTest Professional Software License Installation - Welcome window, which enables you to install the add-in license for external add-ins.</p> <p>The procedure for installing an add-in license is the same as the procedure for installing a QuickTest Professional license. Refer to the <i>QuickTest Professional Installation Guide</i> for additional information.</p>

Notes:

If you install an old version of a QuickTest add-in over your current installation of QuickTest, you must follow specific instructions to ensure that QuickTest functions properly. For more information, refer to the *QuickTest Professional Installation Guide*.

To maximize performance and object identification reliability, load only the add-ins you need.

Matching Loaded Add-ins with Associated Add-ins

When you open a test or component, QuickTest compares the add-ins that are currently loaded with the add-ins associated with your test or component. If they do not match, QuickTest issues a warning message.

If there are add-ins associated with your test or component that are not currently loaded, depending on your requirements, you can either:

- close and reopen QuickTest, and select the required add-ins in the Add-in Manager dialog box, or
- remove the add-ins from the list of associated add-ins for your test or component

If add-ins are loaded but not associated with your test or component, you can either:

- close and reopen QuickTest, and clear the checkboxes for the add-ins in the Add-in Manager dialog box, if they are not required, or
- add the add-ins to the list of associated add-ins for your test or component

To change the list of add-ins associated with your test or component, choose **Test > Settings or Component > Settings** and use the **Modify** option in the Properties tab. For more information on associating add-ins, see “[Associating Add-ins with Your Test or Component](#)” on page 662.

Tips for Working with QuickTest Add-ins

QuickTest add-ins help you to create and run tests and components on applications created in a variety of environments. Once you load an add-in, you can record and run tests and components on applications in that environment similar to the way you do with any other application.

To take full advantage of QuickTest add-in capabilities, keep the following in mind when designing tests or components for using QuickTest add-ins:

- You must install and load an add-in to enable QuickTest to recognize objects from the corresponding environment. To load an add-in, select the add-in from the Add-in Manager dialog box that opens when you start QuickTest.
- If the Add-in Manager does not open when you start QuickTest, choose **Tools > Options** and click the **General** tab. Select the **Display Add-in Manager on startup** check box and click **OK**. Restart QuickTest.



- To maximize performance and object identification reliability, load only the add-ins you need. For example, if you want to test a process that spans a Web application and a .NET application, load only the Web and .NET Add-Ins. Do not load all add-ins unless you need to work with all add-ins.
- You can view the list of add-ins that are currently installed or loaded by choosing **Help > About QuickTest Professional**. The dialog box displays a list of all add-ins installed on your computer. A checked box indicates that the add-in is currently loaded.
- When you record on objects in your application, QuickTest displays an environment-specific icon in the step in the Keyword View and generates a statement with the appropriate test object and method.
- QuickTest offers environment-specific checkpoints and output values that you can use to enhance your test or component. For more information, see Chapter 6, “Understanding Checkpoints.”
- You can also add additional steps to your test or component using the Step Generator or manually in the Expert View. For more information on the objects, methods, and properties available for your application’s environment, refer to the *QuickTest Professional Object Model Reference*.
- For detailed information on testing an environment supported by a core QuickTest add-in, select one of the following chapters:
 - Chapter 30, “Testing Web Objects”
 - Chapter 31, “Testing Visual Basic Applications”
 - Chapter 32, “Testing ActiveX Controls”

For information about QuickTest external add-ins, purchased separately, refer to the documentation included with the relevant QuickTest add-in.

30

Testing Web Objects

QuickTest supports testing Web objects. By adding Web object checkpoints to your tests or components, you can compare Web objects in different versions of your Web site. For information on supported browser versions, refer to the *Readme* file.

This chapter describes:

- ▶ About Testing Web Objects
- ▶ Working with Web Browsers
- ▶ Checking Web Objects
- ▶ Checking Web Pages
- ▶ Checking Web Content Accessibility
- ▶ Accessing Password-Protected Resources in the Active Screen
- ▶ Activating Methods Associated with a Web Object
- ▶ Using Scripting Methods with Web Objects

About Testing Web Objects

You can use QuickTest's Web Add-in to test your Web pages and applications. You can test Web objects such as hyperlinks, images, image maps, and Viewlink objects.

You can create Web object checkpoints to compare the expected values of object properties captured during the recording of the test or component to the object's current values during a run session. You can perform checks on Web page properties, text, and tables.

You can also perform checks on objects within your application or Web site, such as images or form elements. In addition, you can add accessibility checkpoints to help you quickly identify areas of your Web site that may not conform to the W3C Web Content Accessibility Guidelines. You can also output property or text values from the objects in your Web site.

The following checkpoints and output values are supported when testing Web objects:

Type	Checkpoint	Output	For more information, see:
Standard	Yes	Yes	"Checking Object Property Values," on page 123, and "Outputting Property Values," on page 255.
Page	Yes	Yes	"Checking Web Pages," on page 747, and "Outputting Property Values," on page 255
Accessibility	Yes	No	"Checking Web Content Accessibility" on page 760
Text	Yes	Yes	"Creating a Text Checkpoint," on page 153, and "Outputting Text Values," on page 268
Table	Yes	Yes	"Checking Tables and Databases," on page 133, and "Outputting Property Values," on page 255
Bitmap	Yes	No	"Checking Bitmaps" on page 169
XML (Application)	Yes	No	"Checking XML" on page 181

Before you begin recording on Web sites and applications, you should ensure that you have installed and loaded the Web add-in. You can check whether the Web add-in is installed by choosing **Help > About QuickTest Professional**. Loaded add-ins are indicated by a check mark in the add-ins list.

You should also set your preferences in the Web tab of the Record and Run dialog box (for tests only), the Web tab of the Test Settings dialog box or Business Components dialog box, and the Web tab of the Options dialog box. For more information, see Chapter 26, “Setting Record and Run Options,” Chapter 25, “Setting Options for Individual Tests or Components,” and Chapter 24, “Setting Global Testing Options” respectively.

If QuickTest does not record Web events in a way that matches your needs, you can also configure the events you want to record for each type of Web object. For example, if you want to record events, such as a mouseover that opens a sub-menu, you may need to modify the Web event configuration to recognize such events. For more information, see Chapter 35, “Configuring Web Event Recording.”

Note: If you are recording on a drop-down list in a Web page or application, you must click on the drop-down 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). This is because QuickTest only records a step if the value in the list changes.

Working with Web Browsers

You use a Web browser to record tests or components that check Web objects. You select your browser in the Web tab of the Record and Run Settings dialog box. For more information, see “Setting Web Record and Run Options” on page 703.

Note: By default, the name assigned to the Browser test object in the object repository is always the name assigned to the first page recorded for the browser object. The same test object is used each time you record on a browser with the same ordinal ID in future recording sessions. Therefore, the name used for the browser in the steps you record may not reflect the actual browser name.

QuickTest supports recording and running tests or components on the following Web browsers:

- Netscape
 - Microsoft Internet Explorer
 - AOL (America Online)
 - Applications with embedded Web browser control
-

Note: QuickTest tests or components are generally cross-browser—you can record a test or component on one browser and run it on any other browser. For information on supported browser versions, refer to the *Readme* file.

Working with Internet Explorer

QuickTest Professional Web support behaves as a browser extension in Internet Explorer. Therefore, you cannot use the Web Add-in on Internet Explorer 6.x without enabling the **Enable third-party browser extensions** option. To set the option, choose **Tools > Internet Options > Advanced** and select the **Enable third-party browser extensions** option in Internet Explorer 6.x.

Working with Netscape

Keep the following in mind when using Netscape as your Web browser:

- The **Object** property accesses DOM objects. These are not supported by Netscape. For more information on the **Object** property, see “Accessing Run-Time Object Properties and Methods” on page 899.
- Dialog boxes which opened while recording in Netscape are not displayed in the Active Screen.
- You can record only the following objects on dialog boxes opened while recording in Netscape:
 - button
 - check box
 - edit—You cannot apply a checkpoint to or use the Object Spy on an edit object recorded in a dialog box.

Working with AOL and Applications with Embedded Web Browser Controls

To record and run tests or components on AOL or an application with embedded Web browser controls:

- make sure the ActiveX Add-in is loaded
-  select **Record and run tests on any open Web browser** in the Record and Run Settings dialog box
- make sure AOL or the application is opened after QuickTest
- start recording

For more information on browser settings, see “Recording a Test or Component” on page 88.

For additional information on working with the AOL browser, refer to the *QuickTest Professional Readme* file.

Checking Web Objects

You create a checkpoint on a Web object as you create a checkpoint on any standard object. When you create a checkpoint on a Web object, QuickTest captures the object's properties and their values as it does for any standard object. The properties you can check for a Web object depend on the properties of the Web object.

You can create a checkpoint either while recording or editing your test or component.

For example, you can create a checkpoint on your Web site to check the value of an edit field (No. of Passengers, for example) on a specific Web page (Find Flights page, for example).

To create a checkpoint on a Web object:

- 1 In the Keyword View, highlight the step for the page that has the object you want to check. The Active Screen displays the HTML source corresponding to the highlighted step.

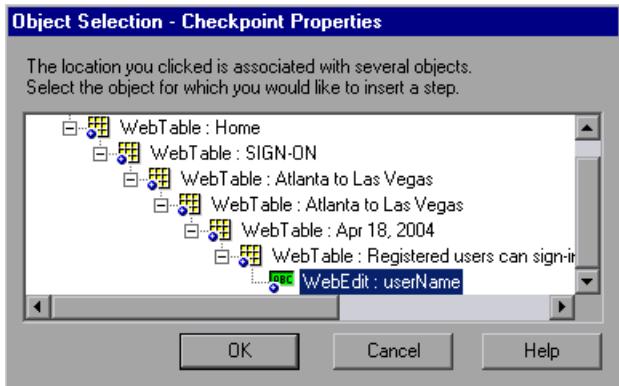
The screenshot shows the QuickTest interface with two main panes: Keyword View and Active Screen.

Keyword View: A table showing steps for a keyword named "A1". The table has columns for Item, Operation, Value, and Documentation.

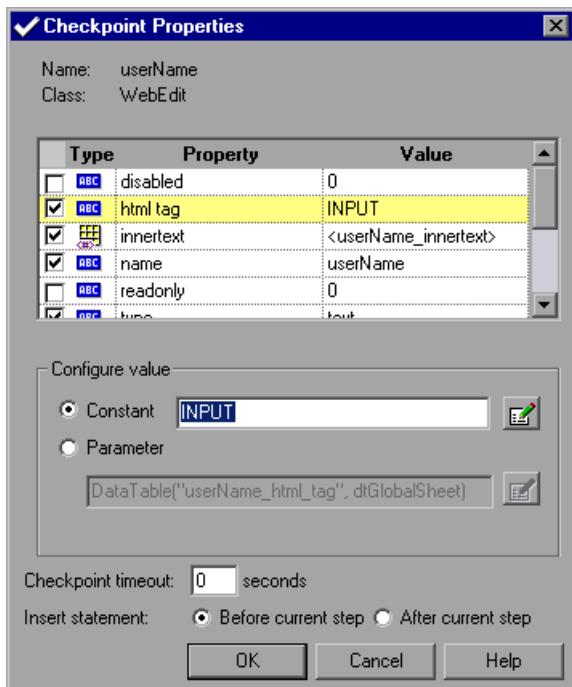
Item	Operation	Value	Documentation
Welcome: Mercury Tours			
Welcome: Mercury Tours			
userN ^{ame}	Set	"mercury"	Enter "mercury" in the "u
password	SetSecure	"4082986e39ea4...	Enter the encrypted string
Sign-In	Click	2,2	Click the "Sign-In" image

Active Screen: Displays a web page titled "Find A Flight". The page contains a form with fields for "User Name" (containing "mercury") and "Password". The "User Name" field is highlighted with a pink rectangle.

- 2 In the Active Screen, right-click the object you want to check and choose **Insert Standard Checkpoint**. The Object Selection - Checkpoint Properties dialog box opens.
- 3 Confirm that the object you want to check is highlighted.



Click **OK**. The Checkpoint Properties dialog box opens.



This dialog box displays the properties of the object:

- The **Name** is the name that QuickTest assigns to the test object.
- The **Class** is the type of object. WebEdit indicates that the object is an edit field.
- The **ABC** icon indicates that the value of the property is a constant.

The table below describes the selected, default checks.

Property	Value	Explanation
html tag	INPUT	INPUT is the html tag as defined in the HTML source code.
innertext		In this case, the value of innertext is empty. The checkpoint checks that the value is empty.
name	numPassengers	numPassengers is the name of the edit field.
type	text	text is the type of object as defined in the HTML source code.
value	1	1 is the value 1 in the edit field.

For each object class, QuickTest recommends default property checks. Click **OK** to accept the default properties for the object (html tag, innertext, name, type, and value). QuickTest adds the object checkpoint to your test or component. It is displayed in the Keyword View as a new step under the Find Flights page.

For more information on creating checkpoints, see Chapter 6, “Understanding Checkpoints.”

Checking Web Pages

You can check statistical information about your Web pages by adding page checkpoints to your test or component. These checkpoints check the links and the sources of the images on a Web page. You can also instruct page checkpoints to include a check for broken links.

Automatic Page Checkpoints

You can instruct QuickTest to create automatic page checkpoints for every page in all tests or components by selecting the **Create a checkpoint for each Web page while recording** check box in the Advanced Web Options dialog box (click the **Advanced** button in the Web tab of the Options dialog box). By default, the automatic page checkpoint includes the checks that you select from among the available options in the Advanced Web Options dialog box.

You can also instruct QuickTest not to perform automatic page checkpoints when you run your test or component by selecting the **Ignore automatic checkpoints while running tests** check box in the Advanced Web Options dialog box of the Web tab of the Options dialog box.

For more information, see Chapter 24, “Setting Global Testing Options.”

Creating Individual Page Checkpoints

You can manually add a page checkpoint to your test or component to check the links and the image sources on a selected Web page either while recording or editing your test or component.

To add a page checkpoint while recording:

- 1 Navigate to a page where you want to add a checkpoint.
- 2 Choose **Insert > Checkpoint > Standard Checkpoint** or click the **Insert Checkpoint** button and click in the page. The Object Selection - Checkpoint Properties dialog box opens.

- 3 Select the **Page** item and click **OK**. The Page Checkpoint Properties dialog box opens.

- 4 Modify the settings for the checkpoint in the Page Checkpoint Properties dialog box, as described in “Understanding the Page Checkpoint Properties Dialog Box” on page 750.
- 5 Click **OK** to close the dialog box. A checkpoint step is added in the Keyword View.

To add a page checkpoint while editing your test or component:

- 1** Make sure the **Active Screen** button is selected.
- 2** Click a step in your test or component where you want to add a checkpoint. The Active Screen displays the HTML source corresponding to the highlighted step.
- 3** Right-click anywhere on the Active Screen and choose **Insert Standard Checkpoint**. The Object Selection - Checkpoint Properties dialog box opens. Select the **Page** item you want to check from the displayed object tree.
- 4** Click **OK**. The Page Checkpoint Properties dialog box opens.



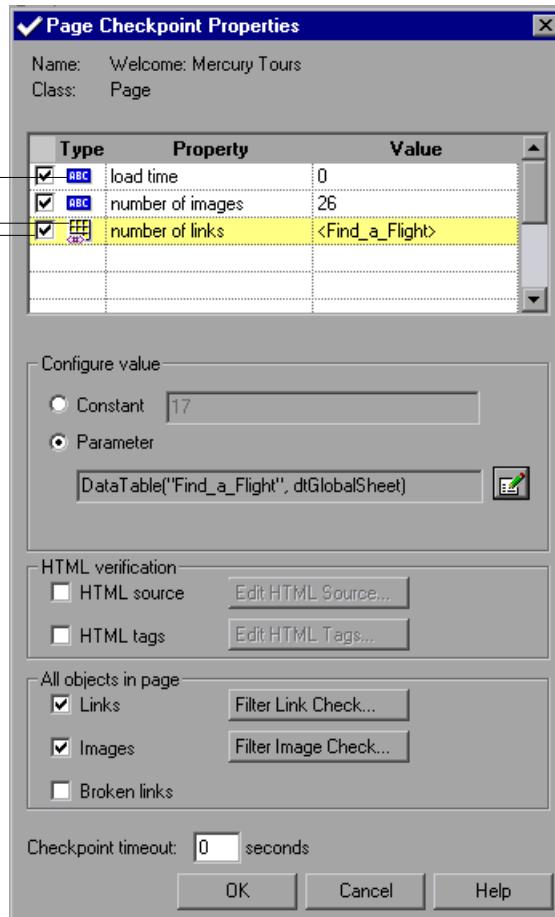
Note: You can also right-click a **Page** item in the Keyword View and choose **Insert Standard Checkpoint** to open the Page Checkpoint Properties dialog box.

- 5** Specify the settings for the checkpoint. For more information, see “Understanding the Page Checkpoint Properties Dialog Box,” below.
- 6** Click **OK** to close the dialog box. A checkpoint step is added in the Keyword View.

Note: You cannot select the **HTML Verification** options while creating a page checkpoint from the Keyword View or Active Screen. You can select these options only when creating a Page checkpoint while recording.

Understanding the Page Checkpoint Properties Dialog Box

The Page Checkpoint Properties dialog box enables you to choose which properties to check.



Identifying the Object

The top part of the dialog box displays information about the object to check:

Information	Description
Name	The title of the Web page as defined in the HTML code.
Class	The type of object. This is always Page.

Choosing Which Property to Check

The default 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, action, or component 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.
Property	The name of the property.
Value	The value of the property. Note that the value in the page will be the expected value of the property when you run your test or component unless you edit this value. For information about editing the value of a property, see Chapter 14, “Configuring Values.”

Note: By default, page checkpoints include a check on the page load time. The load time displayed in the Page Checkpoint Properties dialog box is the amount of time it took the page to load during recording. To add to the time that QuickTest allows for pages to load without causing page checkpoints to fail, increase the value of the **Add seconds to page load time** option in the Web tab of the Options dialog. For more information, see “Setting Web Testing Options” on page 639.

Configuring the Value of a Page 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 286.

Checking the HTML Verification

In the HTML verification area, you can use the following options to check the HTML source and tags of the page:

Option	Description
HTML source	Checks that the source in the Web page being tested matches the expected HTML code (the source code of the page at the time that the test or component is recorded).
Edit HTML Source (enabled only when the HTML Source check box is selected)	<p>Opens the HTML Source dialog box, which displays the expected HTML code. Edit the expected HTML source code and click OK. Note that you can also use regular expressions when editing the expected HTML source code if you click the regular expression check box at the bottom of the page. For more information on regular expressions, see “Understanding and Using Regular Expressions” on page 290.</p> <p>You can search and replace text strings in the HTML Source dialog box by right-clicking and choosing Find or Replace. For more information on the Find dialog box, see “Finding Text Strings” on page 867. For more information on the Replace dialog box, see “Replacing Text Strings” on page 869.</p>
HTML tags	Checks that the HTML tags in the Web page being tested match the expected HTML tags (the HTML tags on the page at the time that the test or component is recorded).
Edit HTML Tags (enabled only when the HTML Tags check box is selected)	<p>Opens the dialog box that displays the expected HTML tags. Edit the expected HTML tags and click OK. Note that you can also use regular expressions when editing the HTML tags if you click the regular expression check box at the bottom of the page. For more information on regular expressions, see “Understanding and Using Regular Expressions” on page 290.</p> <p>You can search and replace text strings in the Edit HTML Tags dialog box by right-clicking and choosing Find or Replace. For more information on the Find dialog box, see “Finding Text Strings” on page 867. For more information on the Replace dialog box, see “Replacing Text Strings” on page 869.</p>

Note: The **HTML verification** options are available only when creating a page checkpoint while recording.

Checking All the Objects in a Page

In the **All objects in page** area, you can check all the links, images, and broken links in a page. You can use the following options to check the objects in a page:

Option	Description
Links	Checks the functionality of the links in the page according to your selections in the Filter Link Check dialog box.
Filter Link Check (enabled only when the Links check box is selected)	Opens the Filter Link Check dialog box, which enables you to specify which hypertext links to check in the page. For additional information, see “Filtering Hypertext Links” on page 756.
Images	Checks that the images are displayed on the page according to your selections in the Filter Images Check dialog box.
Filter Image Check (enabled only when the Images check box is selected)	Opens the Filter Image Check dialog box, which enables you to specify which image sources to check in the page. For additional information, see “Filtering Image Sources” on page 757.
Broken links	Instructs QuickTest to check for broken links. Note that if you want to check only links that are targeted to your current host, you should select the Broken links - checks only links to current host option in the Web tab of the Options dialog box. For more information, see “Setting Web Testing Options” on page 639.

Setting General Checkpoint Options

The bottom part of the 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 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 make sure that the object has sufficient time to achieve that state and therefore enables the checkpoint to pass before the maximum timeout is reached.

You can see information about the checkpoint timeout, including the time interval used by QuickTest to perform the checkpoint, in the Test Results window.

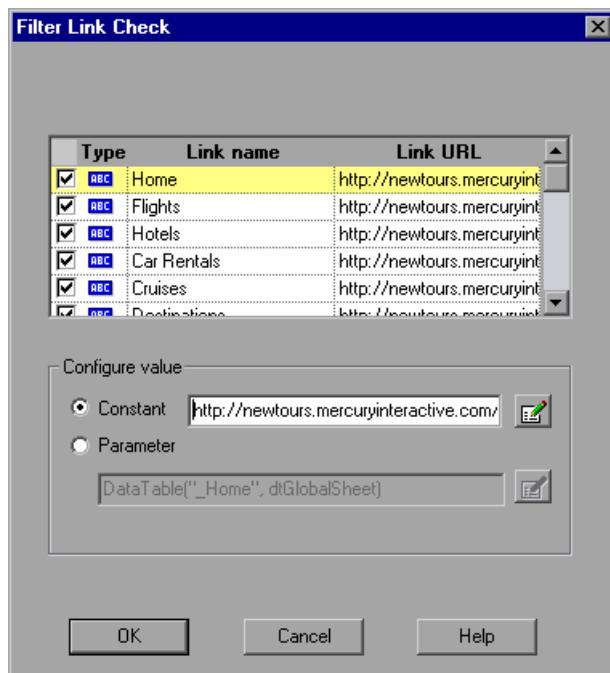
- **Insert statement**—Specifies when to perform the checkpoint in the test or component. Choose **Before current step** if you want to check the value of the object property before the highlighted step is performed. Choose **After current step** if you want to check the value of the object property after the highlighted step is performed.

Note: The **Insert statement** option is not available when adding a page checkpoint during recording or when modifying an existing page checkpoint. It is available only when adding a new page checkpoint to an existing test or component while editing your test or component.

Filtering Hypertext Links

You can filter which hypertext links to check in a page checkpoint using the Filter Link Check dialog box. You open this dialog box by clicking **Filter Link Check** in the Page Checkpoint Properties dialog box. If you select the **Links** check box in the Page Checkpoints Properties dialog box, then by default all the links on the page are selected. To instruct QuickTest not to check a particular hypertext link, clear the link's check box.

For additional information, see “Checking All the Objects in a Page” on page 754.



Choosing Which Hypertext Links to Check

You can use the following options to choose which hypertext links to check in a page checkpoint:

Pane Element	Description
Check box	<p>Each link on the page has a corresponding check box.</p> <p>To check a link, select the corresponding check box (by default all links are selected).</p> <p>To exclude a link from the page checkpoint, clear the corresponding check box.</p>
Type	<p>The  icon indicates that the target URL is currently a constant.</p> <p>The  icon indicates that the value of the property is currently a Data Table parameter.</p> <p>The  icon indicates that the value of the property is currently an environment variable parameter.</p> <p>The  icon indicates that the value of the property is currently a random number parameter.</p>
Link name	The text in the hypertext link.
Link URL	The target URL.

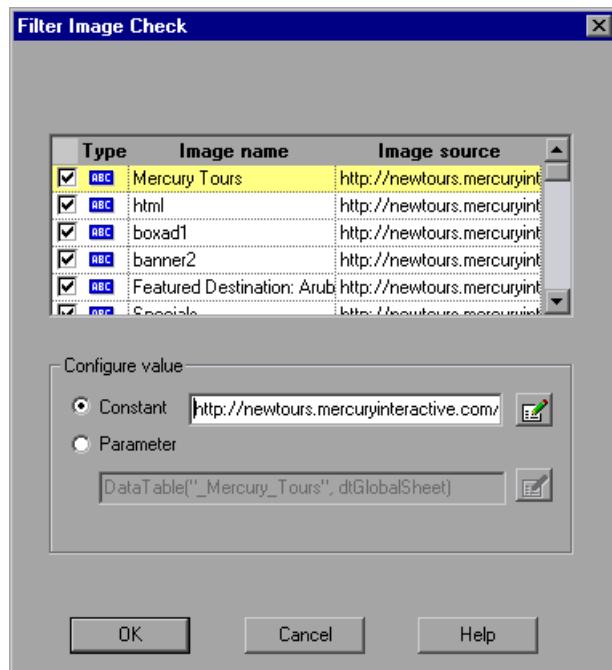
Configuring the Value of the Target URL

In the **Configure value** area, you can define the expected value of the target URL to which the hypertext links as a **Constant** or **Parameter**. For information on modifying values, see “Setting Values in the Configure Value Area” on page 286.

Filtering Image Sources

You can filter which image sources to check in a page checkpoint using the Filter Images Check dialog box. You open this dialog box by selecting **Filter Image Check** in the Page Checkpoint Properties dialog box. If you select the **Images** check box in the Page Checkpoints Properties dialog box, then, by default, all image sources on the page are selected. To instruct QuickTest not to check a particular image source, clear the image's check box.

For additional information, see “Checking All the Objects in a Page” on page 754.



Choosing Which Image Sources to Check

You can use the following options to choose which image sources to check in a page checkpoint:

Pane Element	Description
Check box	Each image source on the page has a corresponding check box. To check an image source, select the corresponding check box (by default all image sources are selected). To exclude an image source from the page checkpoint, clear the corresponding check box.
Type	The  icon indicates that the image source is currently a constant. 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.
Image name	The name of the image.
Image source	The image source file and path.

Configuring the Value of the Path of the Image Source File

In the **Configure value** area, you can define the path of the image source file as a **Constant** or **Parameter**. For information on modifying values, see “Setting Values in the Configure Value Area” on page 286.

Checking Web Content Accessibility

The Section 508 criteria for Web-based technology and information systems are based on access guidelines developed by the Web Accessibility Initiative of the World Wide Web Consortium (W3C). You can add accessibility checkpoints to help you quickly identify areas of your Web site that may not conform to the W3C Web Content Accessibility Guidelines. You can add automatic accessibility checkpoints to each page in your test or component, or you can add individual accessibility checkpoints to individual pages or frames.

Note: Accessibility Checkpoints are designed to help you easily locate the areas of your Web site that require special attention according to the W3C Web Content Accessibility Guidelines. They do not necessarily indicate whether or not your Web site conforms to the guidelines. For more information, see “Reviewing Accessibility Checkpoint Results” on page 763.

Setting Accessibility Checkpoint Preferences

You can set accessibility checkpoint preferences in the Advanced Web Options dialog box and view them in the Accessibility Checkpoint Properties dialog box. All accessibility checkpoints in your test or component use the options that are selected in the Advanced Web Options dialog box at the time of the run session. For information about the accessibility checkpoint options, see “Advanced Web Options” on page 644.

Automatic Accessibility Checkpoints

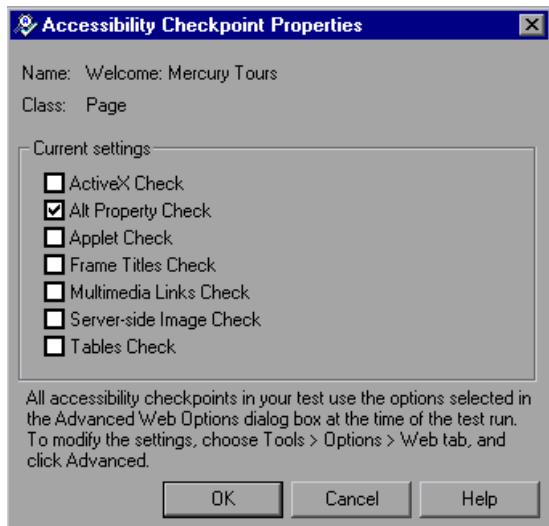
You can instruct QuickTest to create automatic accessibility checkpoints for every page in all tests or components by selecting the **Add Automatic accessibility checkpoint to each Web page while recording** check box in the Advanced Web Options dialog box (click the **Advanced** button in the Web tab of the Options dialog box). If you select this option, an accessibility checkpoint is inserted for each page as you record.

Creating Individual Accessibility Checkpoints

If you did not choose to add accessibility checkpoints automatically while recording, you can add an accessibility checkpoint to help you quickly identify areas of a particular Web page or frame that may not conform to the W3C Web Content Accessibility Guidelines. You can add accessibility checkpoints either while recording or editing your test or component.

To add an accessibility checkpoint while recording:

- 1 Navigate to a page where you want to add an accessibility checkpoint.
- 2 Choose **Insert > Checkpoint > Accessibility Checkpoint**, or click the **Insert Checkpoint** button and click in the page or frame you want to check.
 - If the page contains frames, the Object Selection - Accessibility Checkpoint Properties dialog box opens. Select the **Page or Frame** item you want to check and click **OK**. The Accessibility Checkpoint Properties dialog box opens.
 - If the page does not contain frames, the Accessibility Checkpoint Properties dialog box opens. The dialog box displays the object's name, class (this is always Page or Frame), and the options that are currently selected. You can modify the option settings in the Advanced Web Options dialog box. For more information, see page 644.

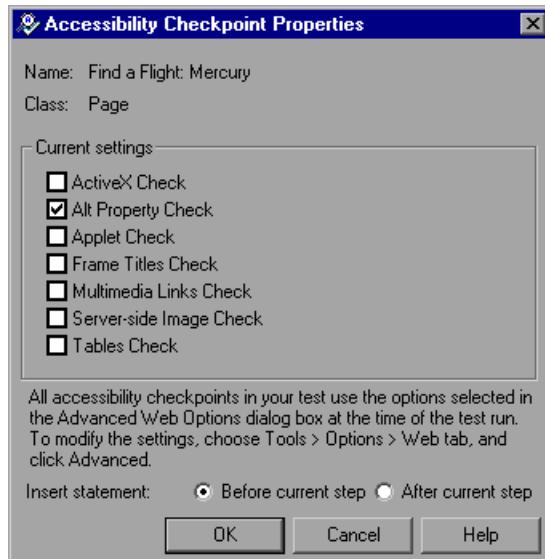


- 3 Click **OK** to close the dialog box. A checkpoint step is added in the Keyword View.

To add an accessibility checkpoint while editing your test or component:



- 1 Make sure the **Active Screen** button is selected.
- 2 Click a step in your test or component where you want to add a checkpoint. The Active Screen displays the HTML source corresponding to the highlighted step.
- 3 Right-click anywhere on the Active Screen, and choose **Insert Accessibility Checkpoint**.
 - If the page contains frames, the Object Selection - Accessibility Checkpoint Properties dialog box opens. Select the **Page** or **Frame** item and click **OK**. The Accessibility Checkpoint Properties dialog box opens.
 - If the page does not contain frames, the Accessibility Checkpoint Properties dialog box opens. The dialog box displays the options that are currently selected. You can modify the option settings in the Advanced Web Options dialog box. For more information, see page 644.



Choose **Before current step** if you want to check the accessibility elements before the highlighted step is performed. Choose **After current step** if you want to check the accessibility elements after the highlighted step is performed.

Note: The **Insert statement** option is not available when adding a page checkpoint during recording or when modifying an existing page checkpoint. It is available only when adding a new page checkpoint to an existing test or component while editing your test or component.

- 4 Click **OK** to close the dialog box. A checkpoint step is added in the Keyword View.

Reviewing Accessibility Checkpoint Results

When you include accessibility checkpoints in your test or component, the Test Results window displays the results of each accessibility option that you checked.

The Test Results window displays a separate step for each accessibility option that was checked in each checkpoint. The results details provide information that can help you pinpoint parts of your Web site or application that may not conform to the W3C Web Content Accessibility Guidelines. The information provided for each check is based on the W3C requirements.

For more information on accessibility checkpoint results, see “Analyzing Accessibility Checkpoint Results” on page 573.

Accessing Password-Protected Resources in the Active Screen

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 or component. This ensures that the disk space used by the Active Screen pages captured with your test or component are not affected by the file size of the resources displayed on the page.

For this reason, a page in the Active Screen (or in your test or component results) may require a user name and password to access certain images or other resources within the page. If this is the case, a pop-up login window may open when you select a step corresponding to the page, or you may note that images or other resources are missing from the page.

For example, the formatting of your page may look very different than the actual page on your Web site if the cascading style sheet (CSS) referenced in the page is password-protected, and therefore could not be downloaded to the Active Screen.

You may need to use one or both of the following methods to access your password-protected resources, depending on the password-protection mechanism used by your Web server:

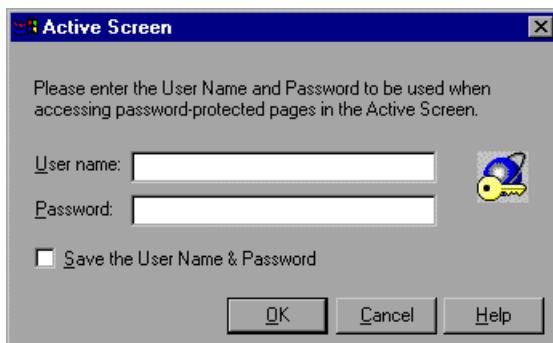
- **Standard Authentication**—If your server uses a standard authentication mechanism, you can enter the login information in the Web tab of the Test Settings dialog box or Business Component Settings dialog box. QuickTest saves this information with your test or component and automatically enters the login information each time you select to display an Active Screen page that requires the information.
- **Advanced Authentication**—If your server uses a more complex authentication mechanism, you may need to log in to the Web site manually using QuickTest's Advanced Authentication dialog box. This gives the Active Screen access to password-protected resources in your Active Screen pages for the duration of your QuickTest session. When using this method, you must log in to your Web site in the Advanced Authentication dialog box each time you open the test or component in a new QuickTest session.

In most cases, the automatic login is sufficient. In some cases, you must use the manual login method. In rare cases, you may need to use both login mechanisms to enable access to all resources in your Active Screen pages.

Note: If your Web site is not password-protected, but you are still unable to view images or other resources on your Active Screen, you may not be connected to the Internet, the Web server may be down, or the source path that was captured with the Active Screen page may no longer be accurate.

Using the Standard Authentication Mechanism

If you select a step in your test or component or results, and an Active Screen login window opens over the Active Screen or results details display, then one or more images or other resources in the Active Screen may be password-protected.



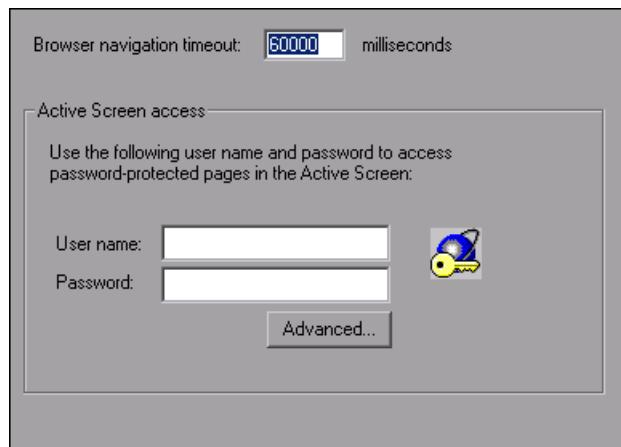
To prevent the pop-up login window from opening and to ensure that all images and resources are displayed in the Active Screen and results each time you open the test or component, you can use QuickTest's automatic Active Screen login mechanism.

To enable the mechanism, you can select **Save the User Name and Password** in the pop-up login window the first time it opens. This adds the login information to the **Active Screen access** area in the Web tab of the Test Settings dialog box or Business Component Settings dialog box.

Alternatively, you can add the login information manually in the Web tab of the Test Settings dialog box or Business Component Settings dialog box.

To set Active Screen access information in the Test Settings dialog box or Business Component Settings dialog box:

- 1 Choose **Test > Settings** or **Component > Settings**. The Test Settings dialog box or Business Component Settings dialog box opens.
- 2 Click the **Web** tab.



- 3 Enter the **User name** and **Password** for the Web site or Web page containing the password-protected resources.
- 4 Click **OK** to save your changes and close the dialog box.
- 5 Refresh the Active Screen by selecting a new step in the Keyword View or toggle the Active Screen button to redisplay the Active Screen. Confirm that the page is displayed correctly.

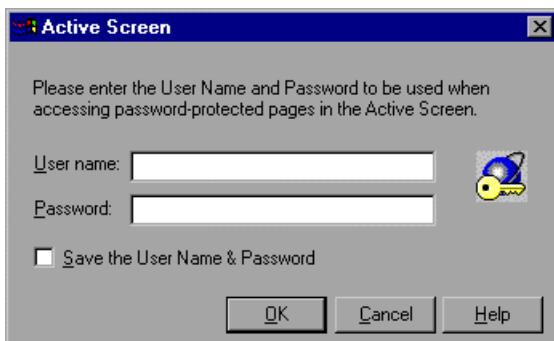


If one or more resources are still missing or displayed incorrectly, you may need to use the Advanced Authentication mechanism.

For more information on the Web tab of the Test Settings dialog box or Business Component Settings dialog box, see “Defining Web Settings for Your Test or Component” on page 692.

For more information on the Advanced Authentication mechanism, see page 767.

Using the Advanced Authentication Mechanism



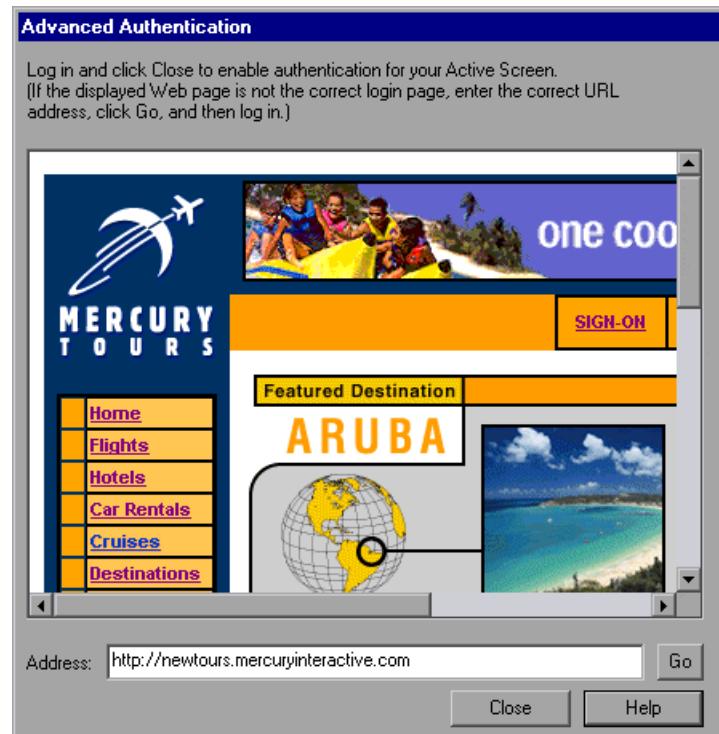
Depending on the authentication mechanisms used to password-protect resources on a Web site, QuickTest's automatic Active Screen login mechanism may not be sufficient.

To enable the Active Screen to access the resources on such a site, you must log in to your site using the Advanced Authentication dialog box. When you log in this way, you remain logged in to the site for the duration of the QuickTest session. If you close and reopen QuickTest and then reopen your test or component, you must log in again.

Note: If the site to which you log in has an inactivity timeout after which you are automatically logged out of the Web site, you may need to log in using the Advanced Authentication dialog box more than once while editing your test or component to re-enable access to your Active Screen pages.

To log in to your Web site using the advanced authentication mechanism:

- 1** Choose **Test > Settings** or **Component > Settings**. The Test Settings dialog box or Business Component Settings dialog box opens.
- 2** Click the **Web** tab.
- 3** Click the **Advanced** button. The Advanced Authentication dialog box opens.



The browser window in the dialog box displays the default Web page for the test or component according to the following guidelines:

-  The first time you open this dialog box for a given test, the browser window displays the URL address set for the test in the Web tab of the Record and Run Settings dialog box.
 -  The first time you open this dialog box for a given component, the browser window displays the URL address of the Mercury Tours sample Web site: <http://newtours.mercuryinteractive.com>.
 - If you navigate to a new URL address using this dialog box, that address becomes the default Advanced Authentication page for this test or component.
- 4** If the displayed Web page is not the correct page for logging in to your site, enter the correct URL address in the **Address** box and click **Go**. Otherwise, proceed to step 5.
- 5** Enter your login information in the page displayed in the Advanced Authentication browser window.
- 6** When the login process is complete, click **Close**. The Advanced Authentication dialog box closes, but the login session remains open for the remainder of your QuickTest session (or until the Web site's inactivity timeout is exceeded).
-  **7** Refresh the Active Screen by selecting a new step in the Keyword View or toggle the Active Screen button to redisplay the Active Screen. Confirm that the pages are displayed correctly.

If you still cannot view images or other resources on your Active Screen, you may not be connected to the Internet, the Web server may be down, or the source path that was captured with the Active Screen page may no longer be accurate.

Activating Methods Associated with a Web Object

In the Expert View, you can use the “Object” property to activate the method for a Web object. Activating the method for a Web object has the following syntax:

WebObjectName.Object.Method_to_activate()

For example, suppose you have the following statement in your script:

```
document.MyForm.MyHiddenField.value = "My New Text"
```

The following example achieves the same thing by using the Object property, where MyDoc is the DOM's document:

```
Dim MyDoc  
Set MyDoc = Browser(browser_name).page(page_name).Object  
MyDoc.MyForm.MyHiddenField.value = "My New Text"
```

In this example, LinksCollecton is assigned to the link collection of the page through the Object property. Then, a message box pops for each of the links, with its innerHTML text.

```
Dim LinksCollection, link  
Set LinksCollection = Browser(browser_name).Page(page_name).Object.links  
For Each link in LinksCollection  
    MsgBox link.innerHTML  
Next
```

For additional information about the **Object** property (**.Object**), see “Retrieving and Setting Test Object Property Values” on page 898.

For a list of a Web object's internal properties and methods see:

http://msdn.microsoft.com/library/default.asp?url=/workshop/author/dhtml/reference/objects/obj_document.asp

Using Scripting Methods with Web Objects

QuickTest provides several scripting methods that you can use with Web objects. You can record some of these methods while recording on Web objects. You can enter statements manually with the other methods in the Expert View. For more information about programming in the Expert View, see Chapter 36, “Working with the Expert View.”

For additional information on these methods, refer to the *QuickTest Professional Object Model Reference*.

31

Testing Visual Basic Applications

QuickTest supports the testing of Visual Basic 6.0 applications.

Note: To test Visual Basic applications, you must install and load the Visual Basic Add-in. To test Visual Basic .NET Windows Forms applications, you must install and load the .NET Add-in. Note that the .NET Add-in is not included with your QuickTest installation. To obtain this add-in, contact your QuickTest supplier or Mercury Interactive customer support.

This chapter describes:

- About Testing Visual Basic Applications
- Recording and Running on Visual Basic Applications
- Checking Visual Basic Objects
- Using Visual Basic Objects and Methods to Enhance Your Test or Component

About Testing Visual Basic Applications

QuickTest recognizes Visual Basic objects and generates Visual Basic-specific statements when you record tests or components on a Visual Basic application.

You check the properties of Visual Basic objects the same way you check the properties of any other object. You can also output property or text values from the objects in your Visual Basic application.

The following checkpoints and output values are supported when testing Visual Basic objects:

Type	Checkpoint	Output	For more information, see:
Standard	Yes	Yes	"Checking Object Property Values," on page 123, and "Outputting Property Values," on page 255
Text	Yes	Yes	"Creating a Text Checkpoint," on page 153, and "Outputting Text Values," on page 268
Text Area	Yes	Yes	"Creating a Text Area Checkpoint," on page 156, and "Outputting Text Values," on page 268
Bitmap	Yes	No	"Checking Bitmaps" on page 169

In the Expert View or using the **Insert > Step** option, you can activate Visual Basic methods, retrieve and set the values of properties (including run-time properties), and check that objects exist.

Note that this chapter provides examples using both the Keyword View and the Expert View. Some of the information provided and procedures described require working in the Expert View. For information on working in the Expert View, see Chapter 36, "Working with the Expert View."

Recording and Running on Visual Basic Applications

QuickTest records and runs steps on Visual Basic 6.0 applications as it does on any other object. You can view the details of your run session in the Test Results window. For more information on running tests and components, see Chapter 21, “Running Tests and Components.” For more information on viewing run results, see Chapter 23, “Analyzing Test Results.”

It is recommended that you begin your recording session before opening the Visual Basic application on which you want to record. You can choose one of the following options:

- Select **Record and run on these applications** in the Windows Applications tab of the Record and Run Settings dialog box and specify the name and full path of the executable file of your Visual Basic application. QuickTest automatically opens this application when the recording session begins.
- Select **Record and run on any application** in the Windows Applications tab of the Record and Run Settings dialog box, and click **OK** to begin recording. When QuickTest begins recording, open your Visual Basic application.

For more information on the Record and Run Settings dialog box, see Chapter 26, “Setting Record and Run Options.”

Note: The Visual Basic application must be executed after setting the Record and Run options.



QuickTest records Visual Basic-specific statements for the operations performed on the Visual Basic objects. When you record on a Visual Basic application, QuickTest displays a Visual Basic icon next to the step in the Keyword View. For example, if you record typing a name in a Visual Basic text field, the Keyword View may be displayed as follows:

Item	Operation	Value	Documentation
▼ Action1			
▼ Login			
txtAgentsName	Set	"John"	Enter "John" in the "txtAgentsName" edit box.

QuickTest records this step in the Expert View as:

```
VbWindow("Login").VbEdit("txtAgentsName").Set "John"
```

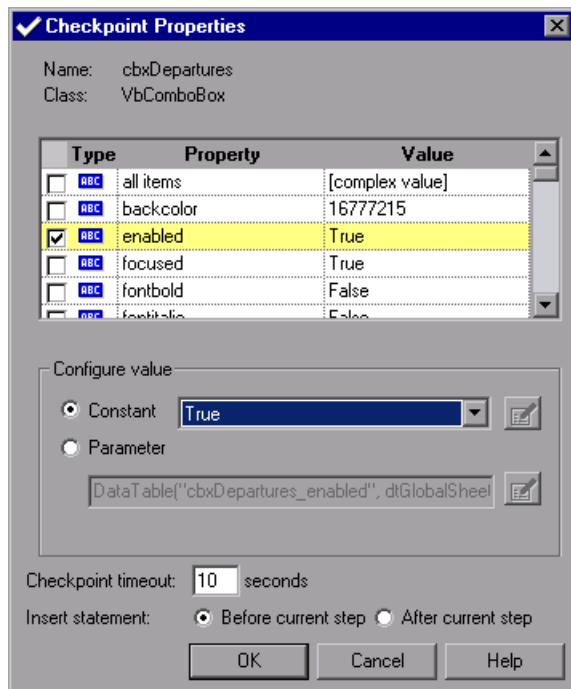
You run tests or components on Visual Basic objects just as you would with any other QuickTest object. The results tree displays the same Visual Basic object icon as that used in the Keyword View.

For more information on running tests or components, see Chapter 21, “Running Tests and Components.” For information on viewing results, see Chapter 23, “Analyzing Test Results.”

Checking Visual Basic Objects

You can check Visual Basic objects by inserting a standard checkpoint. For information on standard checkpoints, see Chapter 7, “Checking Object Property Values.” For information on the properties available for a given Visual Basic object, refer to the *QuickTest Professional Object Model Reference*.

For example, you can create a checkpoint to check the enabled property of a Visual Basic combo box.



A Visual Basic checkpoint is similar in appearance to other checkpoint statements. In the Keyword View, the statement may be displayed as follows:

	Operation	Value	Documentation
 cbxDepartures	Check	Check...	Check whether the "cbxDepartures" list has the proper values for the selected

In the Expert View, the above checkpoint statement would appear as:

```
VbWindow(" MainForm").VbComboBox("cbxDepartures").
Check CheckPoint("cbxDepartures")
```

Using Visual Basic Objects and Methods to Enhance Your Test or Component

QuickTest provides several methods that you can use with Visual Basic objects. You can record some of these methods while recording on Visual Basic objects. You can add additional functionality to your test or component by entering statements manually in the Expert View or by using the Step Generator. For more information about programming in the Expert View, see Chapter 36, “Working with the Expert View.” For more information about using the Step Generator, see “Inserting Steps Using the Step Generator” on page 467.

In addition to the Visual Basic-specific objects and methods, you can use the **Object** property to access the internal properties of any run-time object. The **Object** property is available for all Visual Basic objects.

You can also use the **Object** property to directly access Visual Basic object properties and activate their methods.

For example, you can set the focus and caption of a button using code similar to the following:

```
Set theButton = VbWindow("frmMain").VbButton("OK").Object  
theButton.SetFocus  
theButton.Caption = "Yes"
```

The **Object** property is also useful for checking the value of properties that are not available using a standard Visual Basic checkpoint.

For additional information on Visual Basic objects and methods, refer to the *QuickTest Professional Object Model Reference*.

32

Testing ActiveX Controls

QuickTest supports testing on ActiveX controls in any application.

This chapter describes:

- About Testing ActiveX Controls
- Recording and Running on ActiveX Controls
- Checking ActiveX Controls
- Activating an ActiveX Control Method
- Using Scripting Methods with ActiveX Controls

About Testing ActiveX Controls

Many applications include ActiveX controls developed with third-party vendors. QuickTest can record and run tests or components on these controls as well as check their properties.

You can check the properties of an ActiveX control as you check the properties of any other object. You can also output property or text values from the objects in your ActiveX application.

The following checkpoints and output values are supported when testing ActiveX objects:

Type	Checkpoint	Output	For more information, see:
Standard	Yes	Yes	“Checking Object Property Values,” on page 123, and “Outputting Property Values,” on page 255
Text	Yes	Yes	“Creating a Text Checkpoint,” on page 153, and “Outputting Text Values,” on page 268 Note: Text checkpoints and output values are not supported for windowless ActiveX controls.
Text Area	Yes	Yes	“Creating a Text Area Checkpoint,” on page 156, and “Creating Text Area Output Values,” on page 270
Table	Yes	Yes	“Checking Tables and Databases,” on page 133, and “Outputting Property Values,” on page 255
Bitmap	Yes	No	“Checking Bitmaps” on page 169

In the Expert View, or using the **Insert > Step** option, you can activate ActiveX control methods, retrieve and set the values of properties (including run-time properties), and check that objects exist.

Note that this chapter provides examples using both the Keyword View and the Expert View. Some of the information provided and procedures described require working in the Expert View. For information on working in the Expert View, see Chapter 36, “Working with the Expert View.”

For information on supported controls and versions, refer to the *QuickTest Professional Readme*.

Recording and Running on ActiveX Controls

QuickTest records and runs steps on ActiveX controls as it does on any other object. You can view the details of your run session in the Test Results window. For more information on running tests and components, see Chapter 21, “Running Tests and Components.” For more information on viewing run results, see Chapter 23, “Analyzing Test Results.”

It is recommended that you begin your recording session before opening the application containing the ActiveX controls on which you want to record. You can choose one of the following options:

- Select **Record and run on these applications** in the Windows Applications tab of the Record and Run Settings dialog box and specify the name and full path of the executable file of your application. QuickTest automatically opens this application when the recording session begins.
- Select **Record and run on any application** in the Windows Applications tab of the Record and Run Settings dialog box, and click **OK** to begin recording. When QuickTest begins recording, open your application.

For more information on the Record and Run Settings dialog box, see Chapter 26, “Setting Record and Run Options.”

Note: The application containing the ActiveX controls on which you want to record must be executed after setting the Record and Run options.



QuickTest records clicks inside the ActiveX control. When you record on an ActiveX control, QuickTest displays an ActiveX icon next to the step in the Keyword View. For example, if you record clicking on a calendar that is an ActiveX control, the Keyword View may be displayed as follows:

Item	Operation	Value	Documentation
▶  Action1			
▶  ActiveX Calendars	Activate		Make the "ActiveX Calendars" dialog box active.
▶  SMonth Control	Click	44,72	Click the "SMonth Control" ActiveX object.

QuickTest records this step in the Expert View as:

```
Dialog("ActiveX Calendars").ActiveX("SMonth Control").Click 44,72
```

Recording on an ActiveX control has the following syntax in the Expert View:

...ActiveX (ActiveX_control).Method (arguments)

QuickTest can record on standard controls within an ActiveX control. For example, suppose your ActiveX control is a calendar that contains a drop-down list from which you can choose the month. If you click in the list to select the month of May, QuickTest records this in the Keyword View as follows:

Item	Operation	Value	Documentation
Action1			
ActiveX Calendars	Activate		Make the "ActiveX Calendars" dialog box active.
SMonth Control	Select	"May"	Select item "May" in the "ComboBox" list.
SMonth Control	Click	44,72	Click the "SMonth Control" ActiveX object.

QuickTest records this step in the Expert View as:

```
Dialog("ActiveX Calendars").ActiveX("SMonth Control").  
WinComboBox("ComboBox").Select "May"
```

Note: If an ActiveX control contains another ActiveX control, then QuickTest can record and run on this internal control as well.

You run tests or components on ActiveX controls just as you would with any other QuickTest objects. The results tree displays the same ActiveX control icon as that used in the Keyword View. The bottom right pane of the Test Results window displays the ActiveX control that was captured during the run session and highlights the ActiveX control for each step in the results tree.

For more information about running tests and components, see Chapter 21, “Running Tests and Components.” For more information about results, see Chapter 23, “Analyzing Test Results.”

Checking ActiveX Controls

You create a checkpoint on an ActiveX control as you create a checkpoint on any standard object. When you create a checkpoint on an ActiveX control, QuickTest captures the ActiveX control properties and their values as it does for any standard object. The properties you can check for an ActiveX control depend on the properties of the ActiveX control. By default, when you create a checkpoint on an ActiveX control, QuickTest captures all the properties for an ActiveX control, but it does not select any properties to check.

For example, you can create a checkpoint on an ActiveX control that is a calendar to check the current date in the calendar.

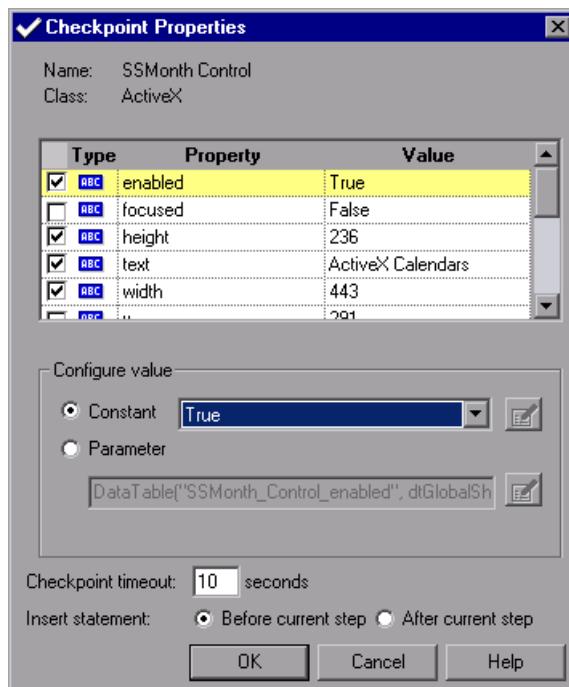
Note: Unlike table objects in other environments, when you check an ActiveX table, you can choose to check the table content and/or the table object properties. For more information, see “Understanding the Table/Database Checkpoint Properties Dialog Box” on page 140.

To create a checkpoint on an ActiveX control:

- 1 Right-click the ActiveX control you want to check in the Active Screen and select **Insert Standard Checkpoint**. The Object Selection - Checkpoint Properties dialog box opens.



- 2** Select the ActiveX control for which you want to create a checkpoint and click **OK**. The Checkpoint Properties dialog box opens and displays all the properties for the ActiveX control.



3 Select the properties you want to check in the check box column.

4 For each property, select the checkpoint options you want to apply.

Click **OK**. A checkpoint step is added in the Keyword View with an ActiveX icon.

For more information on creating checkpoints, see Chapter 6, “Understanding Checkpoints.”

Activating an ActiveX Control Method

In the Expert View, you can use the **Object** property to activate the method for an ActiveX control. The list of available methods depends on the ActiveX control. Activating the method for an ActiveX control has the following syntax:

...ActiveX (ActiveX_control).Object.Method_to_activate()

For example, suppose the **MakeObjVisible** method is supported for your ActiveX control. To activate the **MakeObjVisible** method, you insert the following statement into your test or component script:

```
Browser("Home").Page("HomePage").ActiveX("Calendar")
    .Object.MakeObjVisible()
```

For additional information about the **Object** property (**.Object**), see “Retrieving and Setting Test Object Property Values” on page 898.

Using Scripting Methods with ActiveX Controls

QuickTest provides several scripting methods that you can use with ActiveX controls. You can record some of these methods while recording on ActiveX controls. You can enter statements manually with the other methods in the Expert View or using the Step Generator. For more information about programming in the Expert View, see Chapter 36, “Working with the Expert View.” For more information on the Step Generator, see “Inserting Steps Using the Step Generator” on page 467.

For additional information on these methods, refer to the *QuickTest Professional Object Model Reference*.

Note: When creating a programmatic description for an ActiveX test object and the relevant run-time object is windowless (has no window handle associated with it), you must add the **windowless** property to the description and set its value to **True**.

For example:

```
Set ButDesc = Description.Create  
ButDesc("ProgId").Value = "Forms.CommandButton.1"  
ButDesc("Caption").Value = "OK"  
ButDesc("Windowless").Value = True  
Window("Form1").AcxButton(ButDesc).Click
```

For more information on using programmatic descriptions, see “Using Programmatic Descriptions” on page 878.

Part VII

Advanced Features

33

Configuring Object Identification

When you record an operation on an object, QuickTest learns a set of properties and values that uniquely describe the object within its parent object. 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 describes:

- About Configuring Object Identification
- Understanding the Object Identification Dialog Box
- Configuring Smart Identification
- Mapping User-Defined Test Object Classes

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 an object you record, QuickTest can add some assistive properties and/or an ordinal identifier in order 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 records the value for the selected ordinal identifier. For more information, see “Selecting an Ordinal Identifier” on page 796.

When you run a test or component, 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 it finds more than one object that matches, 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 recorded 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 record 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 or component.

Understanding the Object Identification Dialog Box

You use the main screen of the Object Identification dialog box to set mandatory and assistive recording 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.

Note: The recorded and smart identification properties of certain test objects cannot be configured, for example, the WinMenu, VbLabel, VbObject, 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 Recording Properties” on page 792
- “Selecting an Ordinal Identifier” on page 796
- “Enabling and Disabling Smart Identification” on page 801
- “Restoring Default Object Identification Settings for all Test Objects” on page 802
- “Generating Automation Scripts for Your Object Identification Settings” on page 803
- “Configuring Smart Identification” on page 803
- “Mapping User-Defined Test Object Classes” on page 812

Configuring Mandatory and Assistive Recording 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 you record on 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.

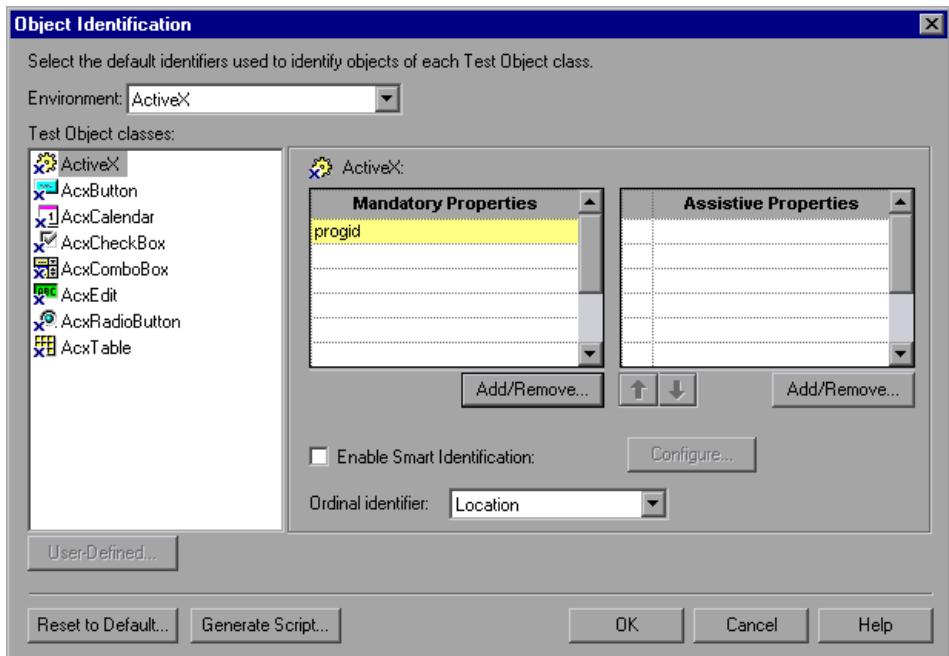
For example, the default mandatory properties for a Web Image object are the **alt**, **html tag**, and **image type** properties. There are no default assistive properties defined. Suppose your Web site contains several space holders for different collections of rotating advertisements. You want to record a test or component 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 recorded when you create the test or component, and most likely another **alt** value will be captured when you run the test or component, causing the run to fail. In this case, you could remove the **alt** property from the Web Image mandatory properties list. Instead, since each ad 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 (like a logo displayed on the bottom and top of a page), the Web designer adds a special **ID** property to the Image tag. Thus, the mandatory properties are sufficient to create a unique description for images that are only displayed 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 Choose **Tools > Object Identification**. The Object Identification dialog box opens.

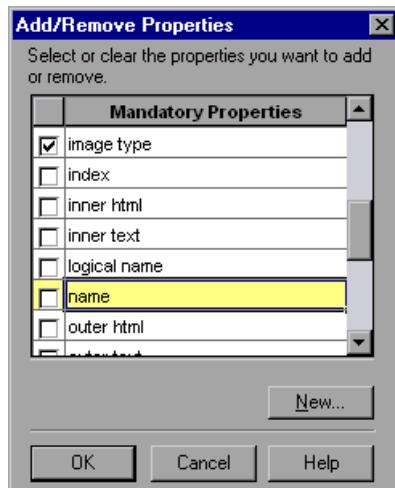


- 2 Select the appropriate environment in the **Environment** list. The test object classes associated with the selected environment are displayed in the **Test object classes** list.

Note: The environments included in the Environment list correspond to the loaded add-in environments. For more information on loading add-ins, see Chapter 29, “Working with QuickTest Add-Ins.”

- 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.



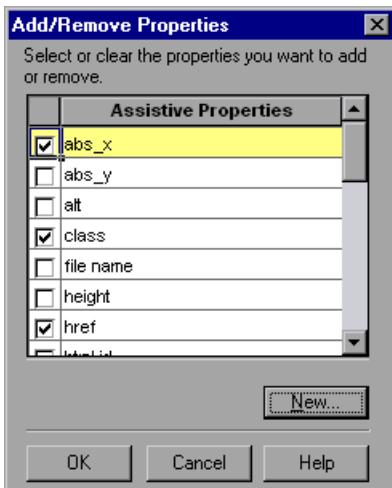
- 5 Select the properties you want to include in the Mandatory Properties list and/or clear the properties you want to remove from the list.

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 in 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 you record a test or component 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 recording the mandatory and assistive properties specified in the Object Identification dialog box, QuickTest can also record 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.

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

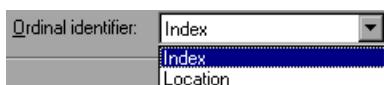
Further, even if QuickTest records an ordinal identifier, it does not use it to identify the object during the run session, unless neither the recorded description nor the Smart Identification mechanism are able to single out the object in your application.

If the other test object properties are sufficient to identify the object during a run session, the ordinal identifier is ignored.

There are three types of ordinal identifiers that QuickTest can use 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 798.
- **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 799.
- **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 800.

An ordinal identifier is selected by default for each test object class. To modify the selected ordinal identifier, select the desired type from the **Ordinal identifier** box.



Tip: If QuickTest successfully creates a unique test object description while recording using the mandatory and assistive properties, it does not record an ordinal identifier value. You can add an ordinal identifier to an object's test object properties at a later time using the **Add/Remove** option from the Object Properties or Object Repository dialog box. For more information, see Chapter 4, "Managing Test Objects."

Identifying an Object Using the Index Property

During recording, QuickTest can assign a value to the **Index** test object property of an object in order 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.

Note that **Index** property values are object-specific. Thus, if you use **Index:=3** to describe a WebEdit test object, QuickTest searches for the fourth WebEdit object in the page.

If you use **Index:=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")`

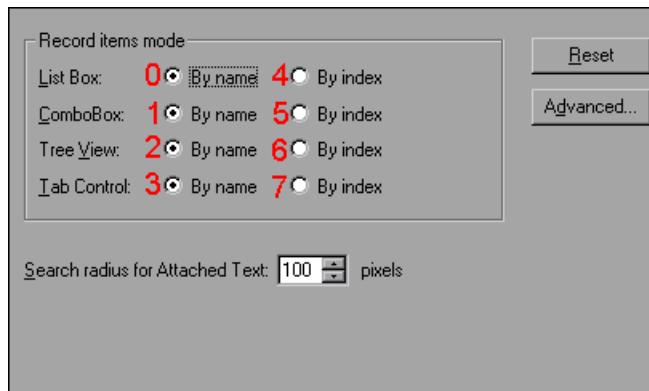
The following description, however, refers to the second item in the list above, as that is the first object of any type (WebElement) with the name `UserName`.

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

Identifying an Object Using the Location Property

During recording, QuickTest can assign a value to the **Location** test object property of an object in order 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 the same properties. The first occurrence of the object is 0. Values are assigned in columns from top to bottom, and left to right.

For example, the radio buttons in the dialog box below are numbered according to their location property.



Note that **Location** property values are object-specific. Thus, 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.

If you use `Location:=3` to describe a `WinObject` object, however, 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 you have a dialog box with 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 description below refers to the third item in the list above, as it is the first check box object on the page with the name Add/Remove.

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

The following description, however, refers to the second item in the list above, 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

During recording, if QuickTest is unable to uniquely identify a browser object based solely on its test object description, it assigns a value to the **CreationTime** test object property. This value indicates the order in which the browser was opened relative to other open browsers with an otherwise identical description.

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 you record a test or component on three otherwise identical browsers, opened at 9:01 pm, 9:03 pm, and 9:05 pm, QuickTest assigns the CreationTime values 0 to the 9:01 pm browser, 1 to the 9:03 pm browser, and 2 to the 9:05 pm browser.

At 10:30 pm, when you rerun your test or component, suppose the browsers are opened at 10:31 pm, 10:33 pm, and 10:34 pm. QuickTest identifies the 10:31 pm browser with the browser test object with CreationTime = 0, the 10:33 pm browser with the test object with CreationTime = 1, and the 10:34 pm browser with the test object with CreationTime = 2.

If you have several open browsers, the one with the highest CreationTime is the last one that was opened, and the one with the lowest CreationTime is the first one that was opened. For example, if you have three or more browsers open, the one with CreationTime = 2 is the third browser that was opened. If you have less than (or exactly) seven browsers, the one with CreationTime = 6 is the last browser that was opened.

If a step was recorded on a browser with, for example, CreationTime = 6, and there is no open browser with this CreationTime value, the step will run on the browser that is currently open with the highest CreationTime value. For example, if two browsers are currently open during the run session, with CreationTime = 0 and CreationTime = 1, then the step does not fail because it cannot find a browser with CreationTime = 6, but instead runs on the last browser opened, in this case, 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 record the property values of all properties specified as the object's base filter 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 with Smart Identification configuration defined. However, even if you define a Smart Identification configuration for a test object class, you may not always want to record the Smart Identification property values. If you do not want to record the Smart Identification properties, clear the **Enable Smart Identification** check box.

Note: Even if you choose to record 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 tab of the Test Settings dialog box. For more information, see Chapter 4, “Managing Test Objects” and Chapter 25, “Defining Run Settings for Your Test.”

However, if you do not record 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 803.

Restoring Default Object Identification Settings for all Test Objects

You can click the **Reset to Default** button to restore the default settings for all elements of the Object Identification dialog box for all environments listed in the Environment box (corresponding to the add-ins that are currently loaded).

Note that the **Reset to Default** button applies to the main object identification properties and the smart identification properties for all QuickTest test object classes and currently available in the Object Identification dialog box and to the user-defined test object classes that are mapped in the Object Mapping dialog box.

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 919, or refer to the *QuickTest Automation Object Model Reference* (**Help > QuickTest Automation Object Model Reference**).

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 recorded description have changed.

When QuickTest uses the recorded description to identify an object, it searches for an object that matches every one 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 recorded object description, or if it finds more than one object that fits the description, then QuickTest ignores the recorded 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, and thus, if configured logically, a Smart Identification definition can probably help QuickTest identify an object, if it is present, even when the recorded 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 as they are unlikely to change on a regular basis, but which 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 recorded description), it follows the following process to identify the object:

- 1 QuickTest “forgets” the recorded 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 property list.
- 2 From that list of objects, QuickTest filters out any object 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 recorded description plus the ordinal identifier to identify the object.

If the combined recorded 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 recorded test or component 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 recorded description, the results receive a warning status and indicate that the Smart Identification mechanism was used.

If the Smart Identification mechanism cannot successfully identify the object, QuickTest uses the recorded description plus the ordinal identifier to identify the object. If the object is still not identified, the test or component 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 587.

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 or component:

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

When you recorded your test or component, QuickTest recorded the following object description for the Login image:

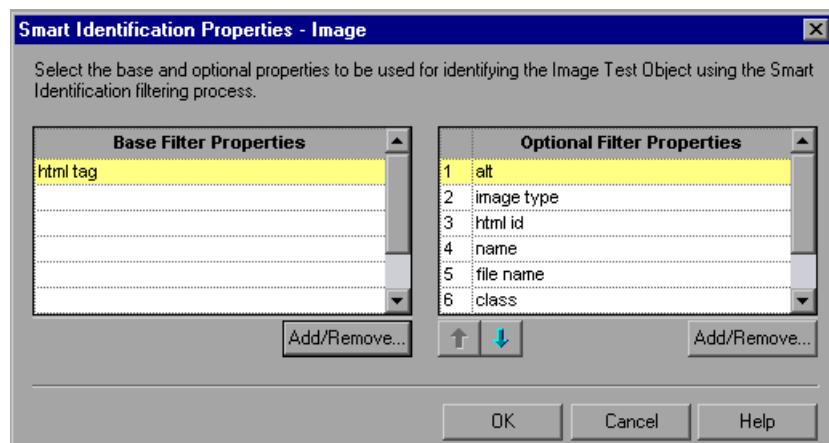
Type	Property	Value
RBC	alt	Login
RBC	html tag	INPUT
RBC	image type	Image Button

At some point after you recorded your test or component, however, a second login button (for logging into a 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 recorded description. Thus, when you run your test or component, QuickTest is unable to identify the Login button based on the recorded 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 recorded the values of the following properties when you recorded the click on the Login image:



The recorded values are as follows:

Base Filter Properties:

Property	Value
html tag	INPUT
image type	Image Button

Optional Filter Properties:

Property	Value
alt	Login
name	login
file name	login.gif
class	<null>
visible	1

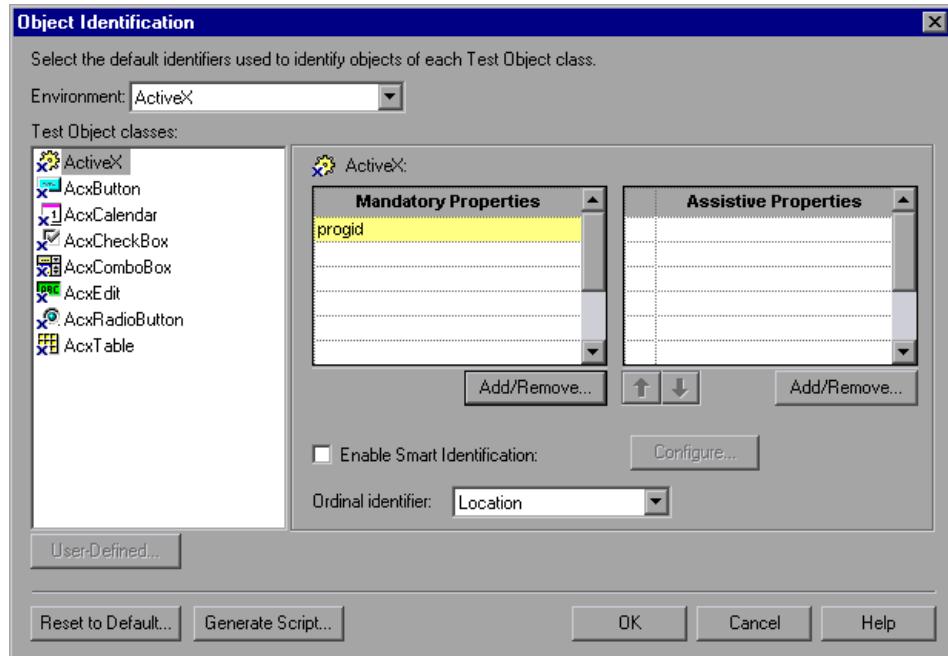
- 2 QuickTest begins the Smart Identification process by identifying the five objects on the Mercury Tours page that match the base filter properties definition (**html tag** = INPUT and **image type** = Image Button). QuickTest considers these to be the object candidates and begins checking the object 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 **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.
- 5 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 Choose **Tools > Object Identification**. The Object Identification dialog box opens.

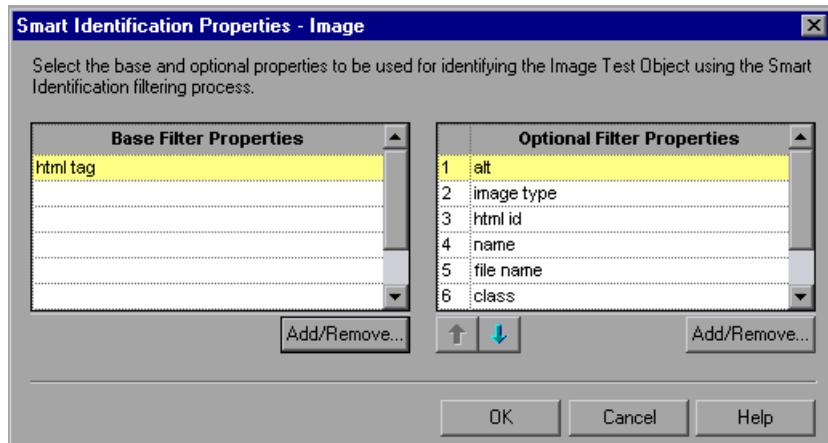


- 2 Select the appropriate environment in the **Environment** list. The test object classes associated with the selected environment are displayed in the **Test object classes** list.

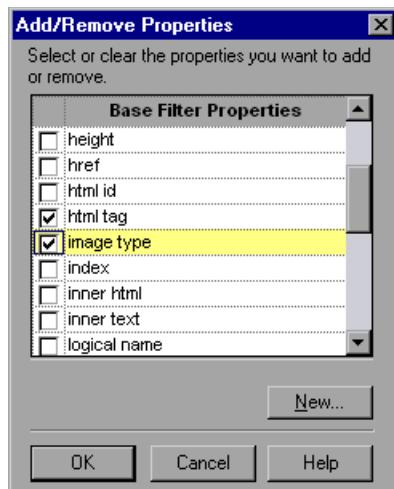
Note: The environments included in the Environment list are those that correspond to the loaded add-in environments. For more information on loading add-ins, see Chapter 29, "Working with QuickTest Add-Ins."

- 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.



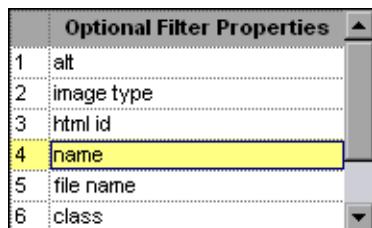
- 9 Select the properties you want to include in the **Optional Filter Properties** list and/or clear the properties you want to remove from the list.

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.



Optional Filter Properties	
1	alt
2	image type
3	html id
4	name
5	file name
6	class

-  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 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 recorded 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. You can configure the object identification settings for a user defined object class just as you would any other object class.

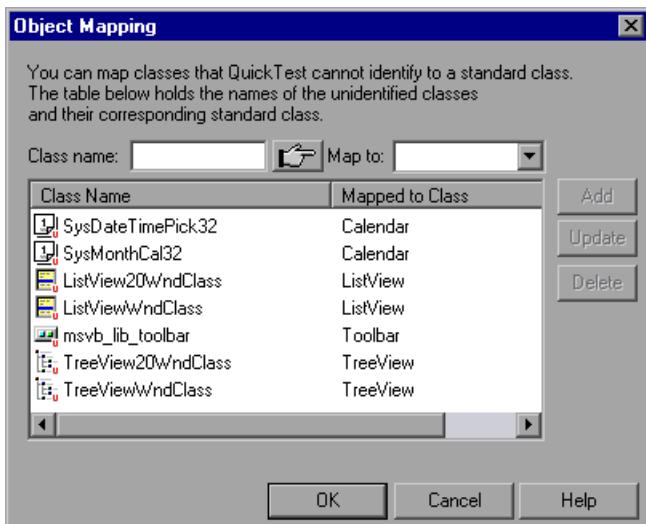
Note that an object that cannot be identified should be mapped only to a standard Windows class with comparable behavior. For example, do not map an object that behaves like a button to the edit class.

Note: You can define user-defined classes only when **Standard Windows** is selected in the **Environment** box.

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

- 1 Choose **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.

Tip: Hold the CTRL key to change the window focus or perform operations such as a right-click or mouseover to display a context menu. Note that pressing the CTRL key does not enable you to select an application from the Windows taskbar, therefore you must make sure that the window you want to access is not minimized.

- 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 4-5 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 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 Recording Properties,” on page 792, and “Configuring Smart Identification,” on page 803.

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.

Choosing the Object Repository Mode

When you create a test or component, all the information about the objects in your test or component is stored in an object repository. You can view and modify this information in the Object Repository dialog box.

There are two types of object repositories—*shared object repository* and *per-action object repository*.

-  Components use shared object repository files stored in Quality Center.
-  When working with tests, you can choose which type of object repository you want to use as the default type for all new tests, and you can change the object repository type as necessary for each new test.

This chapter describes:

- About Choosing the Object Repository Mode
- Deciding Which Object Repository Mode to Choose
- Setting the Object Repository Mode
- Choosing a Shared Object Repository File

About Choosing the Object Repository Mode

When QuickTest runs a test or component, it simulates a human user by moving the mouse cursor over the application, clicking objects, and entering keyboard input. Like a human user, QuickTest must learn the interface of an application to be able to work with it. QuickTest does this by learning the application's objects and their corresponding property values and storing these object descriptions in an object repository file.

 When you plan and create tests, you must consider how you want to store the objects in your test. You can have a separate object repository for each action and store the objects for each action in its corresponding object repository, or you can store all the objects in your test or component in a common (shared) object repository file that can be used among multiple tests.

QuickTest's TestGuard technology enables you to maintain the reusability of your tests or components by storing all the information regarding your test objects in the shared object repository. When objects in your application change, test object information can be updated in one central location for multiple tests or components.

 If you are new to using QuickTest, you may want to use the object repository per-action mode. This is the default setting, and all tests created in QuickTest 5.6 or earlier use this mode. In this mode, QuickTest automatically creates an object repository file for each action in your test so that you can record and run tests without creating, choosing, or modifying object repository files. If you do make changes to an action object repository, your changes do not have any effect on other actions or other tests (except tests that call the action; for more information, see "Inserting Calls to Existing Actions" on page 357).

If you are familiar with testing, it is probably most efficient to work in the shared object repository mode. In this mode, you can use one object repository file for multiple tests or components if the tests or components include the same objects. Object information that applies to many tests or components is kept in one central location. When the objects in your application change, you can update them in one location for multiple tests or components.

 You do not have to use the same object repository mode for all tests. You can select the per-action repository mode or a specific shared object repository file as the default for new tests in the Resources tab of the Test Settings dialog box. When you open a new test, the test is set to use the default repository. If you want to use a different shared object repository or set per-action mode for your new test, you can change the selection for that test in the Resources tab of the Test Settings dialog box before you begin adding objects to the test.

For example, suppose you set a shared object repository file as the default for all tests. If you open a new test and want to use the per-action repository mode for that particular test, select **Per-action** in the Test Settings dialog box before you begin adding objects to the new test.

When you open and work with an existing test or component, the test always uses the mode and file that are specified in the Resources tab of the Test Settings dialog box or the Business Component Settings dialog box.

Notes:

You can change the object repository type for the test only if the test does not include any steps or any objects and the test does not call any external actions. However, if you are using a shared object repository, you can modify the path of the shared object repository file and change the file the test or component is using. For more information, see “Choosing a Shared Object Repository File” on page 831.

There is no need to load object repository files from within your test or component.

 If you want to use a shared object repository from Quality Center, you must save the shared object repository as an attachment in your Quality Center project before you specify the object repository file in the Resources tab of the Test Settings dialog box. For more information, see “Using Shared Object Repository Files with Quality Center” on page 828.

Deciding Which Object Repository Mode to Choose

 To choose the default object repository mode and the appropriate object repository mode for each test, you need to understand the differences between the two modes.

In general, the object repository per-action mode is easiest to use when you are creating simple record and run 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 test object properties.
- You generally create single-action tests.

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

- 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 test object properties.
- You often work with multi-action tests and regularly use the **Insert Copy of Action** and **Insert Call to Action** options.

The sections below describe the effects and implications of using each mode in conjunction with various QuickTest features.

Understanding the Object Repository Per-Action Mode



When working in object repository per-action mode:

- QuickTest creates a new (blank) object repository for each action.
- As you record operations on objects in your application, QuickTest automatically stores the information about those objects in the appropriate action object repository.
- When you create a new action, QuickTest creates another new object repository and again begins adding test objects to the repository as you record.
- When you record on the same object in your application in two different actions, the object is stored as a separate test object in each object repository.
- When you save your test, all of the action object repositories are automatically saved with the test as part of each action within the test. The action object repository is not accessible as a separate file (as is the shared object repository).

Updating Test Objects in Object Repository Per-Action Mode

 Modifying the test object properties, values, or names in one object repository does not affect the information stored for the same test object in another object repository or in other tests.

If you parameterize objects in one action, the parameterization applies only to the objects in that current action. For more information about parameterizing tests, see Chapter 12, “Parameterizing Values.”

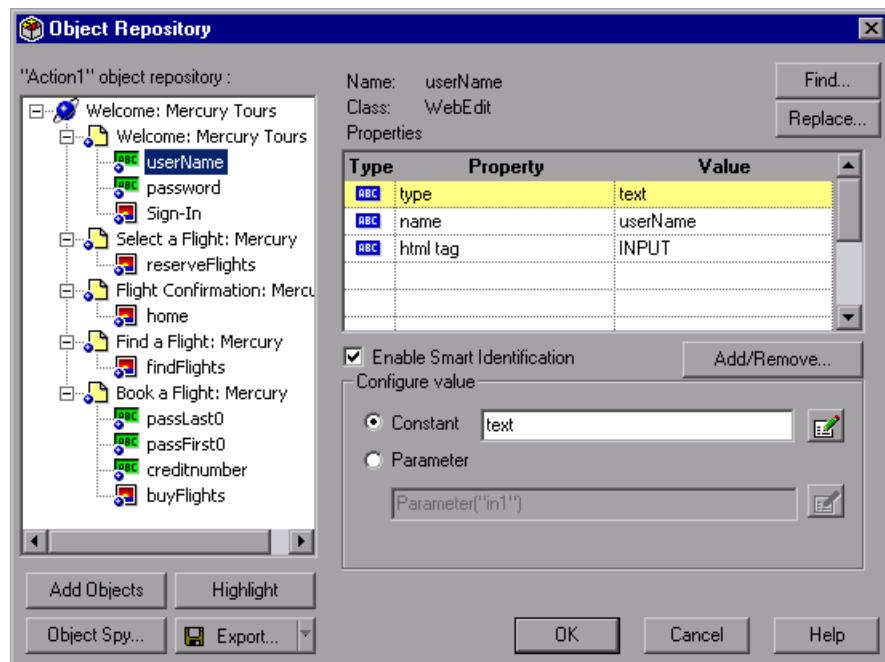
If you want to update object properties, values, or names for the same object in several actions, open the object repository for each action containing the object (by selecting a step in the action you want to modify and choosing **Tools > Object Repository** or clicking the **Object Repository** toolbar button) and make the required change in each object repository.



Using the Object Repository Dialog Box in Object Repository Per-Action Mode

 You can view and modify the information about the objects in an action in the Object Repository dialog box. You also can use the **Export** option to save your object repository as a shared object repository file to use with other tests.

When working in the object repository per-action mode, the Object Repository dialog box displays a hierarchy of all objects in the current action, grouped at each level by test object class.



The Object Repository dialog box contains a variety of options for managing the objects in your action. For more information, see Chapter 4, “Managing Test Objects.”

Splitting and Copying Actions in Object Repository Per-Action Mode

 When you split an action in your test while using the object repository per-action mode:

- QuickTest makes a copy of the action object repository.
- The two actions have identical object repositories containing all of the objects that were in the original object repository.
- If you add objects to one of the split actions, the new objects are added only to the corresponding action object repository.

For more information, see “Splitting Actions” on page 362.

When you insert a copy of an action into your test:

- The copied action’s action object repository is copied along with the action steps.
- You can edit the object repository of the copied action as you would any other object repository in the test.

Note: You cannot insert a copy of an action that uses a shared object repository file into a test that is using action object repositories. This is because a test using action object repositories has a separate object repository for each action. A copied action must use its own action object repository and not one that is shared throughout a test or several tests.

For more information about copying actions, see “Inserting Calls to Existing Actions” on page 352.

Inserting Action Calls in Object Repository Per-Action Mode

 When you insert a call to an action into a test using the object repository per-action mode:

- QuickTest refers to the called action’s object repository when running the test.
- The called action’s object repository is read-only, as are all the steps in the called action.

Note: You cannot insert a call to an action that uses a shared object repository file into a test using action object repositories. This is because a test using action object repositories has a separate object repository for each action and an action in a test using shared object repositories uses a repository that is shared throughout a test or several tests. You also cannot change a called action's repository type within a test that uses per-action object repositories.

For more information about calling actions, see “Inserting Calls to Existing Actions,” on page 357, and “Setting Action Properties,” on page 366.

Understanding the Shared Object Repository Mode

When you use the shared object repository mode, QuickTest uses the object repository file you specify for all the actions in your test, or for your component. When you create a new test or component, you can specify an existing object repository file, or you can create a new one, by specifying a new file name.

Once you have begun creating your test or component, you can specify a different object repository file or create a new one. Before running the test or component, you must ensure that the object repository file being used by the test or component contains all the objects in your test or component. Otherwise, the test or component may fail. For more information, see “Adding Objects to the Object Repository” on page 73.

Tip: If you do not specify a file extension when creating a new object repository file, then QuickTest automatically appends the default extension name for shared object repositories: .tsr

When working in shared object repository mode:

- QuickTest adds new objects to the object repository as you record steps on the object.
- QuickTest uses the existing information and does not add the object to the object repository if you record operations on an object that already exists in the shared object repository (i.e. the object has the same test object description).
- QuickTest uses the same test object from the shared object repository file when you record on the same object in two different actions or in different tests or components.
- QuickTest saves changes to the shared object repository file when you save your open test or component or when you click **Save** in the Object Repository dialog box if the shared object repository is not locked and/or opened in read-only mode.

Note: A shared object repository file may have been locked by QuickTest. If the file is locked, a message regarding locked resources opens. For more information, see “Creating, Opening, and Saving Tests with Locked Resources” on page 109.

Updating Test Objects in Shared Object Repository Mode

Shared object repository mode enables you to:

- easily maintain your tests or components when an object in your application changes
- manage all the objects for a suite of tests or components in one central location

When you modify the test object properties, names, parameterization settings, and so forth, in your shared object repository for an open test or component, QuickTest applies the change to all the tests or components that use the same shared object repository file.

Note: You save the changes you make to the shared object repository by saving the open test or by clicking **Save** in the Object Repository dialog box. You cannot save changes to the shared object repository if it has been locked and/or opened in read-only mode.

Parameterizing Test Objects in Shared Object Repository Mode

When you parameterize an object property in a shared object repository, there are several issues you should consider:

- When you parameterize an object property (with any type of parameter), the change applies to all tests or components that already contain that object.
- When you parameterize an object property that is part of an object's description, and then record on the object in your application again, QuickTest creates a new test object in the shared object repository. This is because the description of the object in the object repository is not identical to the one in the application (one or more of the properties in the test object description has a parameterized value rather than the constant value of the corresponding object property in the application).

Additionally, when parameterizing using a Data Table parameter:

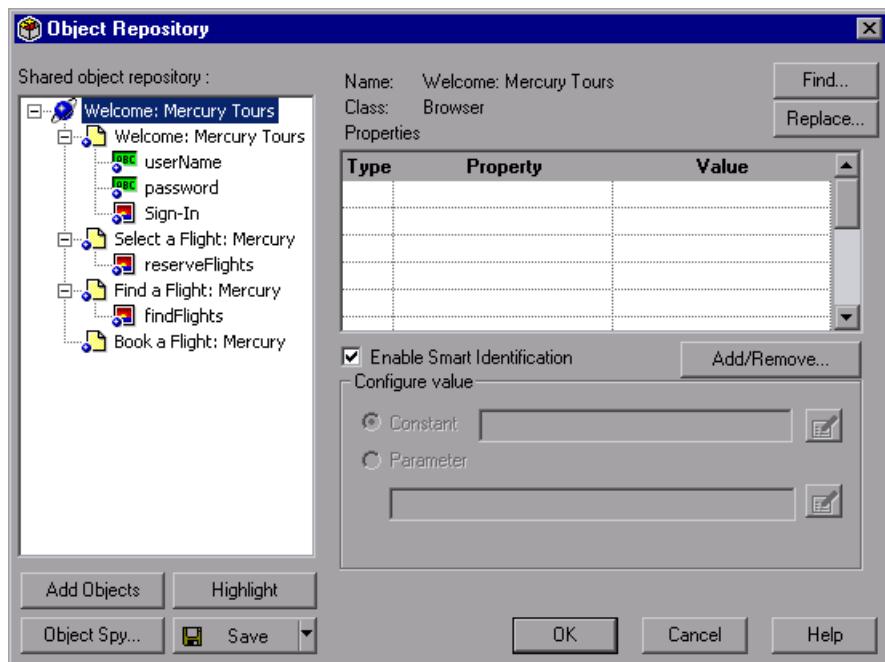
- You must use the global data sheet. The **Current action sheet (local)** option in the Data Table Parameter Options dialog box is disabled.
- Your change to the object is saved in the shared object repository, but the corresponding Data Table is saved only with the test or component in which you are currently working.
- You must ensure that every other test or component containing the parameterized object and using the same shared object repository also has a corresponding column in the global Data Table. Otherwise the test or component may fail.
- The column name in the global Data Table related to the parameterized object must be the same for every test or component using the same shared object repository.

- The data within the global Data Table column can be different for each test or component.

Using the Object Repository Dialog Box in Shared Object Repository Mode

You can view and modify the information about the objects in your test or component in the Object Repository dialog box.

When working in the shared object repository mode, the Object Repository dialog box displays a hierarchy of all objects in the object repository file. Objects in the Object Repository dialog box are grouped at each level by test object class.



The Object Repository dialog box contains a variety of options for managing the objects in your test or component. For more information, see Chapter 4, "Managing Test Objects."

Renaming Objects in a Shared Object Repository

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

Splitting and Copying Actions in Shared Object Repository Mode

 When using the shared object repository mode, splitting an action has no effect on the object repository file. For more information, see “Splitting Actions” on page 362.

When you insert a copy of an action into a test that uses a shared object repository, keep in mind:

- Only the action itself is copied and not its corresponding object repository.
- The copied action will use the same shared object repository as the test into which it is being copied.
- When you insert copies of actions from tests using a different shared object repository or per-action repository mode, you must ensure that all the objects contained in the action are in the test’s shared object repository. If the action uses objects that are not in the shared object repository, the test may fail. For more information, see “Adding Objects to the Object Repository” on page 73.

Inserting Action Calls in Shared Object Repository Mode

 When you insert a call to an external action while working in the shared object repository mode, you can call actions from tests that use:

- the same shared object repository as the calling test
- the object repository per-action mode
- a different shared object repository than the one used by the calling test

When you insert a call to an action whose test uses the object repository per-action mode, you can instruct the external action to refer to its own object repository or to use the same shared object repository as the rest of the current test. You do this in the External Action tab of the Action Properties dialog box. For more information, see “Setting Action Properties” on page 366.

Note: If the called action continues to use its own per-action object repository, that object repository is read-only in the test calling the action, as are all the called action’s steps.

When you insert a call to an external action using a different shared object repository, the called action uses the current test’s shared object repository. Before running the test, ensure that all the objects in the called action are in the current test’s shared object repository file. If the called action contains objects that are not in the current test’s shared object repository file, the test may fail. For information on adding objects to the object repository, see “Adding Objects to the Object Repository” on page 73.

Using Shared Object Repository Files with Quality Center

You can add a new or existing shared object repository to your Quality Center project. Keep in mind:

-  If you add an existing shared object repository file from the file system to a Quality Center project, it will be a copy of the one used by tests not in the Quality Center project.
- You must be connected to Quality Center to work with a shared object repository in the file system.
-  Once you save the file to the project, changes made to the Quality Center repository file will not affect the file in the file system and vice versa.
- You can use the **Save as** option in the Object Repository dialog box to save your shared object repository in Quality Center.

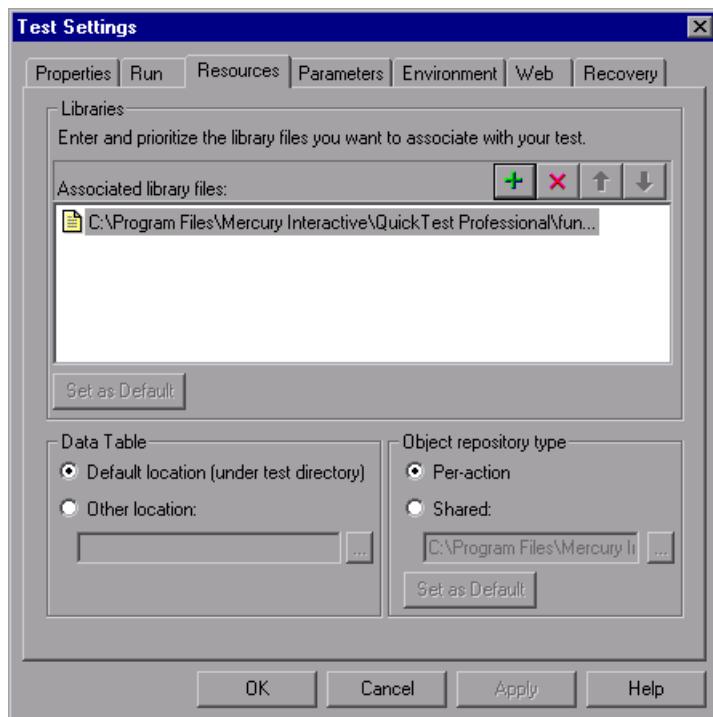
Setting the Object Repository Mode

 You set the object repository mode for your tests in the Resources tab of the Test Settings dialog box. You can change the object repository mode for a test if the test does not call any external actions and if the test does not contain any steps or any objects. You cannot change the object repository mode for a component.

If you do not change your object repository, QuickTest uses the object repository mode and/or file that was set as the default for new tests.

To set the object repository mode:

- 1 Choose **Test > Settings > Resources** tab.



- 2 In the **Object repository type** area, select **Per-action** or **Shared**.

- 3 If you selected **Shared** in step 2, specify the shared object repository file you want to use as the object repository file. To specify a file, enter the object repository file name or click the browse button and select a resource file from the Open dialog box. To create a new shared object repository file, enter a new file name in the **Shared** box.
-

Tip: The default file extension for object repository files is **.tsr**, but the file may have any extension. When browsing for an existing object repository file in the Open dialog box, select **All Files** in the **Files of type** box.

Notes:

You can enter an existing object repository file as a relative path. QuickTest will search for the file in the current test's directory, and then in the folders listed in the Folders tab of the Options dialog box. For more information, see “Setting Folder Testing Options” on page 616. When creating a new shared object repository file, you must enter the full path.

Make sure that you have write permissions for the object repository file you select. Otherwise, you will be able to save changes to your test, but new test objects or any changes you make to object properties or parameterization settings will not be saved in the object repository.

- 4 If you want to make your selection the default setting for new tests, click **Set as Default**.
- If **Per-action** is the default setting, all new tests will be created with a per-action object repository.
 - If you set **Shared** as the default setting for new tests, the shared object repository file listed will be the default shared object repository for all new tests. If this file is in Quality Center, you must be connected to Quality Center when you create a new test.

For guidelines and more information, see “Choosing a Shared Object Repository File,” below.

- 5 Click **OK** to save and close the Test Settings dialog box.
-

Note: Changes you make to the default object repository type apply only to tests opened after you click **OK** to close the Test Settings dialog box. If a new test is already open when you make changes to the default object repository type, the changes are not applied to that test. To use the new settings, create a new test by choosing **File > New**.

Choosing a Shared Object Repository File

When working with an existing test or component, the object repository mode and/or file that is associated with the test or component is displayed in the Resources tab of the Test Settings dialog box or the Business Component Settings dialog box. When you create a test, the default mode is displayed.

If the test does not call any external actions and the test or component does not contain any steps or any objects, you can change the repository mode or the shared object repository file being used for that test or component.

Once objects or steps have been added to a test, the object repository mode cannot be changed from per-action to shared or vice versa. If your existing test or component uses a shared object repository file, you can change the shared object repository file that the test or component uses.

Guidelines for Choosing a Shared Object Repository File

You can create a new shared object repository file or associate an existing shared object repository file with your test or component:

- when creating a new test or component
- when working with an existing test or component that is in shared object repository mode
- if your test does not call any external actions and if the test or component does not contain any steps or any objects

When creating a new shared object repository file or selecting an existing shared object repository file for your test or component, consider the following:

- You create a new shared object repository file by typing a new name in the Resources tab of the Test Settings dialog box or Business Component Settings dialog box, or in the Object Repository dialog box of an existing test or component after selecting the **Save As** option.
- You can select an existing shared object repository file by browsing to it and selecting it in the Resources tab of the Test Settings dialog box or Business Component Settings dialog box.
- You cannot create a new shared object repository file with a relative path. You must designate an absolute path for a new shared object repository file.
- When changing the shared object repository file your test or component uses, changes to the current shared object repository are not automatically saved. If you do not want to lose your changes to the current shared object repository, click **Save** in the Object Repository dialog box before designating a different shared object repository file.
-  If you select a file that has a relative path, QuickTest will search for the file in the current test's folder and then in the folders listed in the Folders tab of the Options dialog box. For more information, see Chapter 24, "Setting Folder Testing Options."
- When associating your existing test or component with a new or existing shared object repository file, you must ensure that all the objects contained in the component or test and all its actions are in the component or test's shared object repository before running the test or component. Otherwise, your test or component may fail. For more information, see "Adding Objects to the Object Repository" on page 73.
- An existing shared object repository file you associate with your test or component may be locked by QuickTest. When opening a test or component with a locked shared object repository, QuickTest enables you to open the test or component in read-only or read-write mode. However, the object repository is locked and any modifications to it cannot be saved. For more information, see "Creating, Opening, and Saving Tests with Locked Resources" on page 109.

Setting a Default Shared Object Repository File

 You set a default shared object repository file for tests by clicking **Set as Default** in the Resources tab of the Test Settings dialog box with the **Shared** option selected. The file listed will be the default shared object repository file for all new tests.

 You set a default shared object repository file for components in the component template. For more information, “Working with Component Templates” on page 987. The file listed will be the default shared object repository file for all new components.

Consider the following when setting a default shared object repository file:

- Any existing or open tests or components will not be affected by setting a different, default shared object repository file.
- All new tests or components will attempt to access your default shared object repository.
-  If the shared object repository file’s path is relative, QuickTest searches for the shared object repository file in the current test’s folder and then in the Folders tab of the Options dialog box.
- It is recommended that you ensure that the path for the shared object repository is absolute or listed in the Folders tab of the Options dialog box.
-  If you choose to set a file with a relative path or one that is in the current test’s folder as the default shared object repository, QuickTest may not be able to locate the file when creating a new test.
-  If the default shared object repository file is in Quality Center, you must be connected to Quality Center when you create a new test.

35

Configuring Web Event Recording

If QuickTest does not record Web events in a way that matches your needs, you can configure the events you want to record for each type of Web object.

This chapter describes:

- ▶ About Configuring Web Event Recording
- ▶ Selecting a Standard Event Recording Configuration
- ▶ Customizing the Event Recording Configuration
- ▶ Saving and Loading Custom Event Configuration Files
- ▶ Resetting Event Recording Configuration Settings

About Configuring Web Event Recording

QuickTest creates your or component by recording the *events* you perform on your Web-based application. An event is a notification that occurs in response to an operation, such as a change in state, or as a result of the user clicking the mouse or pressing a key while viewing the document. You may find that you need to record more or fewer events than QuickTest automatically records by default. You can modify the default event recording settings by using the Web Event Recording Configuration dialog box to select one of three standard configurations, or you can customize the individual event recording configuration settings to meet your specific needs.

For example, QuickTest does not generally record mouseover events on link objects. If, however, you have a mouseover *behavior* connected to a link, it may be important for you to record the mouseover event. In this case, you could customize the configuration to record mouseover events on link objects whenever they are connected to a behavior.

Notes: Event configuration is a global setting and therefore affects all tests or components that are recorded after you change the settings.

Changing the event configuration settings does not affect tests or components that have already been recorded. If you find that QuickTest recorded more or less than you need, change the event recording configuration and then re-record the part of your test or component that is affected by the change.

Changes to the custom Web event recording configuration settings do not take effect on open browsers. To apply your changes for an existing test or component, make the changes you need in the Web Event Recording Configuration dialog box, refresh any open browsers, and then start a new recording session.

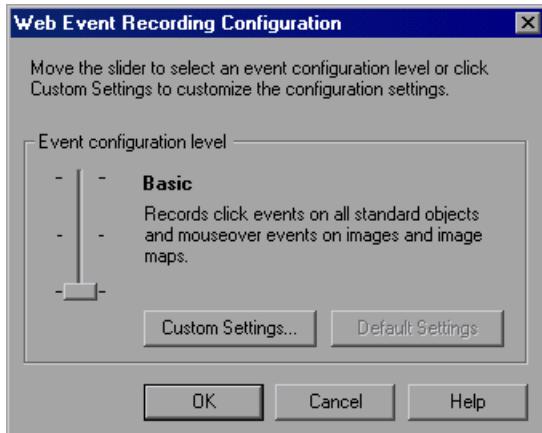
Selecting a Standard Event Recording Configuration

The Web Event Recording Configuration dialog box offers three standard event-configuration levels. By default, QuickTest uses the *Basic* recording-configuration level. If QuickTest does not record all the events you need, you may require a higher event-configuration level.

Level	Description
Basic	Default <ul style="list-style-type: none">• Always records click events on standard Web objects such as images, buttons, and radio buttons.• Always records the submit event within forms.• Records click events on other objects with a <i>handler</i> or <i>behavior</i> connected. For more information on handlers and behaviors, see “Listening Criteria” on page 845.• Records the mouseover event on images and image maps only if the event following the mouseover is performed on the same object.
Medium	Records click events on the <DIV>, , and <TD> HTML tag objects, in addition to the objects recorded in the basic level.
High	Records mouseover, mousedown, and double-click events on objects with <i>handlers</i> or <i>behaviors</i> attached, in addition to the objects recorded in the basic level. For more information on handlers and behaviors, see “Listening Criteria” on page 845.

To set a standard event-recording configuration:

- 1 Choose **Tools > Web Event Recording Configuration**. The Web Event Recording Configuration dialog box opens.



- 2 Use the slider to select your preferred standard event recording configuration.

Tip: You can click the **Custom Settings** button to open the Custom Web Event Recording dialog box where you can customize the event recording configuration. For more information, see “Customizing the Event Recording Configuration,” below.

You can click the **Default Settings** button to return the scale to the **Basic** level.

- 3 Click **OK**.

Customizing the Event Recording Configuration

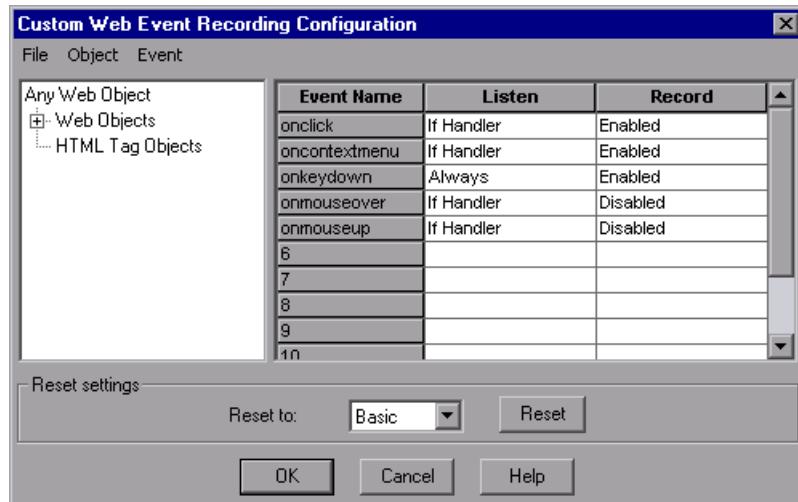
If the standard event configuration levels do not exactly match your recording needs, you can customize the event recording configuration using the Custom Web Event Recording Configuration dialog box.

The Custom Web Event Recording Configuration dialog box enables you to customize event recording in several ways. You can:

- add or delete objects to which QuickTest should apply special listening or recording settings
- add or delete events for which QuickTest should listen
- modify the listening or recording settings for an event

To customize the event recording configuration:

- 1 Choose **Tools > Web Event Recording Configuration**. The Web Event Recording Configuration dialog box opens.
- 2 Click the **Custom Settings** button. The Custom Web Event Recording Configuration dialog box opens.

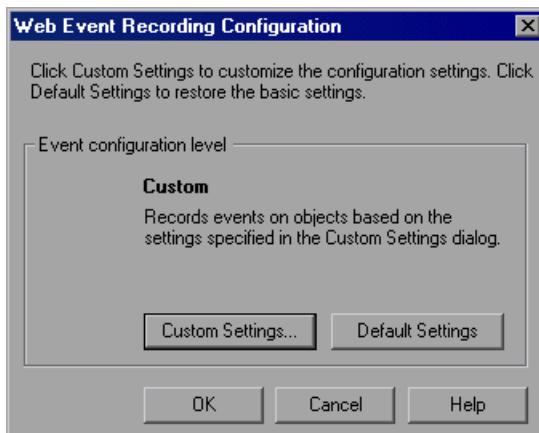


3 Customize the event recording configuration using the following options:

Option	Description
Objects pane	<p>Displays a list of Web test object classes and HTML tag objects.</p> <ul style="list-style-type: none"> • To add an object, choose Object > Add. • Only HTML Tag objects can be deleted. To delete an HTML object from the list, choose Object > Delete. <p>For more information, see “Adding and Deleting Objects in the Custom Configuration Object List” on page 841.</p>
Events pane	<p>Displays a list of events associated with the object.</p> <ul style="list-style-type: none"> • To add an event to the Events pane, choose Event > Add. • To delete an event, choose Event > Delete. <p>For more information, see “Adding and Deleting Listening Events for an Object” on page 843.</p>
Event Name	The name of the event.
Listen	<p>The criteria for when QuickTest listens to the event.</p> <ul style="list-style-type: none"> • Always—Always listens to the event. • If Handler—Listens to the event if a handler is attached to it. A <i>handler</i> is code in a Web page, typically a function or routine written in a scripting language, that receives control when the corresponding event occurs. • If Behavior—Listens to the event if a DHTML behavior is attached to it. A DHTML <i>behavior</i> is a simple, lightweight component that encapsulates specific functionality or behavior on a page. When applied to a standard HTML element on a page, a behavior enhances that element’s default behavior. • If Handler or Behavior—Listens to the event if a handler or behavior is attached to it. • Never—Never listens to the event. <p>For more information, see “Modifying the Listening and Recording Settings for an Event” on page 845.</p>

Option	Description
Record	Enables or disables recording of the event for the selected object, or enables recording of the event only if the subsequent event occurs on the same object.
Reset	Enables you to reset your settings to a preconfigured level.

- 4 Click **OK**. The Custom Web Event Recording Configuration dialog box closes. The slider scale on the Web Event Recording Configuration dialog box is hidden and the configuration description displays **Custom**.



Adding and Deleting Objects in the Custom Configuration Object List

The Custom Web Event Recording Configuration dialog box lists objects in an object hierarchy. The top of the hierarchy is **Any Web Object**. The settings for Any Web Object apply to any object on the Web page being tested, for which there is no specific event recording configuration set. Below this are the **Web Objects** and **HTML Tag Objects** categories, each of which contains a list of objects.

When working with the objects in the Custom Web Event Recording Configuration dialog box, keep the following principles in mind:

- If an object is listed in the Custom Web Event Recording Configuration dialog box, then the settings for that object override the settings for Any Web Object.
- You cannot delete or add to the list of objects in the **Web Objects** category, but you can modify the settings for any of these objects.
- You can add any HTML Tag object in your Web page to the HTML Tag Objects category.

To add objects to the event configuration object list:

- 1 In the Custom Web Event Recording Configuration dialog box, choose **Object > Add**. A **New Object** object is displayed in the HTML Tag Objects list.



- 2 Click **New Object** to rename it. Enter the exact HTML Tag name.

By default the new object is set to listen and record *onclick* events with handlers attached.

For more information on adding or deleting events, see “Adding and Deleting Listening Events for an Object,” below. For more information on listening and recording settings, see “Modifying the Listening and Recording Settings for an Event” on page 845.

To delete objects from the HTML Tag Objects list:

- 1** From the Custom Web Event Recording Configuration dialog box, select the object in the HTML Tag Objects category that you want to delete.
- 2** Choose **Object > Delete**. The object is deleted from the list.

Note: You cannot delete objects from the **Web Objects** category.

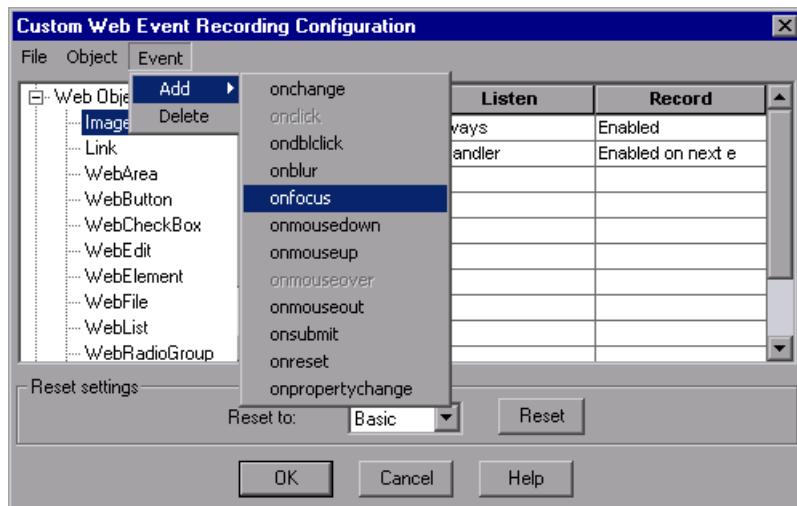
Adding and Deleting Listening Events for an Object

You can modify the list of events that trigger QuickTest to listen to an object.

To add listening events for an object:

- 1** In the Custom Web Event Recording Configuration dialog box, select the object to which you want to add an event, or select **Any Web Object**.

- 2 Choose **Event > Add**. A list of available events opens.



- 3 Select the event you want to add. The event is displayed in the Event Name column in alphabetical order. By default, QuickTest listens to the event when a handler is attached and always records the event (as long as it is listened to at some level).

For more information on listening and recording settings, see “Modifying the Listening and Recording Settings for an Event,” below.

To delete listening events for an object:

- 1 In the Custom Web Event Recording Configuration dialog box, select the object from which you want to delete an event, or select **Any Web Object**.
- 2 Select the event you want to delete from the **Event Name** column.
- 3 Choose **Event > Delete**. The event is deleted from the **Event Name** column.

Modifying the Listening and Recording Settings for an Event

You can select the listening criteria and set the recording status for each event listed for each object.

Note: The listen and record settings are mutually independent. This means that you can choose to listen to an event for particular object, but not record it, or you can choose not to listen to an event for an object, but still record the event. For more information, see “Tips for Working with Event Listening and Recording” on page 847.

Listening Criteria

For each event, you can instruct QuickTest to listen every time the event occurs on the object if an event handler is attached to the event, if a DHTML behavior is attached to the event, if an event handler or DHTML behavior are attached to the event, or to never listen to the event.

An event *handler* is code in a Web page, typically a function or routine written in a scripting language, that receives control when the corresponding event occurs.

A DHTML *behavior* is a simple, lightweight component that encapsulates specific functionality or behavior on a page. When applied to a standard HTML element on a page, a behavior enhances that element's default behavior.

To specify the listening criterion for an event:

- 1 From the Custom Web Event Recording Configuration dialog box, select the object for which you want to modify the listening criterion or select **Any Web Object**.

- 2** In the row of the event you want to modify, select the listening criterion you want from the **Listen** column.

Event Name	Listen	Record
onclick	If Handler	Enabled
onkeydown	Always	Enabled
onmouseover	If Handler	Disabled
4	Always	
5	If Handler	
6	If Behavior	
7	If Handler or Behavior	
8	Never	
9		
10		

You can select **Always**, **If Handler**, **If Behavior**, **If Handler or Behavior**, or **Never**.

Recording Status

For each event, you can enable recording, disable recording, or enable recording only if the next event is dependent on the selected event.

- **Enabled**—Records the event each time it occurs on the object as long as QuickTest listens to the event on the selected object, or on another object to which the event bubbles.
Bubbling is the process whereby, when an event occurs on a child object, the event can travel up the chain of hierarchy within the HTML code until it encounters an event handler to process the event.
- **Disabled**—Does not record the specified event and ignores event *bubbling* where applicable.
- **Enabled on next event**—Same as **Enabled**, except that it records the event only if the subsequent event occurs on the same object. For example, suppose a mouseover behavior modifies an image link. You may not want to record the mouseover event each time you happen to move the mouse over this image. Because only the image that is displayed after the mouseover event enables the link event, however, it is essential that the mouseover event is recorded before a click event on the same object. This option applies only to the Image and WebArea objects.

To set the recording status for an event:

- 1** From the Custom Web Event Recording Configuration dialog box, select the object for which you want to modify the recording status or select **Any Web Object**.
- 2** In the row of the event you want to modify, select a recording status from the Record column.

Event Name	Listen	Record
onclick	Always	Enabled
onmouseover	If Handler	Enabled on next ev▼
3		Disabled
4		Enabled
5		Enabled on next even▼
6		
7		
8		
9		
10		

Tips for Working with Event Listening and Recording

It can sometimes be difficult to find the ideal listen and recording settings. When defining these settings, keep in mind the following guidelines:

- If settings for different objects in the Objects Pane conflict, QuickTest gives first priority to settings for specific **HTML Tag Objects** and second priority to **Web Objects** settings. QuickTest only applies the settings for **Any Web Object** to Web objects that were not defined in the **HTML Tag Object** or **Web Objects** areas.
- To record an event on an object, you must instruct QuickTest to listen for the event, and to record the event when it occurs. You can listen for an event on a child object, even if a parent object contains the handler or behavior, or you can listen for an event on a parent object, even if the child object contains the handler or behavior. However, you must enable recording for the event on the *source object* (the one on which the event actually occurs, regardless of which parent object contains the handler or behavior).

For example, suppose a table cell with an *onmouseover* event handler contains two images. When a user touches either of the images with the mouse pointer, the event also bubbles up to the cell, and the bubbling includes information on which image was actually touched. You can record this mouseover event by:

- Setting **Listen** on the <TD> tag mouseover event to **If Handler** (so that QuickTest “hears” the event when it occurs), while disabling recording on it, and then setting **Listen** on the tag mouseover event to **Never**, while setting **Record** on the tag to **Enable** (to record the mouseover event on the image after it is listened to at the <TD> level).
- Setting **Listen** on the tag mouseover event to **Always** (to listen for the mouseover event even though the image tag does not contain a behavior or handler), and setting **Record** on the tag to **Enabled** (to record the mouseover event on the image).
- Instructing QuickTest to listen for many events on many objects may lower performance, so try to limit listening settings to the required objects.
- In rare situations, listening to the object on which the event occurs (the source object) can interfere with the event.

If you find that your application works properly until you begin recording on the application using QuickTest, your listen settings may be interfering.

If this problem occurs with a mouse event, try selecting the appropriate **Use standard Windows mouse events** option(s) in the Advanced Web Options dialog box. For more information, see “Advanced Web Options” on page 644.

If this problem occurs with a keyboard or internal event, or the **Use standard Windows mouse events** option does not solve your problem, set the listen settings for the event to **Never** on the source object (but keep the record setting enabled on the source object), and set the listen settings to **Always** for a parent object.

Saving and Loading Custom Event Configuration Files

You can save the changes you make in the Custom Web Event Recording Configuration dialog box, and load them at any time.

To save a custom configuration:

- 1** Customize the event recording configuration as desired. For more information on how to customize the configuration, see “Customizing the Event Recording Configuration” on page 839.
- 2** In the Custom Web Event Recording Configuration dialog box, Choose **File > Save Configuration As**. The Save As dialog box opens.
- 3** Navigate to the folder in which you want to save your event configuration file and enter a configuration file name. The extension for configuration files is **.xml**.
- 4** Click **Save** to save the file and close the dialog box.

To load a custom configuration:

- 1** Choose **Tools > Web Event Recording Configuration** and then click **Custom Settings** to open the Custom Web Event Recording Configuration dialog box.
- 2** Choose **File > Load Configuration**. The Open dialog box opens.
- 3** Locate the event configuration file (**.xml**) that you want to load and click **Open**. The dialog box closes and the selected configuration is loaded.

Resetting Event Recording Configuration Settings

You can restore standard settings after you set Custom settings by resetting the event recording configuration settings to the basic level from the Web Event Recording Configuration dialog box.

Note: When you choose to reset standard settings, your custom settings are cleared completely. If you do not want to lose your changes, be sure to save your settings in an event configuration file. For more information, see “Saving and Loading Custom Event Configuration Files” on page 849.

To reset basic level configuration settings from the Web Event Recording Configuration dialog box:

- 1 Choose **Tools > Web Event Recording Configuration**. The Web Event Recording Configuration dialog box opens.
- 2 Click **Default**. The standard configuration slider is displayed again and all event settings are restored to the **Basic** event recording configuration level.
- 3 If you want to select a different standard configuration level, see “Selecting a Standard Event Recording Configuration” on page 837.

You can also restore the settings to a specific (base) custom configuration from within the Custom Web Event Recording Configuration dialog box so that you can begin customizing from that point.

To reset the settings to a custom level from the Custom Web Event Recording Configuration dialog box:

- 1 Choose **Tools > Web Event Recording Configuration**. The Web Event Recording Configuration dialog box opens.
- 2 Click the **Custom Settings** button. The Custom Web Event Recording Configuration dialog box opens.
- 3 In the **Reset to** box, select the standard event recording level you want.
- 4 Click **Reset**. All event settings are restored to the defaults for the level you selected.

36

Working with the Expert View

In QuickTest, tests and components are composed of statements coded in Microsoft's VBScript programming language. The Expert View provides an alternative to the Keyword View for testers who are familiar with 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 components using a few simple programming techniques.

This chapter describes:

- About Working with the Expert View
- Understanding and Using the Expert View
- Navigating in the Expert View
- Understanding Basic VBScript Syntax
- Using Programmatic Descriptions
- Running and Closing Applications Programmatically
- Using Comments, Control-Flow, and Other VBScript Statements
- Retrieving and Setting Test Object Property Values
- Accessing Run-Time Object Properties and Methods
- Running DOS Commands
- Enhancing Your Tests using the Windows API
- Choosing Which Steps to Report During the Run Session

About Working with the Expert View

In the Expert View, you can view an action or component in VBScript. If you are familiar with VBScript, you can add and update statements and enhance your tests and components with programming.

When you record steps, the Expert View displays the steps as VBScript statements. After you record your test or component, you can increase its power and flexibility by adding recordable and non-recordable VBScript statements.

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 a method.

In the Expert View you can add steps to your test or component either manually or using the Step Generator. For more information on using the Step Generator, see “Inserting Steps Using the Step Generator” on page 467.

You can print the test or component displayed in the Expert View 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 or Component” on page 99.

Note: QuickTest Professional is Unicode compliant according to the requirements of the Unicode standard, enabling you to add and update VBScript statements for testing applications developed in many international languages. Unicode represents the required characters using 8-bit or 16-bit code values. For more information on the Unicode standard, refer to: <http://www.unicode.org/standard/standard.html>.

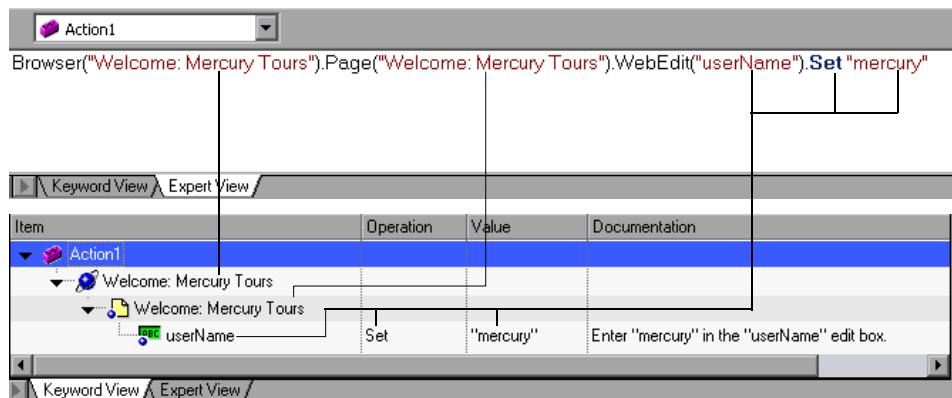
Understanding and Using the Expert View

If you prefer to work with VBScript statements, you can choose to work with your tests and components 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.

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

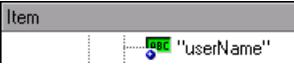
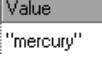
- In the Keyword View, QuickTest displays information about each step and shows the object hierarchy in an icon-based table. For more information, see Chapter 15, “Working with the Keyword View.”
- In the Expert View, QuickTest displays each step as a VBScript line. In object-based steps, the VBScript line 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 or component. 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 method 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 method 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 line of script, 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"
```

When you record your test, QuickTest records the operations you perform on your application in terms of the objects in it.

The objects in QuickTest are divided by environment. QuickTest environments include standard Windows objects, Visual Basic objects, ActiveX objects, and Web objects, as well as objects from other environments available as external add-ins.

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

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

For more information about adding steps that use methods to perform operations, see “Generating Statements in the Expert View” on page 858.

For more information about using VBScript, see “Understanding Basic VBScript Syntax” on page 873.

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
File "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 or component.

For more information on inserting and modifying checkpoints, see Chapter 6, “Understanding Checkpoints.” For more information on inserting and modifying output values, see Chapter 13, “Outputting Values.”

Understanding Parameter Indications

You can use QuickTest to enhance your tests and components 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
<i>Object_Hierarchy</i>	The hierarchical definition of the test object, consisting of one or more objects separated by a dot.
<i>Method</i>	The name of the method that QuickTest executes on the parameterized object.
<i>DataTable</i>	The reserved object representing the Data Table.
<i>parameterID</i>	The name of the column in the Data Table from which to take the value.
<i>sheetID</i>	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 recording a test on the Mercury Tours site, and you select San Fransisco as your destination. The following statement is 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 is 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 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 12, “Parameterizing Values.”

Generating Statements in the Expert View

The Expert View helps you to generate statements in three 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 467.
- You can manually generate VBScript statements that use methods to perform operations. QuickTest includes a statement completion (Intellisense) feature that helps you select the test object or method for your statement and to view the relevant syntax as you type in the Expert View. For more information, see “Generating a Statement for an Object”, below.
- When you start to type a VBScript keyword in the Expert View, QuickTest automatically adds the relevant syntax or blocks to your script, if the **Auto-expand VBScript syntax** option is enabled. For more information, see “Automatically Completing VBScript Syntax” on page 862.

Generating a Statement for an Object

When you type in the Expert View, QuickTest's statement completion (Intellisense) feature lets you select the test object or method 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 additional information, see Chapter 27, "Customizing the Expert View."

When the **Statement Completion** option is enabled:

- When you type a dot after a test object in a statement, QuickTest displays a list of available test objects and methods that you can add after the object you typed.
- When you type an object followed by an open parenthesis (, for example, Browser(, QuickTest displays a list of all test objects of this type in the object repository. If there is only one object of this type in the object repository, QuickTest automatically enters its name in quotes after the open parenthesis.
- When you type a method, QuickTest displays the syntax for the method, including the mandatory and optional arguments for the specific method. When you add a step that uses a method, you must define a value for each mandatory argument associated with the method.
- When you use the **Object** property in your statement, if the object data is currently available in the Active screen or the open application, QuickTest displays native methods and properties of any run-time object in your application. For more information on the **Object** property, see "Accessing Run-Time Object Properties and Methods" on page 899.

To generate a statement using statement completion in the Expert View:

- 1 Confirm that the **Statement Completion** option is selected (**Tools > Editor Options > General** tab).
- 2 In the Expert View, type an object followed by an open parenthesis (.

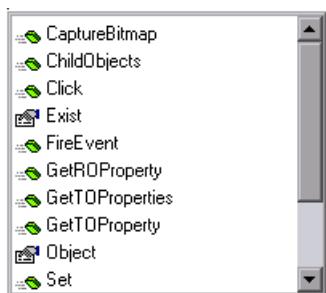


If there is only one object of this type in the object repository, QuickTest automatically enters its name in quotes after the open parenthesis. If more than one object of this type exists in the object repository, QuickTest displays them in a list.

- 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 line of script.
- 4 Type a period (.) after the object on which you want to perform the method.



Alternatively, you can press CTRL+SPACE or choose **Edit > Complete Word** after a period, or after you have begun to type a method name. QuickTest displays a list of the available methods for the object.



- 5 Double-click a method in the list or use the arrow keys to choose a method and press ENTER. QuickTest inserts the method into the line of script.

If you begin typing a method name before opening the list, and only one method matches the text you typed, the word is completed automatically. If more than one word matches the text you typed, the list of available methods is displayed, and the first method (alphabetically) that matches the text you typed is highlighted.

If the method contains arguments, QuickTest displays the syntax of the method in a tooltip.



In the above example, the **Select** method has one argument, called **Item**. The argument name represents the item to select in the list.

You can also place the cursor on any object or method and press CTRL+SHIFT+SPACE or choose **Edit > Argument Info** to display the statement completion tooltip for that item.

- 6 Enter the method arguments after the method as per 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 when you select the Keyword View tab. For more information, see “Handling VBScript Syntax Errors” on page 877.

For more details and examples of any QuickTest method, refer to the *QuickTest Professional Object Model Reference*.

For more information about VBScript syntax, see “Understanding Basic VBScript Syntax” on page 873.

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, 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 a line in the Expert View, 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 Expert View Behavior” on page 714.

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 the required keyword and QuickTest automatically enters the relevant VBScript block in the script. For more information, see “Selecting a VBScript Keyword”, below.

For more information on VBScript syntax, see “Understanding Basic VBScript Syntax” on page 873.

Selecting a VBScript Keyword

When you enter the first two characters of a keyword and then a space at the beginning of a line in the Expert View, and the initial characters exist in multiple VBScript keywords, the Select a Keyword dialog box opens. 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 example, if you enter the letters `pr` and then enter a space, the Select a Keyword dialog box opens containing the `private` and `property` keywords.



This dialog box is displayed only when the **Auto-expand VBScript syntax** option is selected in the General tab of the Editor Options dialog box.

Navigating in the Expert View

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

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.

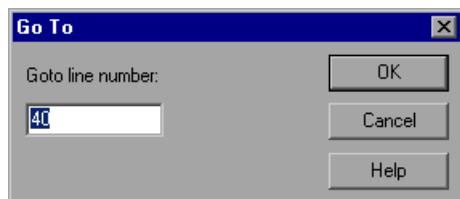
Using the Go To Dialog Box

You can use the Go To dialog box to navigate in the Expert View to a specified line in the action or component.

Tip: You can configure the Expert View to display line numbers, by selecting the **Show line numbers** option in the **Tools > Editor Options > General** tab. For more information on the Editor options, see Chapter 27, “Customizing the Expert View.”

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

- 1 Click the Expert View tab.
- 2 Choose **Edit > Go To**. The Go To dialog box opens.



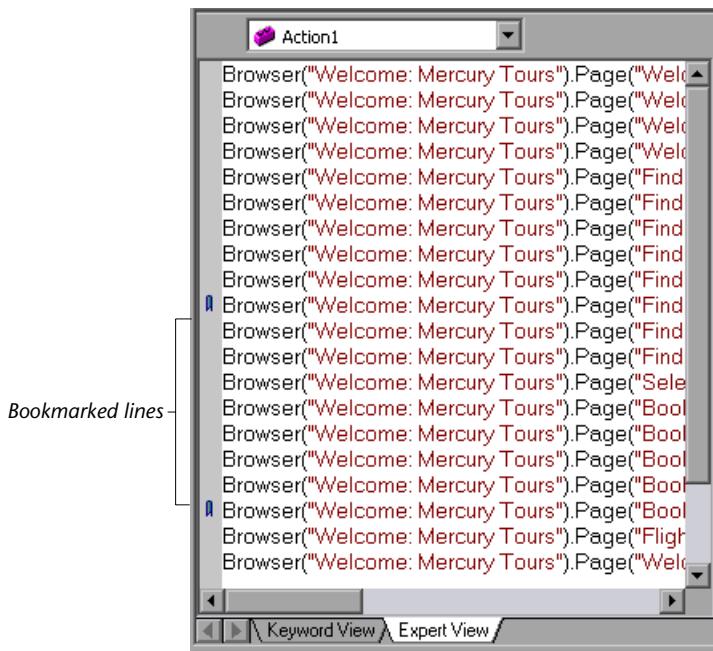
- 3 Enter the line of script to which you want to navigate in the **Goto line number** box. The cursor moves to the script line you specify.

Working with Bookmarks

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

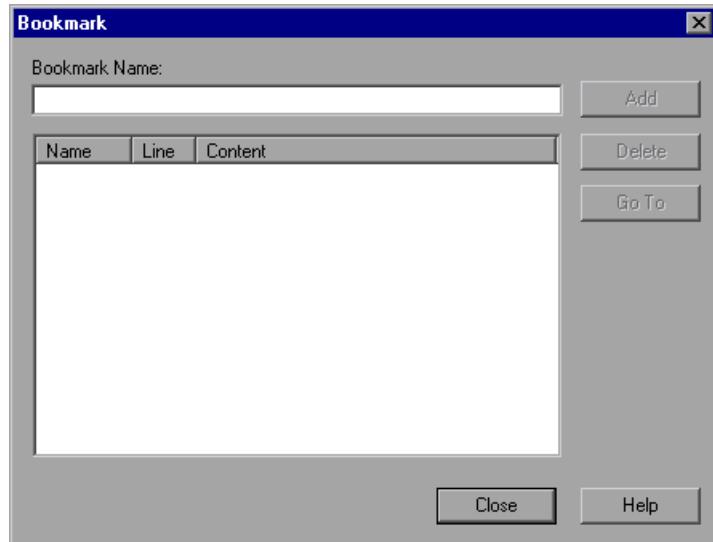
When you assign a bookmark, an icon is added to the left of the selected line in the Expert View. You can then use the **Go To** button in the Bookmarks dialog box to jump to the bookmarked rows.

In the following example, two bookmarks have been added to an action.



To set bookmarks:

- 1 Click the **Expert View** tab.
- 2 Click in the line in your action or component where you want to assign a bookmark.
- 3 Choose **Edit > Bookmarks**. The Bookmark 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 Bookmark 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.
- 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.
- 2** Choose **Edit > Bookmarks**. The Bookmark 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 component.

Tip: You can configure the Expert View to display line numbers, by selecting the **Show line numbers** option in the **Tools > Editor Options > General** tab. For more information on the Editor options, see Chapter 27, “Customizing the Expert View.”

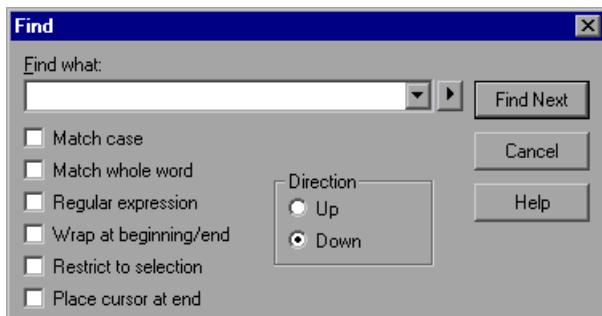
Finding Text Strings

You can specify text strings to locate in the current action in the Expert View. You can also search for strings in the Edit HTML Source and Edit HTML Tags dialog boxes, 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.

To find a text string:

- 1** Click the **Expert View** tab and choose **Edit > Find**,
or
click the **Expert View** tab and choose **Edit > Apply “With” to Script**, and then press **CTRL+F**,
or
in the Page Checkpoint Properties dialog box, click **Edit HTML Source** or **Edit HTML Tags**, and then right-click and choose **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 871.
- 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 exactly matches the text you entered in the **Find what** box.
 - **Match whole word**—Searches for occurrences that are whole words only and not part of larger 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, component, or dialog box 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, component, or dialog box 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 Expert View: **Up** or **Down**.
- 6 Click **Find Next** to highlight the next occurrence of the specified string in the current action or component in the Expert View or in the current dialog box.

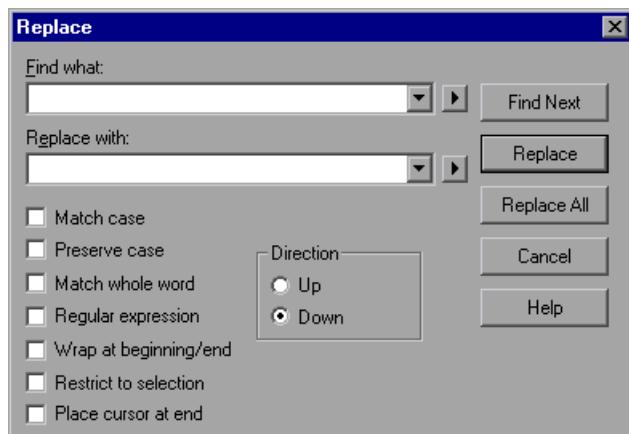
Replacing Text Strings

You can specify text strings to locate in the current action or component in the Expert View, 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 Click the **Expert View** tab and choose **Edit > Replace**, or in the Page Checkpoint Properties dialog box, click **Edit HTML Source** or **Edit HTML Tags**, and then right-click and choose **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.

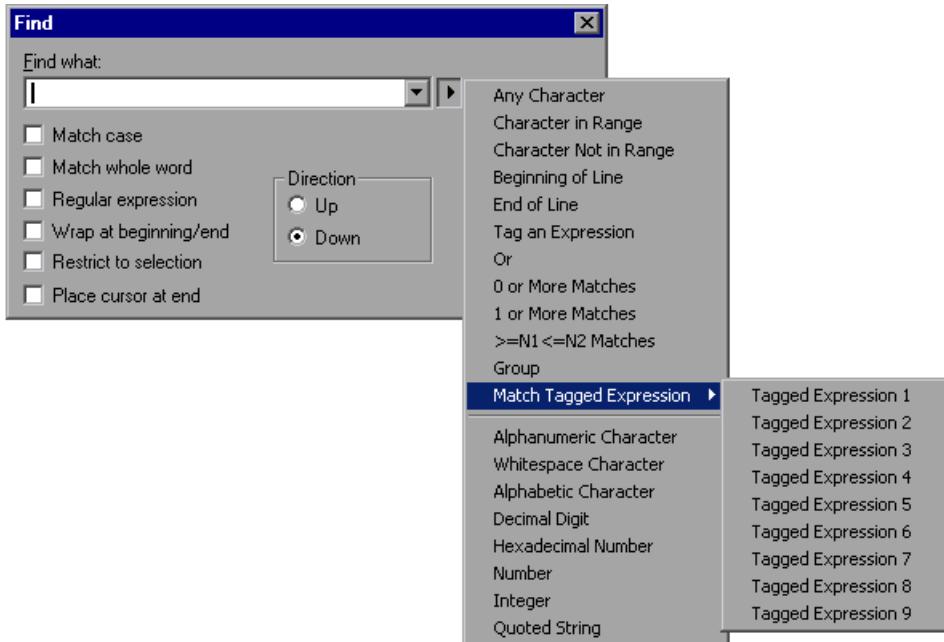
- 3 In the **Replace with** box, enter the text string you want to use to replace the found text.
- 4 If you want to use regular expressions in the **Find what** or **Replace with** string, click the arrow button  and select a regular expression. When you select a regular expression from the list, it 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 871.
- 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 larger 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, component, or dialog box 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, component, or dialog box text.
 - **Place cursor at end**—Places the cursor at the end of the highlighted occurrence when the search string is located.

- 6 Click **Find Next** to highlight the next occurrence of the specified text string in the current action or component in the Expert View.
- 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, component, or dialog box.

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 290. Note, however, 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, and 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** and 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(\:\n)**. 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 component. For more detailed information about 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. You cannot switch to the Keyword View without fixing or eliminating the syntax errors. For more information, see “Handling VBScript Syntax Errors” on page 877.

When working in the Expert View, you should bear in mind 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 method 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,” below.
- **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 875.
- **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 “Inserting Comments” on page 889.
- **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 and components, see “Using Comments, Control-Flow, and Other VBScript Statements” on page 888.

Using Variables

You can specify variables to store test objects or simple values in your test or component. 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 component. 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.

```
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. 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, 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, since 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, since **Call** is being used.

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

The following example requires parentheses around the **WaitProperty** method arguments, since 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, since it does not return a value.

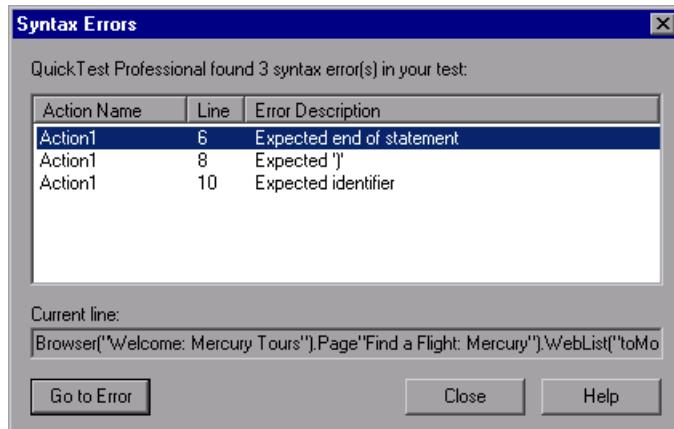
```
Browser("Mercury Tours").Page("Method of Payment").WebTable("FirstName").  
Click 3,4
```

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, QuickTest is unable to display the step in the Keyword View.

Tip: Microsoft's 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 Syntax Errors dialog box lists the syntax errors found in your test or component and enables you to locate each error so that you can correct it.



The summary at the top of the Syntax Errors dialog box shows the total number of syntax errors found in statements in your test or component.

The error list pane shows the details of each error:

Pane Element	Description
Action Name	The action containing the problematic step.
Line	The line containing the error. Lines are numbered from the beginning of the action or component.
Error Description	<p>The description of the 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'.</p> <p>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.</p>

The **Current line** box displays the syntax of the currently selected line.

Go to Error—Jumps to the selected line in the Expert View, positions the cursor at the point in the line where the error occurs, and closes the Syntax Errors dialog box. Alternatively, you can double-click the selected line in the error list pane.

Using Programmatic Descriptions

When you record an operation on an object, QuickTest adds the appropriate test object to the object repository. Once the object exists in the object repository, you can add statements in the Expert View to perform additional methods 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 method.

For example, in the statement below, `username` is the name of an edit field. The edit field is located on a page with the name `Mercury Tours` and the page was recorded 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 test run, 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 Web site or application.

You can also instruct QuickTest to perform methods 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 a method.

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 in order to perform the same operation on several objects with certain identical properties, or in order to perform an operation on an object whose properties match a description that you determine dynamically during the run session.

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. You can either list the set of properties and values that describe the object directly in a test statement, or you can add a collection of properties and values to a `Description` object, and then enter the `Description` object name in the statement.

Entering programmatic descriptions directly into your statements may be the easier method for basic object-description needs. However, in most cases, the Description object method is more powerful and more efficient.

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 test object 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.

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 *, ?, +), use the \ (backslash) character to instruct QuickTest to treat the special characters as literal characters. For more information about regular expressions, see “Understanding and Using Regular Expressions” on page 290.

The statement below specifies a WebEdit test object in the Mercury Tours page with the Name author and an index of 3. When the test runs, 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 about working with test objects, see Chapter 4, “Managing Test Objects.”

If you want to use the same programmatic description several times in one test or component, 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 about the **With** statement, see “With Statement” on page 896.

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 about regular expressions, see “Understanding and Using Regular Expressions” on page 290.

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

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

Set MyDescription = Description.Create()

Once 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.

Once you have filled 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 in order 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 or component if you want to use programmatic descriptions for several objects.

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

Retrieving ChildObjects

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. In order 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.

Once 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
```

For more information about the **ChildObjects** method, refer to the *QuickTest Professional Object Model Reference*.

Using Programmatic Descriptions for the WebElement Object

The **WebElement** object enables you to perform methods on Web objects that may not fit into any other Mercury test object class. The **WebElement** test object is never recorded, but you can use a programmatic description with the **WebElement** object to perform methods on any Web object in your Web site.

For example, when you run the statement below:

```
Browser("Mercury Tours").Page("Mercury Tours").  
WebElement("Name:=UserName", "Index:=0").Click
```

or

```
set WebObjDesc = Description.Create()  
WebObjDesc("Name").Value = "UserName"  
WebObjDesc("Index").Value = "0"  
Browser("Mercury Tours").Page("Mercury Tours").WebElement(WebObjDesc).  
Click
```

QuickTest clicks on the first Web object in the Mercury Tours page with the name **UserName**.

For more information about the **WebElement** object, refer to the *QuickTest Professional Object Model Reference*.

Using the Index Property in Programmatic Descriptions

The *index* property can sometimes be a useful test object property for uniquely identifying an object. The *index* test object 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.

Running and Closing Applications Programmatically

In addition to using the Record and Run dialog box to instruct QuickTest to open a new browser or application when a test run begins, and/or opening the application you want to test manually, you can also 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 most useful when your test includes more than one application and you have selected **Record and run test on any application** 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.

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, refer to the *QuickTest Professional Object Model Reference*.

Using Comments, Control-Flow, and Other VBScript Statements

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

For information on how to use these programming concepts in the Keyword View, see Chapter 20, “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, QuickTest does not process comments. Use comments to explain sections of a script in order to improve readability and to make tests 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 field.  
Browser("Mercury Tours").Page("Mercury Tours").WebEdit("username").  
    Set "mercury"
```

By default, comments are displayed in green in the Expert View. You can customize the appearance of comments in the Editor Options dialog box. For more information, see “Customizing Script Element Appearance” on page 717.

Note: You can also add a comment line using VBScript’s **Rem** statement. For additional information, refer to the Microsoft VBScript Language Reference (choose **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 site. VBScript supports the following mathematical operators:

Operator	Description
+	addition
-	subtraction
-	negation (a negative number—unary operator)
*	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 execute 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 executed 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 execute 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
<i>item</i>	A variable representing the element in the array.
<i>array</i>	The name of the array.
<i>statement</i>	A statement, or series of statements, to be executed 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 execute 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} condition]
      statement
Loop
```

Item	Description
<i>condition</i>	A condition to be fulfilled.
<i>statement</i>	A statement or series of statements to be executed 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 Statement

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

```
While condition
    statement
Wend
```

Item	Description
<i>condition</i>	A condition to be fulfilled.
<i>statement</i>	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 execute a statement or a series of statements based on specified conditions. If a condition is not fulfilled, the next Elseif condition or Else statement is examined. It has the following syntax:

```
If condition Then
    statement
Elseif condition2 Then
    statement
Else
    statement
End If
```

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

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 3.

```
value = 2
If value = 1 Then
    msgbox "one"
Elseif value = 2 Then
    msgbox "two"
Else
    msgbox "three"
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: Applying **With** statements to your script has no effect on the run session itself, only on the way your script appears in the Expert View.

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

```
With object
    statements
End With
```

Item	Description
<i>object</i>	An object or a function that returns an object.
<i>statements</i>	One or more statements to be executed 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 or Component” on page 491

Retrieving and Setting Test Object Property Values

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

When you run your test, QuickTest creates a temporary version of the test object that is stored in the test object repository. You use the **GetTOProperty**, **GetTOProperties**, and **SetTOProperty** methods to set and retrieve the test object 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 test run, any changes you make using the **SetTOProperty** method apply only during the course of the test run, 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 a *test object property* from a run-time object in your application.

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

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

Tip: If you do not know the test object properties of objects in your Web site or 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 test object properties supported by each object, and for additional information about the **GetROProperty**, **GetTOProperty**, **GetTOProperties**, and **SetTOProperty** methods, refer to the *QuickTest Professional Object Model Reference*.

Accessing Run-Time Object Properties and Methods

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

You can use QuickTest’s statement completion feature with object properties to view a list of the available native methods and properties of an object. For more information about the statement completion option, see “Generating Statements in the Expert View” on page 858.

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 900.

Retrieving Run-Time Object 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 additional information about the **Object** property, refer to the *QuickTest Professional Object Model Reference*.

Activating Run-Time Object Methods

You can use the **Object** property to activate the internal methods 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 additional information about the **Object** property, refer to the *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
```

Note: The *attribute/<property name>* notation is not supported in Netscape 4.x.

For more information about programmatic descriptions, see “Using Programmatic Descriptions” on page 878.

Running DOS Commands

You can run standard DOS commands in your QuickTest test 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 execute 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, refer to the Microsoft VBScript Language Reference (choose **Help > QuickTest Professional Help > VBScript Reference > VBScript**).

Enhancing Your Tests using the Windows API

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

The Windows API is documented in the Microsoft MSDN Web site and is accessible at: http://msdn.microsoft.com/library/en-us/winprog/winprog/windows_api_start_page.asp?frame=true

A reference to specific API functions can be found at:

http://msdn.microsoft.com/library/en-us/winprog/winprog/windows_api_reference.asp?frame=true

To use Windows API functions:

- 1 In MSDN, locate the function want to use in your test.
- 2 Read its documentation and understand all required parameters and return value(s).
- 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, refer to the *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 the example above, the foreground window's handle is retrieved. You can enhance your test if the foreground window is not in the object repository or can not 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, you need to find their numerical value in order 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_HWNDFPREV 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  
men_hwnd = Extern.GetMenu(hwin)' Get window's main menu's handle  
MsgBox men_hwnd  
' Use API Functions  
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, refer to the *QuickTest Professional Object Model Reference*.

Working with User-Defined Functions

In addition to the test objects, methods, and functions supported by the QuickTest Test Object Model, you can define your own VBScript functions subroutines, classes, modules, and so forth, and then call them from your test or component.

This chapter describes:

- About Working with User-Defined Functions
- Working with Associated Library Files
- Executing Externally-Defined Functions from Your Test or Component
- Using User-Defined Test Object Methods

About Working with User-Defined Functions

If you have large segments of code that you need to use several times in one test or component or in several different tests or components, you may want to create a user-defined function in order to make your tests or components easier to design and to read.

You can define these functions within an individual test or component, or you can create one or more external VBScript *library files* containing your functions, so that you can access them from any test or component.

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.

Note:  If you define a function directly in an action, the function can be called only in that action. If you want the function to be available for the entire test, save it in a **.vbs** file and specify it as an associated library file in the **Test > Settings > Resources** tab. For more information about specifying associated library files, see “[Associating Add-ins with Your Test or Component](#)” on page 662.

Working with Associated Library Files

You can create VBScript library files containing VBScript functions, subroutines, classes, modules, etc., and then associate the files with your test or component. You can call any VBScript function, subroutine, and so forth, contained within any library file that is associated with your test or component.

Any text file written in standard VBScript syntax can act as a library file. Note that the functions within library files can call QuickTest *reserved objects*, the objects that QuickTest supplies for enhancement purposes, such as utility objects.

Note: In addition to the functions available in the associated library files, you can also call a function contained in a VBScript file directly from any action or component using the **ExecuteFile** function. You can also insert **ExecuteFile** statements within an associated library file. For more information, see “[Executing Externally-Defined Functions from Your Test or Component](#),” below.

You can specify the default library files for all new tests in the Test Settings dialog box (**Test > Settings > Resources** tab). You can specify the default library files for all new components in the component template (**File > Business Component > Edit Template**). You can edit the list of associated library files for an existing test or component in the Test Settings dialog box or the Business Component Settings dialog box respectively. For more information, see “Defining Resource Settings for Your Test” on page 674 or “Defining Resource Settings for Your Component” on page 680.

Note:  After a test is created, the list of files specified in the Test Settings dialog box is independent of the files set as default in the Test Settings dialog box. Changes to the default library files list in the Test Settings dialog box do not affect existing tests.

Using Associated Library Files with Quality Center

When working with Quality Center and associated library files, you must save the associated library file as an attachment in your Quality Center project *before* you specify the associated file in the Resources tab of the Test Settings dialog box. You can add a new or existing library file to your Quality Center project.

Note:  If you add an existing library file from the file system to a Quality Center project, it will be a copy of the one used by tests not in the Quality Center project, and thus once you save the file to the project, changes made to the Quality Center file do not affect the file in the file system and vice versa.

To use an associated library file with Quality Center:

- 1** If you want to add a new library file, create a new file in your file system with a **.vbs** extension.
- 2** In Quality Center, add the library file to the project as an attachment.

3 In the Test Settings dialog box or Business Component Settings dialog box, click the **Resources** tab.

4 To add a new file to the associated library files list, when connected to Quality Center, click the **Add** button . QuickTest adds [QualityCenter], and displays a browse button so that you can locate the Quality Center path.

If you are not connected to Quality Center, hold the SHIFT key and click the **Add** 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.

Executing Externally-Defined Functions from Your Test or Component

You can create a VBScript file and call its functions, subroutines, classes, and so forth, from an action in your test, from a component, or from an associated library file, by inserting an **ExecuteFile** statement in your action, component, or library file.

When you run your test or component, the **ExecuteFile** statement executes all global code in the specified file in order to make all definitions in the file available from the global scope of the action's, component's, or library file's script.

Tip: If you want to include a certain **ExecuteFile** statement in every action you create, or every component created in a specific Quality Center project, you can add the statement to an action template or to the project's business component template. For more information, see “Creating an Action Template” on page 391, or “Working with Component Templates” on page 987.

To execute an externally-defined function:

- 1** Create a VBScript file using standard VBScript syntax. For more information, refer to 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 machine running your test or component.
- 3** Add an **ExecuteFile** statement to a component, to an action in your test, or in an associated library file using the following syntax:

ExecuteFile *FileName*

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

- 4** Use the functions, subroutines, classes, and so forth, from the specified VBScript file as necessary in your action or component, or within other functions in an associated library file.
-

Notes:

The **ExecuteFile** statement utilizes the VBScript **ExecuteGlobal** statement. For more information, see

<http://msdn.microsoft.com/library/default.asp?url=/library/en-us/script56/html/vsstmExecuteGlobal.asp>.

 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 library files list in the Resources tab of the Test Settings dialog box or the Business Component Settings dialog box. For more information, see "Working with Associated Library Files" on page 908.

Using User-Defined Test Object Methods

You can use the **RegisterUserFunc** method to add new methods to test objects or to change the behavior of an existing test object method during a run session.

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

To register a method, you first define a function in your test, component, or in an associated library file. You then enter a **RegisterUserFunc** statement in your test or component 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 in order to (temporarily) override the existing functionality of the specified method.

Note that your registered method applies only to the test or component in which you register it and that 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 or component if you want to limit its use only to the local action or component, or you can store the function in an associated library file in order to make it available to many actions and tests or components (recommended). Note that if the same function name exists within your action or component and within an associated library file, QuickTest uses the function defined in the action or component.

When you run a statement containing a registered method, it sends the test object as the first argument. Thus, 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. Note that if the function overrides an existing method, it should have 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** test object 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, or functions, and any method associated with the test object specified in the first argument of the function.

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

You can also use **RegisterUserFunc** statements within your user-defined function in order to call additional registered methods. See “Registering User-Defined Test Object Methods,” below, for information on how to write a **RegisterUserFunc** statement.

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 or component, 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 also takes 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*

Item	Description
<i>TOClass</i>	Any test object class.
<i>MethodName</i>	The method you want to register. 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.
<i>FunctionName</i>	The name of your user-defined function. The function can be located in your test or component, or in any library file associated with your test or component.

Note: You cannot register a method for a QuickTest reserved object (such as **DataTable**, **Environment**, **Reporter**, and so forth).

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 in order 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, refer to the *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 component, 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 library file, then the calling test may succeed (assuming the library file 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
<i>TOClass</i>	The test object class for which your method is registered.
<i>MethodName</i>	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 in order 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"

```

Guidelines for Registering User-Defined Test Object Methods

When registering user-defined methods, consider the following tips and guidelines:

- You can re-register the same method to use different user-defined functions without first unregistering the method. However, note that 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.

- QuickTest clears all method registrations at the beginning of each run session. Thus, if you use the begin running the test or component from a point after a method registration (using the **Run from step** option), QuickTest does not recognize the registration.
- You cannot register a method for a specific test object or test object instance. Method registrations apply to an entire test object class.
-  If you register a method in 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.

38

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 QuickTest's automation object model, you can write programs that configure QuickTest options and run tests or business components instead of performing these operations manually using the QuickTest interface.

Automation programs are especially useful for performing the same tasks multiple times or on multiple tests or components, or quickly configuring QuickTest according to your needs for a particular environment or application.

This chapter describes:

- About Automating QuickTest Operations
- Deciding When to Use QuickTest Automation Programs
- Choosing a Language and Development Environment for Designing and Running Automation Programs
- Learning the Basic Elements of a QuickTest Automation Program
- Generating Automation Scripts
- Using the QuickTest Automation Object Model Reference

About Automating QuickTest Operations

You can use the QuickTest Professional automation object model to write programs 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 easily 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 program.

Automation programs are especially useful for performing the same tasks multiple times or on multiple tests or components, or quickly configuring QuickTest according to your needs for a particular environment or application.

For example, you can create and run an automation program 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 or Business Component Settings, and Record and Run Settings dialog boxes, runs the test, and saves the test.

You can then add a simple loop to your program so that your single program can perform the operations described above for a number of tests.

You can also create an initialization program that opens QuickTest with specific configuration settings. You can then instruct all of your testers to open QuickTest using this automation program to ensure that all of your testers are always working with the same configuration.

Deciding When to Use QuickTest Automation Programs

Like the tests and business components you design using QuickTest, creating a useful QuickTest automation program 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 program.

The following are just a few examples of useful QuickTest automation programs:

- **Initialization programs**—You can write a program that automatically starts QuickTest and configures the options and the settings required for recording on a specific environment.
- **Maintaining your test or component database**—You can write a program that iterates over your entire test or component database to accomplish a certain goal. For example:
 - **Updating values**—opening each test or component with the proper add-ins, running it in update run mode against an updated application, and saving it in order to update the values in all of your tests or components to match the updated values in your application.
 - **Applying new options to existing test or components**—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 test or components. You can write a program that opens each existing test or component, sets values for the new options, then saves and closes it.
- **Calling QuickTest from other applications**—You can design your own applications with options or controls that run QuickTest automation programs. 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 Programs

You can choose from a number of object-oriented programming languages for your automation programs. For each language, there are a number of development environments available for designing and running your automation programs.

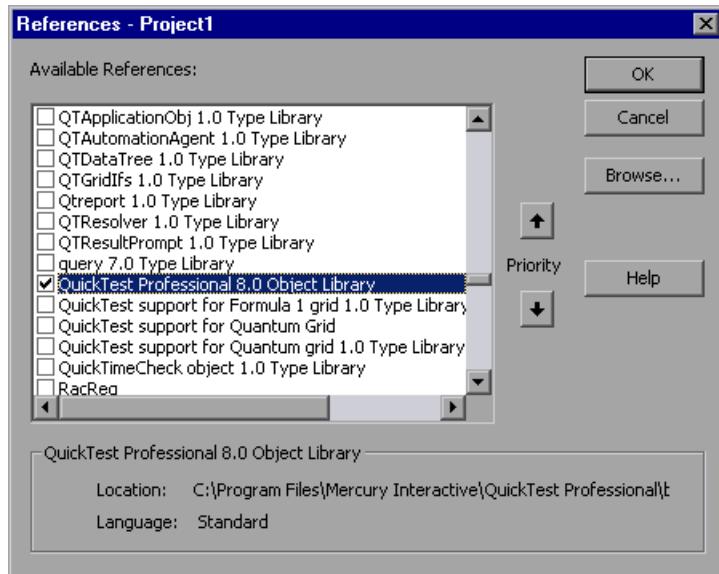
Writing Your Automation Program

You can write your QuickTest automation programs 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 program. 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 program. For example, if you are working in Microsoft Visual Basic, choose **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 Program

There are several applications available for running automation programs. You can also run automation programs from command line using Microsoft's Windows Script Host.

For example, you could use the following command line to run your automation program:

```
WScript.exe /E:VBSCRIPT myScript.vbs
```

Learning the Basic Elements of a QuickTest Automation Program

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 or business component 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 or components, 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 program 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 program). For more information, refer to the “Running Automation Programs on a Remote Computer” section of the online *QuickTest Automation Object Model Reference*.

The structure for the rest of your program depends on the goals of the program. You may perform a few operations before you start QuickTest such as retrieving the associated add-ins for a test or component, 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 program are performed after the **Launch** method.

For information on the operations you can perform in an automation program, refer to the online *QuickTest Automation Object Model Reference*. For more information on this Help file, see “Using the QuickTest Automation Object Model Reference” on page 927.

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 tab of the Test Settings dialog box, the General tab 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.NonUniqueListItemSelectedMode = "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, Test Settings, and Business Component Settings dialog boxes, see Chapter 24, “Setting Global Testing Options,” Chapter 33, “Configuring Object Identification,”, and Chapter 25, “Setting Options for Individual Tests or Components” respectively.

Using the QuickTest Automation Object Model 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 *QuickTest Automation Object Model Reference* from the:

- QuickTest program folder (**Start > Programs > QuickTest Professional > Documentation > QuickTest Automation Reference**)
- QuickTest Help menu (**Help > QuickTest Automation Object Model Reference**)

Part VIII

Working with Other Mercury Products

39

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 describes:

- About Working with WinRunner
- Calling WinRunner Tests
- Calling WinRunner Functions

About Working with WinRunner

If you have WinRunner 7.5 or later installed on your computer, you can include calls to WinRunner tests and functions in your QuickTest test or component.

Note: For WinRunner versions earlier than 7.6, you cannot run WinRunner tests on Web pages (using WinRunner's WebTest Add-in) from QuickTest if the QuickTest Web Add-in is loaded. For WinRunner 7.6, you can enable this functionality by installing patch **WR76P10 - Support WR/QTP integration** from the patch database on the Mercury Interactive Customer Support site (<http://support.mercury.com>). For future versions of WinRunner, this functionality will be provided built-in.

Once you create a call to a WinRunner test or function, you can modify parameter values in existing call statements by editing them in the Expert View. For information on working in the Expert View, see Chapter 36, "Working with the Expert 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.

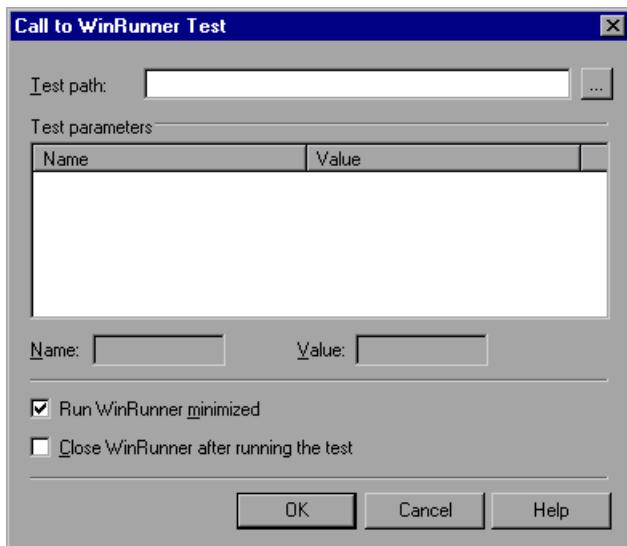
You can insert a call to a WinRunner test using the Call to WinRunner Test dialog box or by entering a **TSLTest.RunTestEx** statement in the Expert View.

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 Choose **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.

If you are connected to Quality Center when you click the browse button, the Open WinRunner Test from Quality Center project dialog box opens so that you can select the module from the Quality Center project. For more information on this dialog box, see “Opening Tests from a Quality Center Project” on page 952.

- 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 environment 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 935.

- 4 Select **Run WinRunner minimized** if you do not want to view the WinRunner window while the test runs. (This option is supported only for WinRunner 7.6 and later.)
- 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. (This option is supported only for WinRunner 7.6 and later.)
- 6 Click **OK** to close the dialog box.

For information on WinRunner test parameters, refer to the *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:

	Operation	Value
TSSTest	RunTestEx	"C:\WinRunner\Tests\basic flight",True,0,MyValue

- 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, if you are working with WinRunner 7.6 or later, it is recommended to update your tests to the **RunTestEx** method (and corresponding argument syntax). For more information on these methods, refer to the *QuickTest Professional Object Model Reference*.

For additional information on the **RunTestEx** method and an example of usage, refer to the *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, refer to the Utility section of the *QuickTest Professional Object Model Reference*.

Viewing the Results

When you run a call to a WinRunner test, and WinRunner 7.6 or later is installed on your computer, 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 603.

For more information on designing and running WinRunner tests, refer to 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.

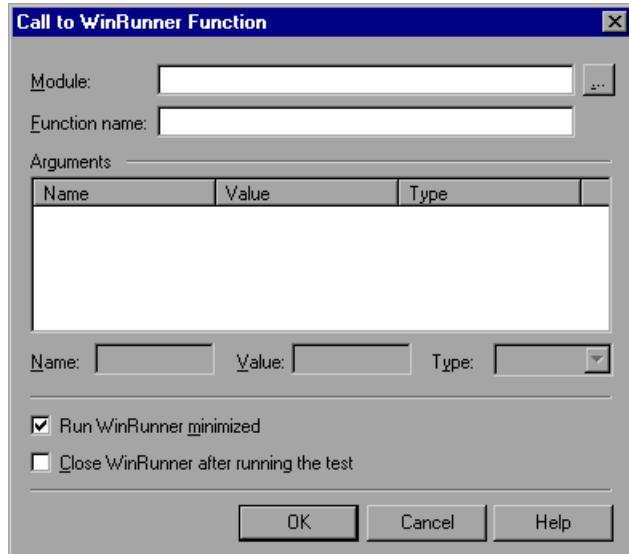
You call a WinRunner function from QuickTest by specifying the function and the compiled module containing the function.

Note: You cannot retrieve the values returned by the WinRunner function in your QuickTest test or business component. However, you can view the returned value in the results.

To call a user-defined function from a WinRunner compiled module:

- 1 Choose **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.

If you are connected to Quality Center when you click the browse button, the Open WinRunner Test from Quality Center project dialog box opens so that you can select the compiled module from the Quality Center project.

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.
 - In the **Type** box, select the correct argument type (**in/out/inout**).
 - If the argument type is “**in**” or “**inout**,” enter the value in the **Value** box.

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,” below.

For more information on function parameters, refer to the *WinRunner User’s Guide*.

- 6 Select **Run WinRunner minimized** if you do not want to view the WinRunner window while the function runs. (This option is supported only for WinRunner 7.6 and later.)

- 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. (This option is supported only for WinRunner 7.6 and later.)

- 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:

	Operation	Value
TSLTest	CallFuncEx	"C:\WinRunner\Tests\TIScript","TIScript",True,0,"MyArg1"

- a **TSLTest.CallFuncEx** statement in VBScript in the Expert View. For example:

```
CallFuncEx "C:\WinRunner\Tests\TIScript","TIScript1",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, if you are working with WinRunner 7.6 or later, it is recommended to update your tests to the **CallFuncEx** method (and corresponding argument syntax). For more information on these methods, refer to the *QuickTest Professional Object Model Reference*.

For additional information on the **CallFuncEx** method and an example of usage, refer to the *QuickTest Professional Object Model Reference*.

For information on WinRunner functions, function arguments, and WinRunner compiled modules, refer to the *WinRunner User's Guide* and the *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)
```

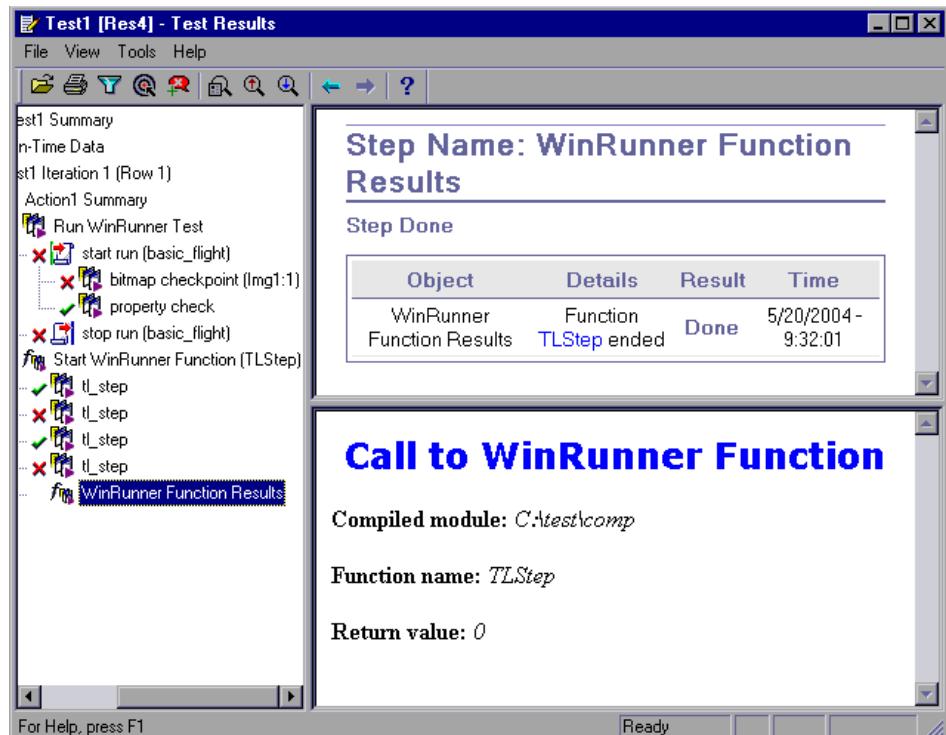
For more information on the syntax and usage of the **RandomNumber**, **Environment** and **DataTable** methods, refer to the Utility section of the *QuickTest Professional Object Model Reference*.

Viewing the Results

After you run a WinRunner function in WinRunner 7.6 or later 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.

Note: If you have WinRunner version 7.5 installed on your computer, you can view basic information about the WinRunner function run in the QuickTest Test Results window.

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, refer to your WinRunner documentation.

Part VIII • Working with Other Mercury Products

40

Working with Quality Center

To ensure comprehensive testing of your application or applications, you typically must create and run many tests. Mercury Quality Center, the centralized quality solution (formerly TestDirector), 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 both Quality Center and TestDirector. Refer to the *QuickTest Professional Readme* for a list of the supported versions of Quality Center and TestDirector.

This chapter describes:

- About Working with Quality Center
- Connecting to and Disconnecting from Quality Center
- Saving Tests to a Quality Center Project
- Opening Tests from a Quality Center Project
- Running a Test Stored in a Quality Center Project
- Managing Test Versions in QuickTest
- Setting Preferences for Quality Center Test Runs

About Working with Quality Center

QuickTest integrates with Quality Center, the Mercury 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 Mercury 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 components and save them in your Quality Center project. After you run your tests, you can view the results in Quality Center.

 Note that when working with Quality Center, you can associate tests with external files attached to 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 file stored in Quality Center.

For more information on specifying external files for all tests, see Chapter 24, "Setting Global Testing Options." For more information on specifying external files for a single test, see Chapter 25, "Setting Options for Individual Tests or Components."

You can report defects to a Quality Center project either automatically as they occur, or manually directly from QuickTest's 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 597.

You can use Quality Center with Business Process Testing support to create components and use them to build business process tests. For more information on components, see Chapter 41, "Working with Business Process Testing." For more information on Quality Center with Business Process Testing support, refer to the *Business Process Testing User's Guide*.

You can run QuickTest tests or components from Quality Center and then use Quality Center to review and manage the results. You can also use Quality Center with Business Process Testing support to create business process tests, comprised of the components you create in either QuickTest or Quality Center with Business Process Testing support. For more information, see Chapter 41, "Working with Business Process Testing."

For more information on working with Quality Center, refer to the *Mercury Quality Center User's Guide*. For the latest information and tips regarding QuickTest and Quality Center integration, refer to the *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 or component 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 or TestDirector. Refer to the *QuickTest Professional Readme* for a list of the supported versions of Quality Center and TestDirector.

To work with TestDirector 7.6 or earlier, you must install the TestDirector Connectivity Add-in on your QuickTest computer. For more information, see “Working with the Quality Center Connectivity Add-in” on page 950.

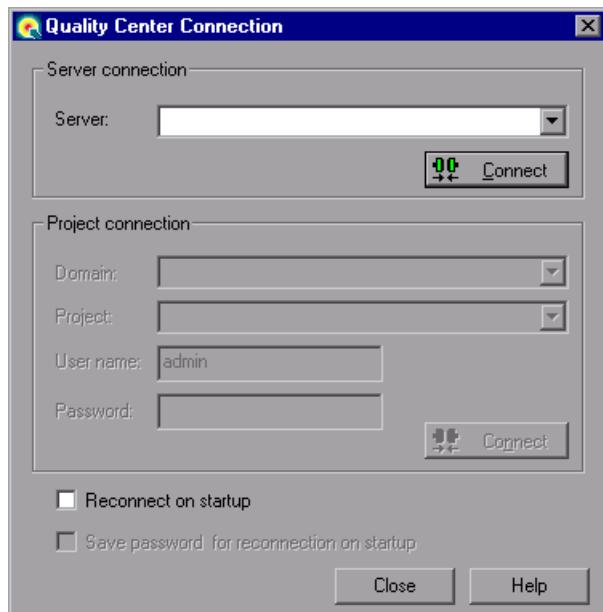
Connecting QuickTest to Quality Center

The connection process has two stages. First, you connect QuickTest to a local or remote Quality Center Web server. This server handles the connections between QuickTest and the Quality Center project.

Next, you choose the project you want QuickTest to access. The project stores tests or components and run session information for the Web site or 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 Quality Center:

- 1 Choose **Tools > Quality Center Connection** or click the **Quality Center Connection** toolbar button. The Quality Center Connection dialog box opens.



- 2 In the **Server** box, type the URL address of the Web server where Quality Center is installed.

Note: You can choose a Web server accessible via a Local Area Network (LAN) or a Wide Area Network (WAN).

- 3 In the **Server connection** area, click **Connect**.

Once the connection to the server is established, the server's name is displayed in read-only format in the Server box.

- 4 If you are connecting to a project in TestDirector 7.6 or later or Mercury Quality Center 8.0 or later, in the **Domain** box, select the domain that contains the Quality Center project.

- 5 In the **Project** box, select the project with which you want to work.
- 6 In the **User name** box, type a user name for opening the selected project.
- 7 In the **Password** box, type the password for the selected project.
- 8 In the **Project connection** area, click **Connect** to connect QuickTest to the selected project.

Once the connection to the selected project is established, the fields in the **Project connection** area are displayed in read-only format.

- 9 To automatically reconnect to the Quality Center server and the selected project the next time you open QuickTest, select the **Reconnect on startup** check box.
- 10 If the **Reconnect on startup** check box is selected, then the **Save password for reconnection on startup** check box is enabled. To save your password for reconnection on startup, select the **Save password for reconnection on startup** check box.

If you do not save your password, you will be prompted to enter it when QuickTest connects to Quality Center on startup.

- 11 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. To open the Quality Center Connection dialog box, double-click the **Quality Center** icon.

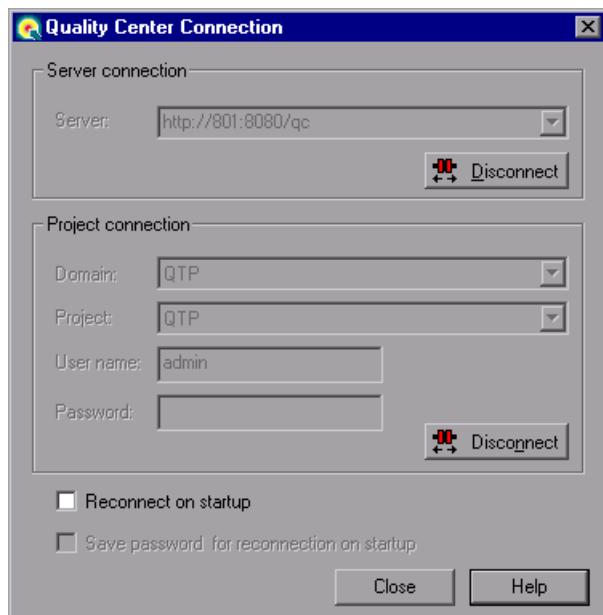
Disconnecting QuickTest from Quality Center

You can disconnect from a Quality Center project or a Web server. Note that if you disconnect QuickTest from a Web server without first disconnecting from a project, QuickTest's connection to that project database is automatically disconnected.

Note: If a Quality Center test, component, 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 Quality Center:

- 1 Choose **Tools > 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 **Project connection** area, click **Disconnect**.

- 3 To disconnect QuickTest from the selected Web server, in the **Server connection** area, click **Disconnect**.
- 4 Click **Close** to close the Quality Center Connection dialog box.

Working with the Quality Center Connectivity Add-in

Connecting to Quality Center requires the Quality Center Connectivity Add-in.

If you are working with Quality Center 8.0, this add-in is installed automatically when you connect to Quality Center in the Quality Center Connection dialog box.

To work with TestDirector 7.6 or earlier, you must manually install the TestDirector Connectivity Add-in that corresponds to your TestDirector version on your QuickTest computer.

To view the version of the Quality Center Connectivity Add-in that is currently installed on your computer, go to <**QuickTest Professional installation folder**>\TDAPIClient. Right-click the **tdclient.dll** file and click **Properties**.

To install the Quality Center Connectivity Add-in, choose **Quality Center Connectivity** from the Quality Center Add-ins page (available from the Quality Center main screen).

Note: The Quality Center Connectivity Add-in also enables access to TDOTA functionality (i.e. via an automation program), even if Quality Center is not installed on your computer. For more information on accessing TDOTA using automation programs, refer to the *QuickTest Automation Object Model Reference*.

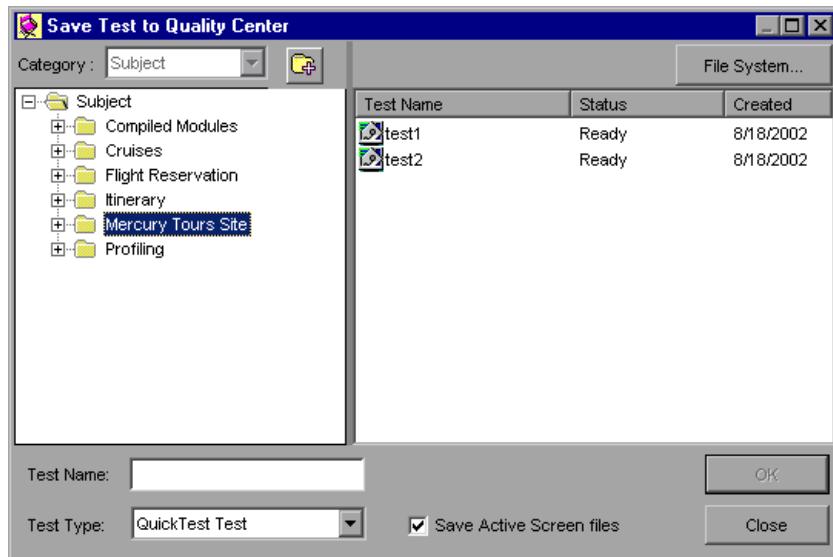
Saving Tests to a Quality Center Project

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.

Note: You also save components to a Quality Center project. For more information, see “Saving Components” on page 985.

To save a test to a Quality Center project:

- 1  Connect to a Quality Center server and project. For more information, see “Connecting QuickTest to Quality Center” on page 946.
- 2 In QuickTest, click **Save** or choose **File > Save** to save the test. The Save Test to Quality Center dialog box opens and displays the test plan tree.



Note that the Save Test to Quality Center dialog box opens only when QuickTest is connected to a Quality Center project.

To save a test directly in the file system, click the **File System** button to open the Save QuickTest Test dialog box. (From the Save QuickTest Test dialog box, you can return to the Save Test to Quality Center project dialog box by clicking the **Quality Center** button.)

- 3 Select the relevant subject in the test plan tree. To expand the tree and view a sublevel, double-click a closed folder. To collapse a sublevel, double-click an open folder.
- 4 In the **Test Name** box, enter a name for the test. Use a descriptive name that will help you easily identify the test. You cannot use the following characters in a test name: \ / : " ? < > | * % '
- 5 Confirm that the **Save Active Screen files** is selected if you want to save the Active Screen files with your test. Note that 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 96.
- 6 Click **OK** 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, the new test will be included in Quality Center’s test plan tree. For more information, refer to the *Mercury Quality Center User’s Guide*.

Opening Tests from a Quality Center Project

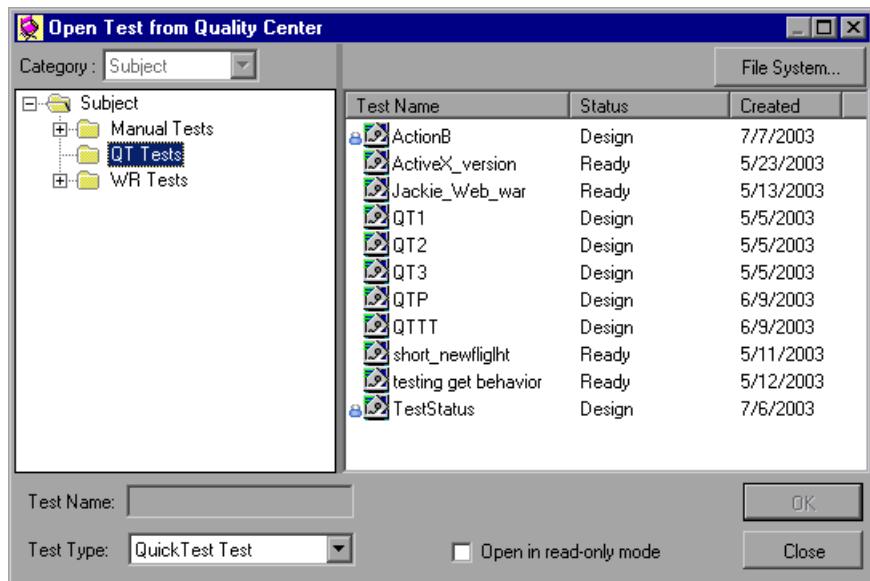
 When QuickTest is connected to a Quality Center project, you can open QuickTest tests that are a part of your Quality Center project. You locate tests according to their position in the test plan tree, rather than by their actual location in the file system. You can also open tests from the recent tests list in the **File** menu.

When you open a test in a Quality Center project with version control support, icons indicate the test's version control status. Quality Center version control support is not available for components.

Note:  You also open components from a Quality Center project. For more information, see “Opening Existing Components” on page 983

To open a test from a Quality Center project:

- 1 Connect to a Quality Center server and project. For more information, see “Connecting QuickTest to Quality Center” on page 946.
- 2 In QuickTest, click **Open Test** or choose **File > Open Test** to open the test. The Open Test from Quality Center dialog box opens and displays the test plan tree.



Note that the Open Test from Quality Center Project dialog box opens only when QuickTest is connected to a Quality Center project.

Note: To open a test directly from the file system while you are connected to Quality Center, click the **File System** button to open the Open Test dialog box. (From the Open Test dialog box, you can click the **Quality Center** button to return to the Open Test from Quality Center Project dialog box.)

- 3 Click the relevant subject in the test plan tree. To expand the tree and view sublevels, double-click closed folders. To collapse the tree, double-click open folders.

Note that when you select a subject, the tests that belong to the subject are displayed in the right pane of the Open Test from Quality Center Project dialog box.

► If the test is stored in a Quality Center project with version control support, icons next to the **Test Name** indicate the test's version control status. For more information, see "Opening Tests from a Quality Center Project with Version Control Support" on page 956.

► The **Test Name** column lists the names of the tests that belong to the selected subject.

► The **Status** column indicates whether each test is in **Design** stage or is **Ready** for test runs. Note that by default, tests saved to a Quality Center project from QuickTest are labeled as **Design**. The status can be changed only from the Quality Center client.

► The **Created** column indicates the date on which each test was created.

- 4 Select a test in the **Test Name** list. The test is displayed in the read-only **Test Name** box.

- 5 If you want to open the test in read-only mode, select the **Open in read-only mode** check box.

- 6 Click **OK** to open the test.

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 locked by another user, or whose attachments are currently locked by another user
- 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)

For more information, see “Opening Tests from a Quality Center Project with Version Control Support” on page 956 and “Creating, Opening, and Saving Tests with Locked Resources” on page 109.

Opening Tests and Components from the Recent Files List

You can open Quality Center tests and components from the recent files list in the File menu. If you select a test or component located in a Quality Center project, but QuickTest is currently not connected to Quality Center or to the correct project for the test or component, 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 or component on this computer.



Log in to the project, and click **OK**.

The Connect to Quality Center Project dialog box also opens if you choose to open a test or component 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 choose **File > Open** to open a test from a Quality Center project with version control support, the Open QuickTest Test from Quality Center Project 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.

Note: Quality Center version control support is not available for components.

The table below summarizes the version control status icons and the open mode for each status:

Icon	Description	Open Mode
<None>	The test is currently checked in to the version control database.	Read-only
	The test is currently checked out to you.	Read-write
	The test is currently checked out to another user.	Read-only
	An old version of the test is currently open on your computer.	As is

For more information about working with tests stored in a Quality Center project with version control, see “Managing Test Versions in QuickTest” on page 959.

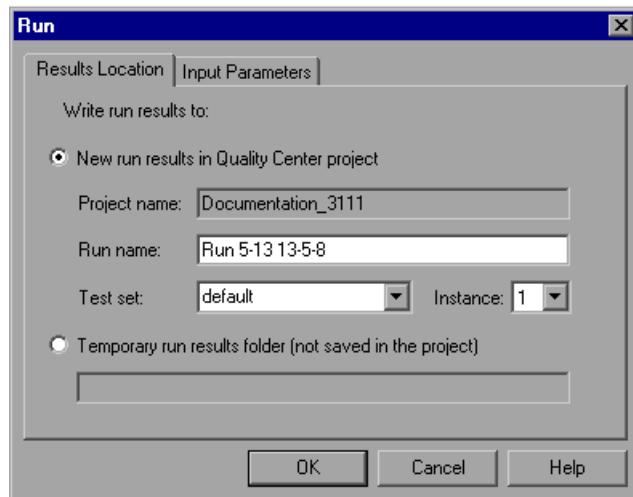
Running a Test Stored in a Quality Center Project

 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.

Note: Although components are saved in Quality Center, they are run in the same way as tests that are not stored in Quality Center. For more information, see Chapter 21, “Running Tests and Components.”

To save run results to a Quality Center project:

- 1 In QuickTest, click the **Run** button or choose **Test > 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 browse to select another 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 Quality Center's test run mode. For more information, refer to 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 <System Drive:>\Temp\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 tab of the Options dialog box. For more information about the Options dialog box, see Chapter 24, "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 QuickTest's Test Results window. For more information, see "Submitting Defects Detected During a Run Session" on page 597.

Managing Test Versions in QuickTest

 When QuickTest is connected to a Quality Center project with version control support, you can update and revise your automated test scripts while maintaining old versions of each test. This helps you keep track of the changes made to each test script, see what was modified from one version of a script to another, or return to a previous version of the test script.

You add a test to the version control data base 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 project with version control support requires the installation of version control software as well as Quality Center's Version Control Add-in. For more information, refer to 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. Quality Center version control support is not available for components.

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 962.

Checking Tests Out of the Version Control Database

 When you choose **File > Open** to open a test that is currently checked in to the version control database, it is opened in read-only mode.

Note: The Open Test from Quality Center Project dialog box displays icons that indicate the version control status of each test in your project. For more information, see “Opening Tests from a Quality Center Project” on page 952.

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. You can 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 962. For more information on undoing the check-out, see “Cancelling a Check-Out Operation” on page 967.

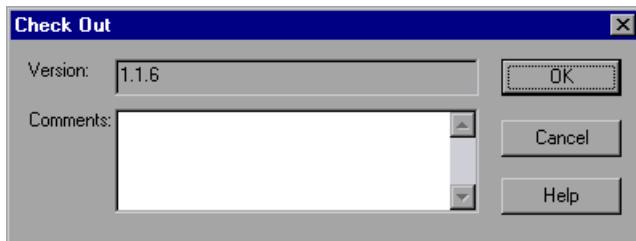
By default, the check out option accesses the latest version of the test. You can also check out older versions of the test. For more information, see “Using the Version History Dialog Box” on page 964.

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 Project” on page 952.

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 962. For more information on undoing the check-out, see “Cancelling a Check-Out Operation” on page 967.

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 “Cancelling a Check-Out Operation” on page 967.

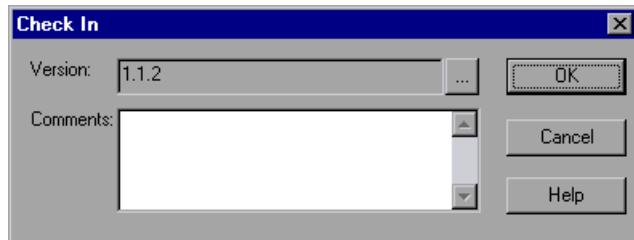
When you check a test back into the version control database, Quality Center deletes the test copy from your checkout directory and unlocks the test in the database so that the test version will be available to other users of the Quality Center project.

To check in the currently open test:

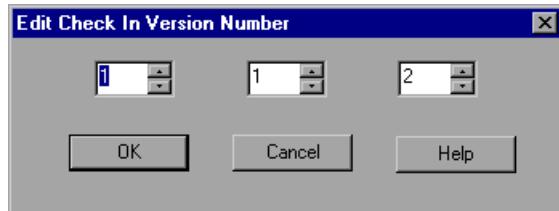
- 1 Confirm that the currently open test is checked out to you. For more information, see “Viewing Version Information For a Test” on page 964.

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.



- 3 Accept the default new version number and proceed to step 7, or click the browse button to specify a custom version number. If you click the browse button, The Edit Check In Version Number dialog box opens.



- 4 Modify the version number manually or using the up and down arrows next to each element of the version number. You can enter numbers 1-900 in the first element. You can enter numbers 1-999 in the second and third elements. You cannot enter a version number lower than the most recent version of this test in the version control database.

- 5 Click **OK** to save the version number and close the Edit Check In Version Number dialog box.
- 6 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.
- 7 Click **OK** to check in the test. The test closes and automatically reopens as a read-only test.

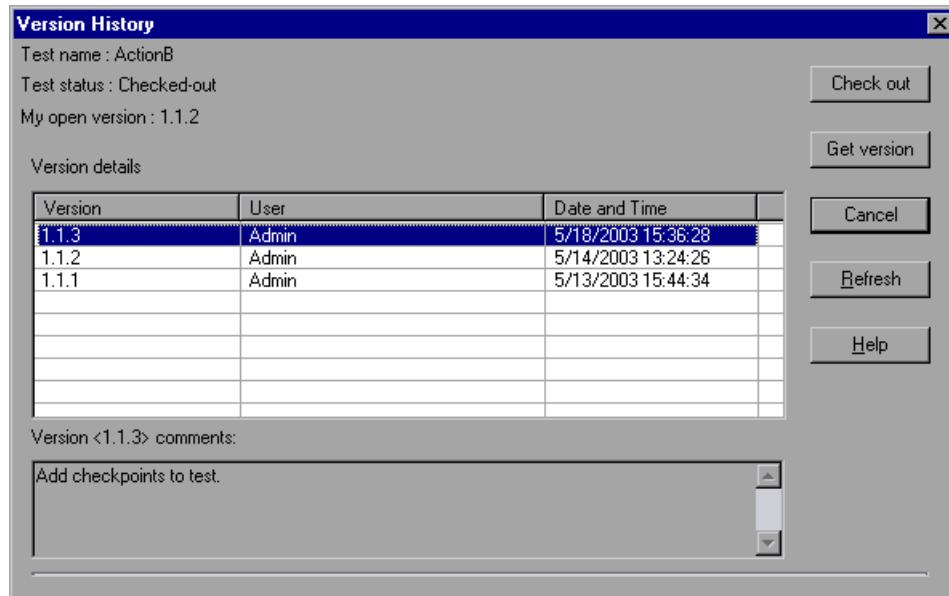
Using the Version History Dialog Box

 You can use the Version History dialog box to view version information about the currently open test and to view or retrieve an older 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

- **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 old version of a test in read-only mode or you can check out an old version and then check it in as the latest version of the test.

To view an old 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 Project” on page 952.
- 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.

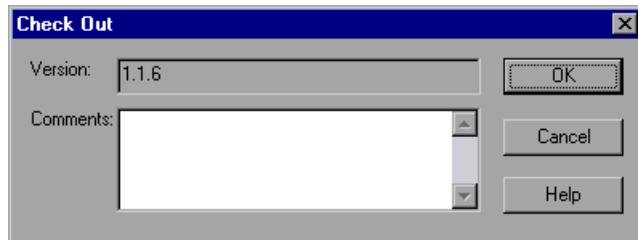
- 4 Click the **Get Version** button. QuickTest reminds you that the test will open in read-only mode because it is not checked out.
- 5 Click **OK** to close the QuickTest message. The selected version opens in read-only mode.

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 old version in read-only mode, you can check-out the open test by choosing **File > Quality Center Version Control > Check Out**. This is equivalent to using the **Check Out** button in the Version History dialog box.

To check out an old 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 Project” on page 952.
- 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 older 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 new, 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 962. For more information on undoing the check-out, see “Cancelling a Check-Out Operation” on page 967.

Cancelling 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.

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. Note that when a Quality Center client runs your QuickTest test, it uses the associated add-ins list to load the proper add-ins for your test. For more information, see “Modifying Associated Add-Ins” on page 663.

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 597.

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 Mercury products to run tests and components** option.

To enable remote Quality Center clients to run tests on your QuickTest computer:

- 1** Open QuickTest.
-  **2** Choose **Tools > Options** or click the **Options** toolbar button. The Options dialog box opens.
- 3** Click the **Run** tab.
- 4** Select the **Allow other Mercury products to run tests and components** check box.

For more information on this option, see “Setting Run Testing Options” on page 625.

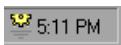
Tip: To access QuickTest tests from Quality Center, you must also have the QuickTest Add-in for Quality Center installed on the Quality Center computer. For additional information on this add-in, refer to the QuickTest Professional Add-in screen (accessible from the main Quality Center screen).

Setting QuickTest Remote Agent Preferences

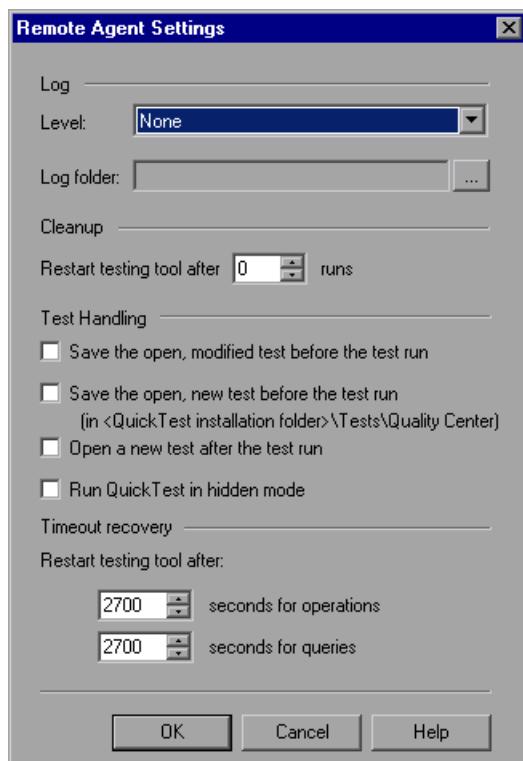
 When you run a QuickTest test from Quality Center, the QuickTest Remote Agent opens on the QuickTest computer. The QuickTest Remote Agent determines how QuickTest behaves when a test is run by a remote application such as Quality Center.

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 QuickTest test on your computer.

To open the Remote Agent Settings dialog box:



- 1 Choose **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 choose **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,” below.
- 4** Click **OK** to save your settings and close the dialog box.
- 5** Right-click the Remote Agent icon and choose **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 your QuickTest application uses when Quality Center runs a QuickTest test on your computer.

The Remote Agent Settings dialog box contains the following options:

Option	Description
Level	<p>The level of detail to include in the log that is created when Quality Center runs a QuickTest test.</p> <p>None (default)—No log is created.</p> <p>Low—The log lists any Quality Center-QuickTest communication errors.</p> <p>Medium—The log includes Quality Center-QuickTest communication errors and information on other major operations that result in Quality Center-QuickTest communication.</p> <p>High—The log includes all available information related to Quality Center-QuickTest communications.</p>
Log folder	<p>The folder path for storing the log file. Required if a log type is specified in the Level option.</p>
Restart testing tool after _____ runs	<p>Restarts the QuickTest application after the Quality Center completes the specified number of test runs. When QuickTest restarts, it continues with the next test in the test set.</p> <p>You may want to use this option to maximize available memory.</p> <p>If you do not want QuickTest to restart during a test set, enter 0 (default).</p>

Option	Description
Save the open, modified test before the test run	If an existing (named) test is open in QuickTest when the Remote Agent begins running a test, this option ensures that any modifications to the test are 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, this option saves the test 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.
Run QuickTest in Hidden Mode	Specifies whether to run QuickTest in hidden (silent) mode.
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 Quality Center's 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.

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 describes:

- About Working with Business Process Testing
- Understanding Components
- Creating Components
- Opening Existing Components
- Working with Component Templates
- Recording Components
- Running Components

About Working with Business Process Testing

Business Process Testing enables subject matter experts to create tests using a new methodology for testing as well as an improved automated testing environment.

Business Process Testing is combined with Quality Center and can be enabled by purchasing a specific Business Process Testing license. In order 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, refer to the *Business Process Testing User's Guide*.

The remaining sections in this chapter describe options and features that are unique to working with components in QuickTest. QuickTest options and features that are common or similar for both components and tests are described in the relevant chapters throughout this user's guide.

Understanding Business Process Testing Roles

The Business Process Testing model is role-based, allowing non-technical subject matter experts to define and document business processes, business components, and business process tests, while Testing Tool Engineers record and program the individual steps of business components, and QA testers can concentrate on running and debugging individual components.

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. For example, the tasks of the Business Component Expert and the Business Process Expert may be performed by the same person.

The following four basic user roles are identified in the Business Process Testing model:

- **Business Component Expert**—The Business Component Expert is a subject matter expert who has a detailed understanding of the individual elements and tasks that are fundamental to an application. This also enables the Business Component Expert to identify the key business activities that are common to more than one business process. Using the Business Components module, the Business Component 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.

For example, most applications require users to log in before they can access any of the application functionality. The Business Component Expert could create one business component that represents this login procedure. These component procedures can also be reused in other business process tests, resulting in easier and more cost-efficient maintenance, updating, and test management.

The Business Component Expert also determines whether there is a need to implement any new business components requested by the Business Process Expert, and if so, where in the component tree that the new components should be located.

- **Business Process Expert**—The Business Process Expert is a subject matter expert who has specific knowledge of the application logic, and has a high-level understanding of the entire system. This enables the Business Process Expert to determine the operating scenarios or business processes that must be tested. Using the Test Plan module, the Business Process Expert combines the business components created by the Business Component Expert into business process tests, composed of a serial flow of the business components.

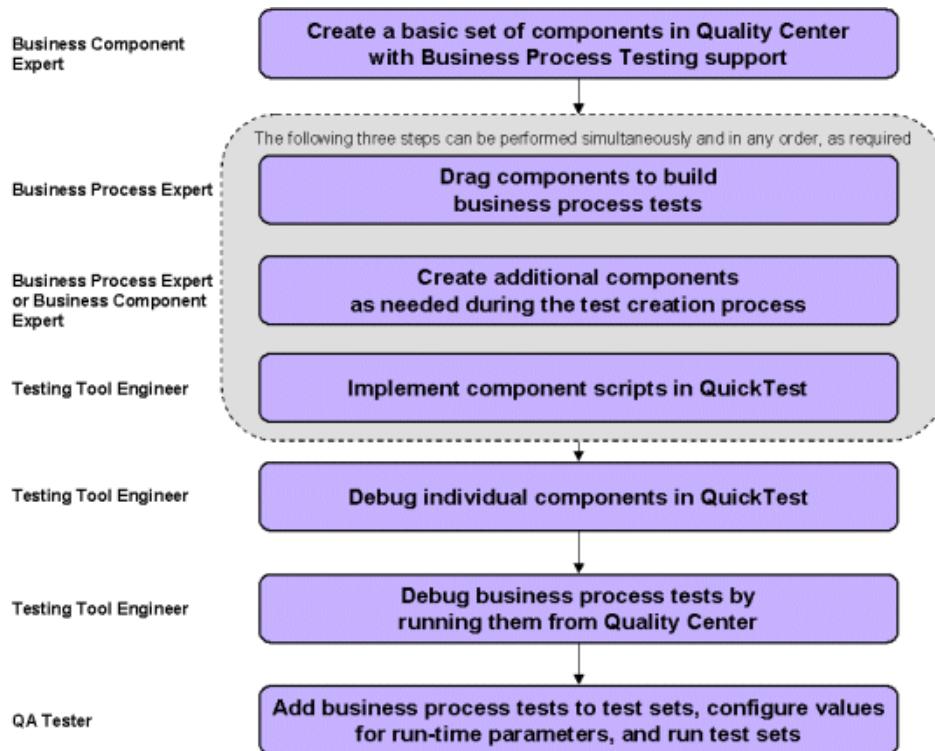
The Business Process Expert also generates requests for new business components if he or she thinks that no existing component answers the needs of a business process test.

- **Testing Tool Engineer**—The Testing Tool Engineer is an expert for an automated testing tool, such as QuickTest Professional. The Testing Tool Engineer implements and maintains the testing steps for each of the individual business components initiated by the Business Component Expert.
- **QA Tester**—The QA Tester is a Quality Center user who configures the values used for business process tests, runs them in test sets, and reviews the results.

One of the great advantages of the Business Process Testing model is that the work of the subject matter expert is not dependent on the completion of business components by the testing tool specialist. Hence, using Business Process Testing, the testing process can start before the application to be tested is at a level at which automated business components can be recorded.

Understanding the Business Process Testing Workflow

The Business Process Testing workflow may differ according to your testing needs. Following is an example of a common workflow:



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.

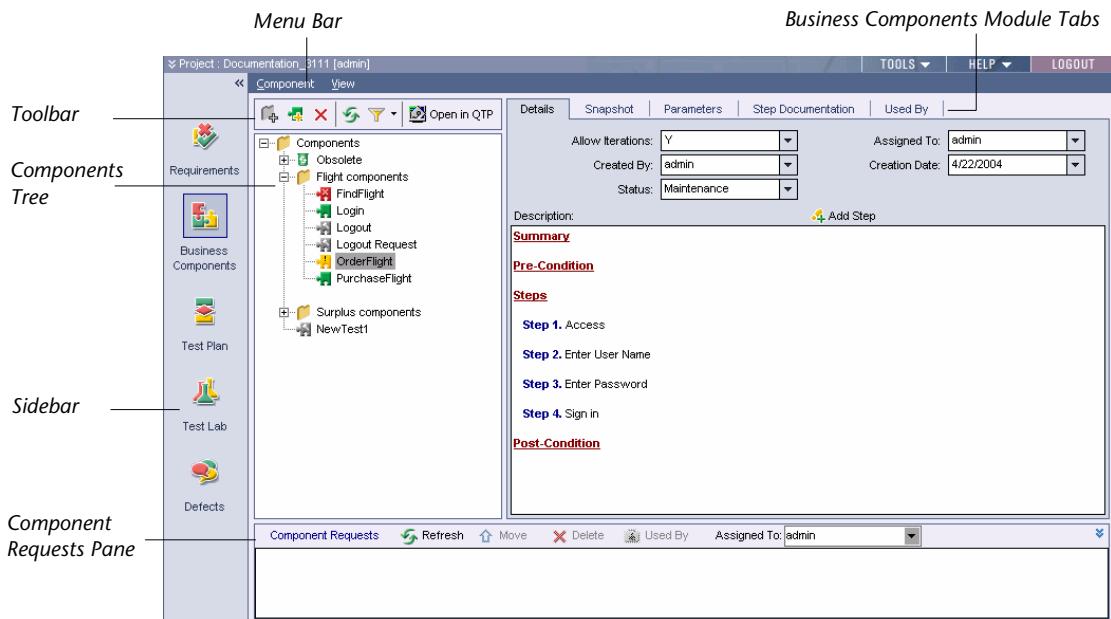
Understanding Components

A component is a reusable script that is easily maintained. The component creation process can be divided into two elements: the component shell and the component implementation.

- The *component shell* is the component's outer layer. The information in the shell is visible or available at the test level. The Business Component Expert defines the component shell information. Once the Business Component Expert creates a component shell, it can be used to build business process tests even if the implementation has not yet begun.
- The *component implementation* is the component's inner layer. It includes the actual script and specific settings for the component. The information can be seen only at the component level. You create the component implementation using QuickTest Professional.

Creating Components in the Quality Center Components Module

The Business Component Expert can create a new component and define its shell in the Quality Center Components module.



The component shell includes the following elements:

- **Details**—A general summary of the component’s purpose or contents, plus more detailed instructions that define the preconditions, for example, the application’s state at the start of the component, the steps the component should perform, and the post conditions for the component, for example, the state in which the application should finish and whether the component should finish in such a way that multiple component iterations are possible.
- **Snapshot**—An image that provides a visual cue or description of the component’s purpose or operations.
- **Input parameters**—The name, default value, and description of the data the component can receive.
- **Output parameters**—The name and description of the values that the component can return to the business process test.
- **Status**—The current status of the component, for example, whether the component is fully implemented and ready to be run, or whether it has errors that need to be fixed. The highest severity component status defines the overall status of the business process test.

The component shell is exposed to other components that are used in the same business process test. Using output and input parameters, you can transfer data from one component to a later component in the business process test.

Implementing Components in QuickTest Professional

You implement the component in QuickTest Professional. In most cases, you open and implement an existing component whose shell was defined by the Business Component Expert.

From QuickTest, you implement components by recording steps on any supported environment, adding checkpoints and output values, parameterizing selected items, and enhancing your test with flow statements, user-defined functions, and other VBScript statements.

From QuickTest you can also view and set options specific to components. For example, you can view the component description, modify the component screenshot, and you can determine whether component iterations are applicable for the component.

Once a component has been implemented, the Business Component Expert can open the component in the Quality Center Components module to view a summary of a component's steps in understandable sentences in the **Summary** tab.

Creating Business Process Tests in the Quality Center Test Plan Module

To create a business process test, the subject matter expert selects (drags and drops) the components that apply to the business process test and configures their run settings.

Each component can be used differently by different business process tests. 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, the subject matter expert realizes that a component has not been defined for an element that is necessary for the business process test, 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. 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 Quality Center tester runs it from the Test Lab module similar to the way any other test is run in Quality Center. Before running the test, Quality Center tester can define run-time parameter values and iterations using the **Iterations** column in the Test Lab module grid.

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 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 create and edit components in QuickTest by recording steps on any supported environment, adding checkpoints and output values, parameterizing selected items, and enhancing the component with flow statements, user-defined functions, and other VBScript programming statements. A subject matter expert combines components into business process tests, which are used to check that the Web site or application behaves as expected.

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 record, edit, and run components. The guidelines in the sections below provide an overview of these differences.

General Differences Between Components and Tests

Following are guidelines and information regarding differences between components and tests:

- 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 files are stored in the Quality Center project to which you are currently connected.
- There is no per-action (or per-component) object repository option. All components must use a shared object repository file (stored in Quality Center).
- The name of the component node in the Keyword View is the same as the saved component. You cannot rename the node.
- Specific menu options are used to create and edit components (under **File > Business Component**). If a component is currently open, you can use the **New** and **Open** toolbar buttons to create or edit additional components.



Differences When Using the Data Table with Components

Following are guidelines and information you should consider when using the Data Table with components. For more information on Data Tables, see Chapter 18, “Working with Data Tables.”

- A component has only one (local) data sheet. It does not have a global data sheet. You can refer to the data sheet in a statement by specifying the name of the sheet (same as the name of the component) in quotes or by using the constant, **dtlocalsheet** (without quotes). You can also refer to the data sheet using the **DataTable.LocalSheet** method.
- You can use only the first row in the Data Table for Data Table parameter and output values. This is because component iterations are defined in Quality Center (based on the number of component parameter iterations you define).

Other rows in the Data Table can still be used for calculating formulas, and so forth.

- The component always uses the Data Table file that is saved with the component. You cannot associate an external Data Table file with a component.

Creating Components

When QuickTest is connected to a Quality Center project with Business Process Testing support, you can create a new component in that project.

New components are based on the component template for the Quality Center project in which you intend to save the component. For information on component templates, see “Working with Component Templates” on page 987.

Tip: You can also create a new component in Quality Center, as described in the *Business Process Testing User’s Guide*.

To create a component:

- 1 Connect to a Quality Center project with Business Process Testing support. For information on connecting to Quality Center, see “Connecting QuickTest to Quality Center” on page 946.
- 2 Choose **File > Business Component > New**. A new component opens, based on the project component template.



Tip: If a component is already open, you can also click the **New** toolbar button to open a new component.

Notes: If your component template associates new components with a locked shared object repository, a message regarding locked resources opens when you create a new component. For more information, see “Creating, Opening, and Saving Tests with Locked Resources” on page 109.

If the shared object repository with which new components are associated by default cannot be found, QuickTest creates a new, blank shared object repository with the original shared object repository name.

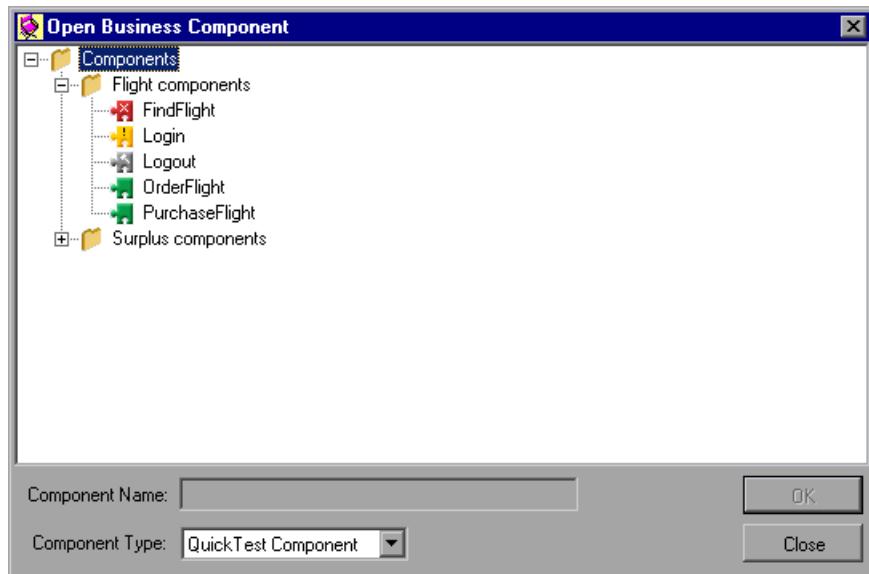
Opening Existing Components

When QuickTest is connected to a Quality Center project with Business Process Testing support, you can open a component that is stored in the project to view, modify, or run it. You find components according to their location in the component tree.

To open an existing component:

- 1 Connect to the Quality Center project in which your component is saved. For information on connecting to Quality Center, see “Connecting QuickTest to Quality Center” on page 946.

- 2** Choose **File > Business Component > Open**. The Open Business Component dialog box opens showing all of the components in the current Quality Center project.



The status of each component is indicated by its icon. For information on component statuses and their icons, refer to the *Business Process Testing User's Guide*.



Tip: You can open a recently used component by selecting it from the Recent Files list in the **File** menu. If a component is already open, you can also click the **Open** toolbar button to open another component.

-
- 3** Click the relevant folder in the component tree. To expand the tree and view the components, double-click closed folders. To collapse the tree, double-click open folders.
- 4** Select a component. The component name is displayed in the read-only **Component Name** box.
- 5** Click **OK** to open the component.

As QuickTest downloads and opens the component, the operations it performs are displayed in the status bar.

When the component opens, the QuickTest title bar displays [Component] [Quality Center], the full path and the component name. For example, the title bar for a flight_login component may be:

[Component] [Quality Center] Components\Flight\flight_login

Notes: If the component you are opening is associated with a shared object repository that cannot be found, a message opens enabling you to create a new, blank shared object repository for the component, to select a different shared object repository file for the component, or to open a new blank test with a per-action object repository.

If the component you are opening is associated with a locked shared object repository, a message opens instructing you how to open the component. For more information, see “Creating, Opening, and Saving Tests with Locked Resources” on page 109.

Saving Components

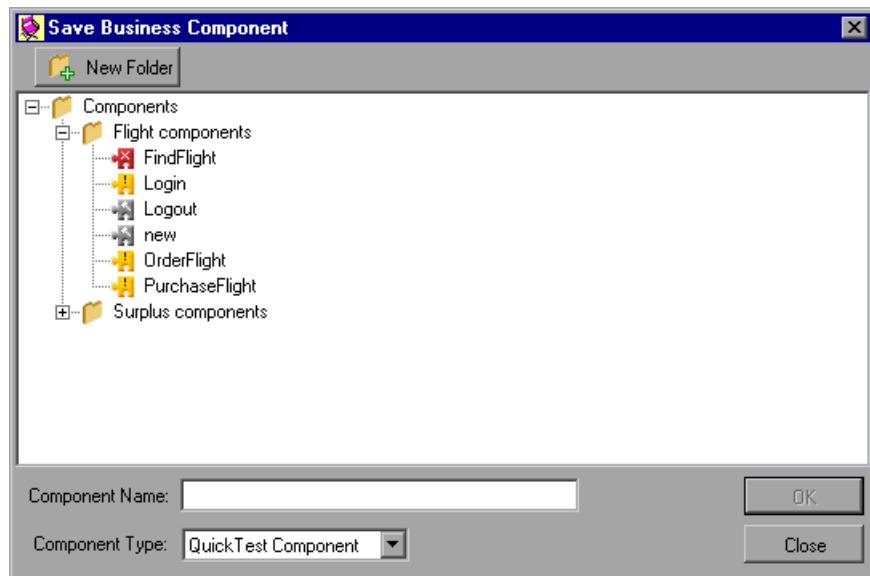
When QuickTest is connected to a Quality Center project with Business Process Testing support, you can save components to your project. To save a component, you give it a descriptive name and associate it with the relevant folder in the component tree. This helps you to keep track of the components created for each subject and to quickly view the progress of component planning and creation.

To save a component to a Quality Center project:

- 1 Connect to a Quality Center server and project with Business Process Testing support. For more information, see “Connecting QuickTest to Quality Center” on page 946.
- 2 In QuickTest, click **Save** or choose **File > Save** to save the component.



The Save Business Component dialog box opens and displays the component tree.



- 3 Select the component folder in the component tree. To expand the tree and view a sublevel, double-click a closed folder. To collapse a sublevel, double-click an open folder.

You can either save the component in an existing folder in your Quality Center project or click the **New Folder** button to create a new folder in which to save it.

- 4 In the **Component Name** box, enter a name for the component. Use a descriptive name that will help you easily identify the component. You cannot use the following characters in a component name:
\ / : ? < > | * % '
- 5 Click **OK** to save the component and close the dialog box. Note that the text in the status bar changes while QuickTest saves the component.

The next time you start Quality Center, the new component will be included in Quality Center's Component module. For more information, refer to the *Quality Center User's Guide*.

Working with Component Templates

The component template enables you to specify default component-specific settings that apply to all new components created in a specific Quality Center project. For example, you may want to specify that all new components use the same recovery scenarios.

You can also specify one or more statements to include in each new component you create in a specific Quality Center project, by adding these statements to the component template. For example, you may want to add a comment with project information at the start of each component, such as 'ActiveX Project for AutoTesting Version 8.1.11. You may also want to add a line that specifies that all components should call a specific function, such as `InitApplication("\\serverA\DomainXXX","UserName","Password")`.

Although this is not its main purpose, when working with the component template you can also perform other standard testing operations, such as recording steps on objects.

Each project has its own component template that is used as the base for all new components created in that project. You can modify the information in the component template to suit your needs. For information on the default component settings included with the component template, see "Guidelines for Specifying Component Template Settings" on page 988.

To modify the component template:

- 1 Connect to the Quality Center project whose component template you want to modify. For information on connecting to Quality Center, see "Connecting to and Disconnecting from Quality Center" on page 946.
- 2 Choose **File > Business Component > Edit Template**. The component template opens.

Note: If the component template is associated with a shared object repository that cannot be found, a message opens enabling you to create a new, blank shared object repository for the component template, or to select a different shared object repository file for the component template.

- 3 Modify the default component settings in the various tabs of the Business Component Settings dialog box (**Component > Settings**). For more information, see “Guidelines for Specifying Component Template Settings” on page 988.
- 4 If you wish, add or modify default statements. For example, comments, function calls, steps, or statements that open necessary applications or close specific applications.
- 5  Click **Save** or choose **File > Save** to save the component template. All new components created in this Quality Center project will use the settings and component information you defined. Existing components in the project, or components in other projects, are not affected by the changes you make to the component template.

Guidelines for Specifying Component Template Settings

You can choose **Component > Settings** to open the Business Component Settings dialog box and specify the settings to be used by default for all new components in a Quality Center project.

Since some options cannot be specified as default settings or are not relevant for the component template, the Business Component Settings dialog box for the component template has fewer options than the Business Component Settings dialog box for a regular component.

For general information on component settings and how to define them, see Chapter 25, “Setting Options for Individual Tests or Components.”

Note: Settings made previously in the Test Settings dialog box, including those that were specified as default settings for tests, do not apply to components or to the component template.

Following are the settings that can be defined in the component template as default settings for all new components. Any tabs or options not mentioned below are either not available for component templates, or do not have any effect on the default component settings.

- In the Properties tab, you can specify the default set of add-ins that are associated with new components. Note that the specified add-ins may be different than the add-ins that are loaded when a new component is created.
- The default component template specifies the Web Add-in and the ActiveX Add-in as the associated add-ins for all new components.
- In the Resources tab, you can specify the default library files and shared object repository file for all new components.

The default component template does not specify any library files and specifies the shared object repository file for all new components as **[QualityCenter] Subject\Default.tsr**.

- In the Applications tab, you can specify the Windows-based applications on which new components can record and run by default, and you can view other environments on which you can currently record and run (based on the currently loaded add-ins).

The default component template does not specify any Windows-based applications.

- In the Recovery tab, you can specify the default recovery files to use with new components.

The default component template does not specify any recovery scenarios.

- If you currently have add-ins loaded that have corresponding settings tabs (such as Web), you can specify the default settings for those environments.

Guidelines for Working with Component Template Scripts

When working with the component template, there are several testing operations that are either not available or not relevant, as described below:

- You cannot insert checkpoints and output values on steps in the component template.
- The Active Screen context (right-click) menu is not available.
- Active Screen files captured and saved with the template are not applied to new components.
- Values entered into the template Data Table are not applied to new components based on the template.
- Although the parameterization options are available, these options are not relevant for component templates and should not be used.

Recording Components

You record a component by performing and recording the typical processes that users perform.

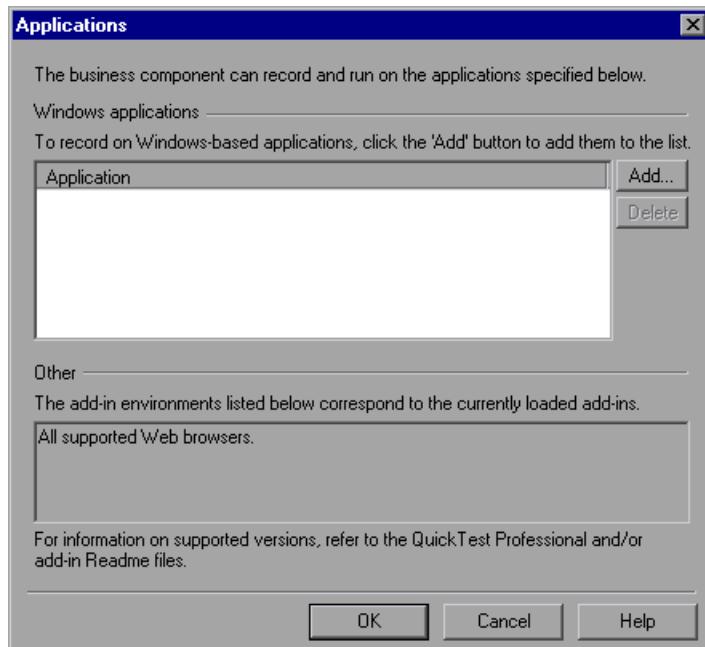
To record a component:



- 1 Open QuickTest. For more information, see “Starting QuickTest” on page 12.
- 2 Open a new or existing component.
For more information, see “Creating Components” on page 982 and “Opening Existing Components” on page 983.
- 3 Click the **Record** button or choose **Component > Record**.



If you are recording a new component and have not yet set your application settings in the Applications tab of the Business Component Settings dialog box (**Component > Settings**), the Applications dialog box opens.



The Applications dialog box serves the following purposes:

- Informs you on which environments you can currently record (based on the currently loaded add-ins).
 - Enables you to define on which Windows-based applications you want to record and run steps. You can record and run steps only on the specified applications.
-

Notes: Web applications are automatically recorded on when the Web Add-in is loaded, so there is no need to add them to the list.

To record on an application, you must open it manually or include statements in your component that open the applications you need. The Applications dialog box does not include settings for automatically opening applications.

The Applications dialog box will not open the next time you start a session in the same component. However, you can choose **Component > Settings** to open the Business Component Settings dialog box and use the Applications tab to set or modify your application preferences in the following scenarios:

- You have already recorded one or more steps in the component and you want to modify the settings before you continue recording.
 - You want to run the component on a different application than the one you previously used.
- 4** Specify the Windows-based applications on which you want to record and run this component. The options in the Applications dialog box are identical to the options in the Applications tab of the Business Component Settings dialog box. For more information, see “Defining Application Settings for Your Component” on page 670.
- 5** Click **OK** to close the Applications dialog box and begin recording your component.
- 6** Navigate through your application or Web site. QuickTest records each step you perform and displays it in the Keyword View and Expert View.

- 7 To determine whether your Web site or application is functioning correctly, you can insert a variety of checkpoints. For more information, see Chapter 6, “Understanding Checkpoints.”
- 8 You can parameterize your component 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 12, “Parameterizing Values.”
- 9 When you complete your recording session, click the **Stop** button or choose **Component > Stop**.
- 10 To save your component, click the **Save** button or choose **File > Save**. In the Save Business Component dialog box, assign a name to the component.



Note: You cannot use the following characters in a component name:
\\ / : ? < > | * % '



You can either save the component in an existing folder in your Quality Center project or click the **New Folder** button to create a new folder in which to save it.

For additional information on recording sessions, see “Recording a Test or Component” on page 88.

Running Components

When you run a component, QuickTest performs the steps you recorded on your application or Web site. When the run session is complete, QuickTest displays a report detailing the results. For general information on run sessions, see Chapter 21, “Running Tests and Components.”

QuickTest does not open any applications automatically at the beginning of a component run. You must record the opening of necessary applications, open them manually, or insert **SystemUtil.Run** statements into your component to instruct QuickTest to open the applications during the run session. For more information on the **SystemUtil.Run** method, refer to the *QuickTest Professional Object Model Reference*.

The run session preference options that are available in the Run tab of the Test Settings dialog box are not available for components, for the following reasons:

- The Data Table options that can be defined for tests are not relevant for components, since iterations for individual components and business process tests are defined in Quality Center.
- When running a test, the **On Error** behavior is determined by the run mode specified for the component. When running a component from QuickTest or when running from Quality Center in debug mode, message boxes open when errors occur. When running a component from Quality Center in normal mode, QuickTest proceeds to the next step after an error occurs.
- The object synchronization timeout for a component is always 20 seconds (20000 milliseconds).
- Smart identification can be disabled for tests but not for components.

When you run a component from QuickTest, the results are stored locally on your computer. By default, QuickTest saves the results in the Quality Center cache path on your computer. You can modify this path to any other file system path or choose to save the results in the temporary results folder. For general information on viewing results, see Chapter 23, “Analyzing Test Results.”

Note: Results are stored in Quality Center only when the component is run as part of a business process test from Quality Center (not in **Debug** mode). For information on running business process tests, refer to the *Business Process Testing User’s Guide*.

Working with Mercury Performance Testing and Application Management Products

 Once you have used 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.

Mercury LoadRunner tests the performance of applications under controlled and peak load conditions. To generate load, LoadRunner runs hundreds or thousands of virtual users. These virtual users provide consistent, repeatable, and measurable load to exercise your application just as real users would.

Mercury Application Management enables real-time monitoring of the end user experience. The 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.

This chapter describes:

- About Working with Mercury Performance Testing and Application Management Products
- Using QuickTest's Performance Testing and Application Management Features

- Designing QuickTest Tests for Use with LoadRunner or the Business Process Monitor
- Inserting and Running Tests in LoadRunner or Mercury Application Management

About Working with Mercury Performance Testing and Application Management 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 recording mechanisms used in all Mercury Performance Testing and Application Management products are the same. This means that you can create tests that are compatible with Mercury LoadRunner and Mercury Application Management, enabling you to take advantage of tests or test segments that have already been designed and debugged in QuickTest, and use them as the basis for your work in other Mercury Performance Testing and Application Management products.

For example, you can add QuickTest tests to specific points in a LoadRunner scenario to confirm that the application's functionality is not affected by the extra load at those sensitive points.

QuickTest also offers several features that are designed specifically for integration with LoadRunner and the Business Process Monitor. However, since LoadRunner and the Business Process Monitor 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 and/or Mercury Application Management, you should take into account the different options supported in each product as you design your test.

Using QuickTest's Performance Testing and Application Management Features

QuickTest includes an option for saving integration data with your test. This data makes it possible to run tests designed in QuickTest using Mercury Application Management or LoadRunner. You can also take advantage of other QuickTest features that were designed primarily for LoadRunner and Mercury Application Management users.

Saving Integration Data with Your Tests

To enable integration with LoadRunner and Mercury Application Management Virtual User technology, QuickTest must generate special integration files. By default, the option to generate this data is enabled. However, you, or someone else working on your QuickTest tests may have disabled the option in order to preserve disk space. Before you begin creating tests for use with LoadRunner or Mercury Application Management, enable this option as follows:



- 1** In QuickTest, choose **Tools > Options** or click the **Options** toolbar button. The Options dialog box opens and displays the General tab.
- 2** In the General tab, confirm that **Save data for integrating with Mercury performance testing and application management products** is selected.
- 3** If you want to integrate a test with LoadRunner or Mercury Application Management that was saved without this option, open and save the test again after selecting this option.

For more information on the Options dialog box, see Chapter 24, “Setting Global Testing Options.”

Tip: To check whether a test was saved with integration data, look for a **<testname>.usr** file in your test folder with the same **modified** date as the test.

Adding Statements for Checking Performance Testing and Application Management

You can use the **Services** object and its associated methods to insert statements that are specifically relevant to Performance Testing and Application Management. These include **Abort**, **GetEnvironmentAttribute**, **LogMessage**, **Rendezvous**, **SetTransactionStatus**, **ThinkTime**, **UserDataTable**, **StartTransaction** and **EndTransaction**. For more information on these methods, refer to your LoadRunner or Mercury Application Management 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 504.

Note: Your test must include transactions to be used by LoadRunner or the Business Process Monitor. LoadRunner and the 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 LoadRunner or the Business Process Monitor

The QuickTest tests you use with LoadRunner or the Business Process Monitor should be simple, designed to pinpoint specific operations, and should avoid using external actions and references to other external files.

Designing Tests for LoadRunner

Consider the following guidelines when designing tests for use with LoadRunner:

- LoadRunner cannot run nested action iterations.
- Do not include references to external actions or other external resources, such as an external Data Table file, environment variable file, shared object repositories, and so on.

Designing Tests for the Business Process Monitor

Consider the following guidelines when designing tests for use with the Business Process Monitor:

- Corresponding **StartTransaction** and **EndTransaction** statements must be contained within the same action.
- The Business Process Monitor does not use the iteration settings from the Run tab of the QuickTest Options dialog box. Instead, it uses the number of lines in the Data Table file.

Inserting and Running Tests in LoadRunner or Mercury Application Management

In addition to designing your test appropriately for use with LoadRunner or Mercury Application Management, there are a few issues you should be aware of when using your QuickTest test in LoadRunner or Mercury Application Management.

Inserting and Running Tests in a LoadRunner Scenario

When inserting and running tests in a LoadRunner scenario, consider the following guidelines:

- You can run only one GUI VUser concurrently per machine.
- To insert a QuickTest test in a LoadRunner scenario, browse to the test folder in the Controller Open Test dialog box and select **Astra Tests** in the **Files of type** box in order to view QuickTest tests in the folder.
- Ensure that QuickTest is closed on the QuickTest computer before running a QuickTest test from LoadRunner.
- In the Run-time Settings for script dialog box, only the **General** categories and sub-categories (**General**, **Iterations**, **Miscellaneous**, **Think Time**) are relevant for QuickTest tests. The **Replay** options are not relevant.

For more information on working with LoadRunner, refer to your LoadRunner documentation.

Inserting and Running Tests in the Business Process Monitor

When inserting and running tests in the Business Process Monitor, consider the following guidelines:

- The Business Process Monitor can run only one QuickTest test (transaction file) at a time.
- Transaction Breakdown is not supported for tests (transaction files) recorded with QuickTest.

For more information on working with Mercury Application Management, refer to your Mercury Application Management documentation.

Part IX

Appendix

A

Working with QuickTest—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:

- Recording and Running Tests
- Programming in the Expert View
- Working with Dynamic Content
- Advanced Web Issues
- Test Maintenance
- Testing Localized Applications
- Improving QuickTest Performance

Recording and Running Tests

- How does QuickTest capture user processes in Web pages?

QuickTest hooks the browser (Netscape, Microsoft Internet Explorer, or AOL). As the user navigates the Web-based application, QuickTest records the user actions. (For information about modifying which user actions are recorded, see Chapter 35, “Configuring Web Event Recording.”) QuickTest can then run the test by running the steps as they originally occurred.

- How can I record on objects or environments not supported by QuickTest?

You can do this in a number of ways:

- In addition to the core environments supported by QuickTest, many external add-ins are available for QuickTest Professional to support the environments you use in your application, such as 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 812.
- 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 16, “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 101.

Programming in the Expert View

- Can I store functions and subroutines in a function library?

You can define functions within an individual test, or you can create one or more external VBScript library files containing your functions, and then call them from any test.

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 test run, or you can register a new method for a test object class.

For more information, see Chapter 37, “Working with User-Defined Functions.”

Working with Dynamic Content

- How can I record and run tests on objects that change dynamically from viewing to viewing?

Sometimes the content of objects in a Web page or 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. For more information, see Chapter 4, “Managing Test Objects.”

- 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** method to check whether or not a window exists. For example:

```
Browser("Window_logical_name").Exist
```

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.

For additional information about the **Exist** and **ChildObjects** methods, refer to the *QuickTest Professional Object Model Reference*.

- How does QuickTest record on dynamically generated URLs and Web pages?

QuickTest actually clicks on 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.

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.

► 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?

Dynamically created Web pages utilizing Active Server Page technology have an .asp extension. This technology is completely server-side and has no bearing on QuickTest.

► 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 11, “Checking XML.”

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. When your application changes, you can rerecord part of a test. If the change is not significant, you can manually edit a test to update it.

You can also use QuickTest’s action feature to design more modular and efficient tests. While recording, you divide your test into several actions, based on functionality. When your application changes, you can rerecord 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 additional information, see Chapter 17, “Working with Actions.”

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. For more information, see Chapter 34, “Choosing the Object Repository Mode.”

To update the information in your checkpoints, the Active Screen, or about your test object properties when object properties change, or to add new objects or steps on an Active Screen image without rerecording steps, use the **Update Run** option. For more information, see “Updating a Test or Component” on page 519.

► **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 96.
- 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 tab of the Options dialog box is set to capture the amount of information you need and then:

- Perform an **Update Run** operation to save the required amount of information in the Active Screen for all existing steps.
- Re-record the step(s) containing the object(s) 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 will bring your application to the correct place in order to record the step.

For more information on changing the amount of information saved in the Active Screen for Windows applications, see “Setting Active Screen Options,” on page 618.

For more information on the **Update Run** options, see “Updating a Test or Component” on page 519.

For more information on setting breakpoints, see “Setting Breakpoints” on page 533.

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 additional information, see Chapter 12, “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 tab of the Test Settings dialog box.

For more information on working with Data Tables, see Chapter 18, “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 674.

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:

- Do not load unnecessary add-ins in the Add-in Manager when QuickTest starts. This will improve both recording time and test run performance. For more information about loading add-ins, see “Loading QuickTest Add-ins” on page 732.
- Run your tests in “Fast mode.” From the Run tab 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 tab of the Options dialog box, see “Setting Run Testing Options” on page 625.
- 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 tab of the Options dialog box. For more information, see “Setting Active Screen Options” on page 618.

- If you are testing Web applications, you can disable screen capture of all steps in the Active Screen. From the Active Screen tab 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 tab of the Options dialog box, see “Setting Active Screen Options” on page 618.
- 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.
- Decide when you want to capture and save images of the application for the test results. From the Run tab in the Options dialog box, select an option from the **Save step screen capture to test results** box. You can improve test run time and reduce disk space by saving screen captures only in certain situations or by not saving the images at all. For more information on the Active Screen tab of the Options dialog box, see “Setting Active Screen Options” on page 618.

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** option to recapture screens for all steps in your test. For more information, see “Updating a Test or Component” on page 519.

► **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 when you want to capture and save images of the application for the test results. From the Run tab in the Options dialog box, select an option from the **Save step screen capture to test results** box. You can reduce disk space and improve test run time by saving screen captures only in certain situations or not saving images at all. For more information on the Active Screen tab of the Options dialog box, see “Setting Active Screen Options” on page 618.
- 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 tab of the Options dialog box. For more information, see “Setting Active Screen Options” on page 618.
 - If you are testing Web applications, you can disable screen capture of all steps in the Active Screen. From the Active Screen tab, 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 tab of the Options dialog box, see “Setting Active Screen Options” on page 618.

- 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** option to recapture screens for all steps in your test. For more information, see “Updating a Test or Component” on page 519.

- **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 17, “Working with Actions.”

Index

Numerics

1_APP_ENV variable 711
1_DIR_ENV variable 711

A

accessibility. *See* Web content accessibility
action call
 iterations 377
 parameter values 379
 properties 375
 run properties 377
 setting run properties 377
action data sheets 345, 398
action Data Table parameters 221
Action List 21, 347
action parameters 204, 211–214, 364
 guidelines 394
 setting options 213
 storing output values 252, 262
Action tab, Data Table 345
Action toolbar 21, 347
ActionIteration, environment variable 228
actions 341–394
 adding to Keyword View 310
 creating 348
 diagram 342, 353
 dragging and dropping in hierarchy
 321
 external 344
 guidelines for working with 392
 inserting
 call to 357
 copy of 354
 existing 352
 moving in hierarchy 321
 multiple, in tests 344–345

actions (*continued*)
 nesting 360
 non-reusable 344
 overview 342–344
 parameterization data, storage
 location 374
 removing 385
 renaming 389
 reusable 344
 running from a step 517–518
 setting parameters 370
 setting properties 366
 sharing values 382
 using Dictionary objects 383
 splitting 362
 syntax 350
 template 391
Active Screen
 access, Test Settings dialog box
 Web tab 766
 advanced authentication 767
 changing 108
 defining capture settings 620
 defining Web settings 623
 increasing/decreasing saved
 information 1008
 saving and deleting files 96
Active Screen button 20
Active Screen dialog box 765, 767
Active Server Page technology 1006
ActiveX controls
 activating, syntax 785
 checking 783–784
 checkpoints and output values
 supported 780
 recording and running tests on
 781–782

- ActiveX controls (*continued*)
 - scripting methods 785
 - testing 779–785
 - Add Schema dialog box, XML checkpoint 200
 - Add Synchronization Point dialog box 500
 - Add/Remove dialog box, object
 - identification 793, 808
 - Add/Remove Properties dialog box 68, 70
 - Add-in License 732, 735
 - Add-in Manager dialog box 732
 - adding tests to version control 960
 - add-ins
 - associated and loaded 736
 - associating with a test or component 662–664
 - loading 732
 - tips 736
 - using 731–737
 - advanced authentication
 - Active Screen 767
 - Advanced Web Options dialog box 644
 - Advanced Windows Applications Options dialog box 631
 - Agent, Remote 969
 - Allow other Mercury tools to run tests 968
 - America Online (AOL) browser 743
 - analog recording 101, 103
 - analyzing test results 541
 - checkpoints 557
 - filtering results 549
 - output values 580
 - parameterized values 578
 - previewing results 556
 - printing results 555
 - run-time Data Table 581
 - Test Results window 542
 - API
 - using Windows 902
 - Application crash trigger 428
 - Application Details dialog box 707
 - application, sample xvi, 9
- applications
 - adding 707
 - closing 887
 - editing 707
 - running 887
 - specifying for a component 670
 - testing localized versions 1009
 - Applications dialog box
 - environment variables 672
 - ASCII 400
 - asp files 1006
 - Assignment column, Keyword View 308
 - assistive properties, configuring 792
 - associated and loaded add-ins 736
 - associated library files 908
 - with Quality Center 909
 - associating add-ins with a test or component 662–664
 - attribute property 900
 - authentication
 - for Active Screen 767
 - auto-expand VBScript syntax 717
 - automation
 - Application object 925
 - definition 920
 - development environment 923
 - language 923
 - object model 919
 - type library 923
 - automation script, generating for a test 660

B

- Back button 21
- backslash (\) 295
- Basic event recording configuration level 837
- behavior, DHTML 845
- Bitmap Checkpoint Properties dialog box 170
- bitmap checkpoints
 - analyzing results 562
 - creating 170–177
 - modifying 177–179

- bitmaps, checking 169–179
bookmarks 865
breakpoints 529–539
 deleting 534
 setting 533
 using in Keyword View 328
BROWSER_ENV variable 711
browsers, supported 742
bubbling 846
built-in environment variables 227–229
business analyst
 role in Business Process Testing 974
Business Component Settings dialog box
 Applications tab 670
 Environment tab 685–692
 Parameters tab 682
 Properties tab 660
 Recovery tab 694
 Resources tab 680
 Snapshot tab 669
 Web tab 692
business components. *See* components
Business Process Expert 975
Business Process Testing 973
 roles 974
 workflow 976
business process tests 976
 capturing a snapshot for a component 669
 running 979
- C**
- calculations, in the Expert View 890
Call to WinRunner Function dialog box 937
Call to WinRunner Test dialog box 933
calling TSL functions from QuickTest 937–941
CGI scripts 1006
character set support, Unicode 3, 86, 852
Check In command 960, 962
Check Out command 960
checking tests out of version control 960
- Checkpoint Properties dialog box
 Expected Data tab 145
 for checking databases 140–149
 for checking objects 126
 for checking tables 140–149
checkpoints
 accessibility options 620, 631, 632, 637, 644
 adding 116
 definition 93, 115
 for ActiveX controls 780
 for bitmaps 169–179
 for databases 133–149
 for images 130–132
 for objects 124–129
 for pages 747
 for tables 133–135, 140–149
 for text 151–168
 in Expert View 856
 modifying 130, 132
 parameterizing 238
 standard, for checking text 155
 supported for Web objects 740
text area 156
types 117
understanding 115
using formulas 414
Web content accessibility 760–763
XML 181–202
Close application process operation 438
Close method 887
CMDLINE_ENV variable 711
collection, properties. *See* programmatic descriptions
collections, of virtual objects 329
columns, displaying in Keyword View 323
COM 1006
command line options, deleting test results using 592
Comment column, Keyword View 308
comments
 in the Expert View 889
 in the Keyword View 498

Completing the Recovery Scenario Wizard
 screen 449

complex value 288

component iterations, specifying 661

component parameters 204, 211–214
 setting options 213
 storing output values 252, 262
 using 685

component settings. *See* Business Component Settings dialog box

components

- application details environment
 - variables 672
- creating 982
- debugging 529–539
- guidelines 981
- guidelines for templates 990
- opening 983
- pausing runs 532
- programming 465
- recording 991
- running 511–528, 993
- running from a step 517–518
- running using optional steps 525–526
- saving 985
- template guidelines 988
- templates 987
- updating 519
- using Data Table with 982

conditional statements 484

- using in Keyword View 327

configuration levels

- customizing 839–847
- standard 837–838

Configure Text Selection dialog box 161

Configure value area 286

configuring values 285

connecting to Quality Center 598, 946–950

connection string, specifying for database

- checkpoints 138

Constant Value Options button 288

Constant Value Options dialog box 288

constant value, defining 285

content property check, on databases 135–139

context menu, shortcut key 22

ControllerHostName, environment variable 228

conventions. *See* typographical conventions

cookies 1006

creating

- a new test 95
- test objects during a test run 66
- tests 85–114
- tests with locked resources 111

creation time identifier. *See* ordinal identifier

CreationTime property, understanding 800

currencies, setting custom format 407

Custom Active Screen Capture Settings dialog box 620

custom event-recording configuration 839–847

- adding listening events 843
- adding objects to the list 842
- deleting objects from the list 843
- procedure 839
- specifying listening criteria 845

custom number format, setting 407

custom objects, mapping 812

custom web event configuration files

- loading 849
- saving 849

Custom Web Event Recording Configuration dialog box 839

customer support, online xvi

customizing scripts 713

- general options 714
- highlighting script elements 717
- script window customization 715

D

Data Driver 241

Data menu commands, Data Table 406

data sheets

- action 398
- activating next/previous, shortcut key 22
- global 398
- global/action, choosing 345
- local 398

- Data Table 14, 19, 395–417
action data sheets 398
Action tab 345
changing focus, shortcut key 22
Data menu commands 406
data sheets 398
design-time 395
Edit menu commands 404
editing tables 400–407
File menu commands 403
Format menu commands 407
Global tab 345
importing data, in various formats 400
iteration options for individual tests 666
local data sheets 398
location 399
menu commands, for editing tables 403
parameters 215–221
run-time 396
saving 399–400
scripting functions, using 417
Sheet menu commands 404
specifications 402
storing output values 253, 264
table columns 216
table rows 216
using formulas 412–416
using with components 982
viewing results 581
with Quality Center 408
worksheet functions 412
- Data Table button 20
- Data Table parameters
setting options 218
- database checkpoints
analyzing results 560
modifying 149
specifying cell identification settings 147–148
specifying cells 144
specifying expected data 145–146
specifying value type 146–147
- database output values 250, 278
- Database Query wizard 136
database values, outputting 277
databases
checking 133–149
connection string 138
creating a query in ODBC / Microsoft Query 412
creating a query with Microsoft Query / SQL statement 139
creating checkpoints for 135–139
manually defining an SQL statement 136–139
result set 135
Specify SQL statement screen 138
data-driven test 204, 253
dates, setting custom format 407
Debug toolbar, QuickTest window 14, 21
Debug Viewer 535–537
 changing focus, shortcut key 22
 pane 19
Debug Viewer button 20
debugging tests and components 529–539
 deleting breakpoints 534
 example 538
 pausing runs 532
 setting breakpoints 533
decreasing Active Screen information 1008
default object identification settings 802
default optional steps 526
default parameter definition 287, 290
default properties, modifying 33–48, 49–81
defects, reporting 597
 automatically during test 602
 from Test Results 598
- deleting
 actions 385
 breakpoints 534
 objects from list 843
 objects from the object repository 80
 test results 590
- description, test objects 37
 modifying 67–71
 See also objects
- descriptive programming. *See* programmatic descriptions
- design-time Data Table 395

Dictionary object 383
Dim statement, in the Expert View 875
disconnecting from Quality Center 949
disk space, saving 1010
Display selected action button 21
Do...Loop statement, in the Expert View 893
documentation
 online xv
 printed xv
 updates xvii
documentation, printed
 Installation Guide xv
 Shortcut Key Reference Card xv
 Tutorial xv
Domain command line option 593
DOS commands, run within tests 901
dynamic Web content 1005
dynamically generated URLs and Web pages
 1005

E

Edit menu commands, Data Table 404
Edit Schema dialog box, XML checkpoint
 200
Element Value dialog box, XML checkpoint
 196
embedded Web browser controls 743
encoding passwords 416
End Transaction button 21
End Transaction dialog box 508
ending transactions 508
environment variables 222–232, 685–692
 application details
 predefined variable names 711
 understanding 709
 application details for components
 672
 built-in 227–229
 files, with Quality Center 226
 storing output values 254, 266
 types 222
 user-defined external 224
 user-defined internal 222
 user-defined, exporting 691

environment variables (*continued*)
 user-defined, modifying 689
 user-defined, viewing 689
environments, specifying for a component
 671
error behavior options for tests 666
errors in VBScript syntax 877
event-recording configuration 835–850
 customizing levels 839
 resetting 850
 standard levels 837
Excel formulas
 for parameterizing values 413
 in checkpoints 414
 in the Data Table 412–416
Excel. *See Microsoft Excel*
ExecuteFile function 910
EXEPATH_ENV variable 711
Exist function 1005
Exist statement 503
existing actions, inserting 352
Expert View 17, 851–905, 1004
 checkpoints 856
 closing applications 887
 customizing appearance of 713
 finding text 867
 replacing text 869
 running applications 887
 understanding 853
 understanding parameters 857
expressions, using in the Expert View 871
eXtensible Markup Language (XML) 1007
external action
 data location 374
 definition 344
external functions, executing from script 910
external user-defined environment variables
 224

F

FAQs 1003–1013
File menu commands, Data Table 403
File toolbar, QuickTest window 14, 20
Filter Images Check dialog box 754, 757
Filter Links Check dialog box 754, 756

filter properties 803
 filtering
 hypertext links 756
 image sources 757
 finding text in Expert View 867
 For...Each statement, in the Expert View 892
 For...Next statement, in the Expert View 891
 Format menu commands, Data Table 407
 formulas
 for parameterizing values 413
 in checkpoints 414
 in the Data Table 412–416
 frequently asked questions 1003–1013
 FromDate command line option 593
 function arguments, passing parameters
 from QuickTest to WinRunner 940
 Function call operation 438
 function libraries. *See* associated library files
 functions, user-defined 907–917

G

general options 714
 Generate Script option 926
 GetROProperty method 898
 global data sheet 345, 398
 global Data Table parameter 220
 global/action data sheets, choosing 345
 Go To dialog box 864
 GroupName, environment variable 228
 guidelines
 for business component templates 990
 for component templates 988
 for components 981

H

handler 845
 Help, online, from within QuickTest Professional xv
 hierarchy
 moving actions and steps 321
 High event recording configuration level 837

HTML Source dialog box 753
 HTML Tags dialog box 753
 HTML verification 753–754
 hypertext links, filtering 756

I

identifying test objects 33–48
 If...Then...Else statement, in Expert View 895
 Image Checkpoint Properties dialog box 130
 image checkpoints
 comparing image contents 131
 editing the property value 131
 image sources, filtering, for page checkpoints 757
 image, capturing for a component 669
 images, checking 130–132
 increasing Active Screen information 1008
 index identifier. *See* ordinal identifier
 Index property, programmatic descriptions 886
 Index property, understanding 798
 initialization scripts 921
 Insert Checkpoint button 21
 Insert New Action dialog box 349
 Insert Report dialog box
 inserting transactions 507
 Installation Guide, QuickTest Professional xv
 IntelliSense 716, 858
 internal user-defined environment variables 222
 Internet Explorer 743
 Item column, Keyword View 307
 iterations 215, 377
 options for individual tests 666
 specifying for components 661

K

key assignments, in Expert View 719
 key column 148
 keyboard keys, in Keyword View 308
 Keyboard or mouse operation 438
 keyboard shortcuts
 in Expert View 719

Keyword View 16, 303
adding steps after block 313
adding steps to 310
columns 306
deleting steps from 322
modifying steps 314
setting display options 323
using keyboard keys in 308
keywords, selecting 863

L

language support, Unicode 3, 86, 852
library files
 associated 908
 specifying for a component 680
 specifying for a test 674
license information 9
loaded and associated add-ins 736
loading QuickTest add-ins 732
local data sheet. *See* action data sheets
local test 344
LocalHostName, environment variable 228
localization 222, 399
localized applications, testing 1009
location identifier. *See* ordinal identifier
Location property, understanding 799
locked resources 109
 creating tests 111
 opening tests 112
 saving tests 114
Log command line option 593
loop statements 488
 using in Keyword View 327
low-level recording 101, 107, 1004
Low-Level Recording button 21

M

managing test objects 49–81
managing tests 95–99
 creating new 95
 opening 95
 printing 99
 saving 96
 testing process 8, 943

managing tests (*continued*)
 unzipping 98
 zipping 98
mandatory properties, configuring 792
mapping custom objects 812
mathematical formulas, in the Data Table 412–416
measuring transactions 504
Medium event recording configuration level 837
menu bar, QuickTest window 14
Mercury Quality Center. *See* Quality Center
Mercury Tours, sample application xvi, 9
meta tags 1006
methods
 adding new or changing behavior of 912
 run-time objects 899
 user-defined 912
 viewing test objects 33–48
Microsoft Excel 400, 412
Microsoft Internet Explorer 742
Microsoft Query
 choosing a database for a database
 checkpoint 139, 412
 database check 136–139
Microsoft Visual Basic scripting language 8
MinSize command line option 594
modifying
 default properties 33–48, 49–81
 test object descriptions 67–71
 test object properties during a run 66
 your license 9
multiple actions in tests 344–345

N

Name and Description screen 448
Name command line option 594
nesting actions 360
Netscape 743
New Action button 21
New button 20
non-reusable action 344

O

object identification
 generating automation scripts 803
 restoring defaults 802
 Object Identification dialog box 791
 Object Mapping dialog box 812
 object model
 automation 919
 definition 920
 object names, modifying 65
 Object Properties dialog box 57, 60
 Object property, run-time methods 900
 object repository
 adding an object 73
 deleting an object 80
 finding an object property or value 63
 replacing an object property value 64
 Object Repository button 20
 Object Repository dialog box 51
 object repository mode
 choosing 815–833
 per-action 818
 setting for tests 674, 828
 setting the default 828
 shared 822
 with Quality Center 828
 Object Spy 46
 Object Spy button 20
 Object state trigger 428
 objects
 checking 124–129
 descriptions, modifying 67–71
 identification 789–814
 identifying 33–48
 methods, run-time 899
 names, modifying 65
 properties
 adding 71
 run-time 899
 viewing methods 33–48
 ODBC, choosing a database for a database
 checkpoint 412
 Open button 20
 Open dialog box 95, 983
 Open QuickTest Test dialog box 90, 95, 983

Open Test from Quality Center Project dialog box 953, 956
 opening tests 95
 in a Quality Center project 952
 with locked resources 112
 Operation column, Keyword View 308
 optional steps 525–526
 default 526
 setting 525
 Options button 20
 Options dialog box 612
 Active Screen tab 618
 Folders tab 616
 General tab 614
 Generate Script option 926
 generating automation scripts 614
 Run tab 625
 Web tab 639
 Windows Applications tab 628
 ordinal identifier 796
 OS, environment variable 228
 OSVersion, environment variable 228
 output types 261
 action parameters 262
 component parameters 262
 Data Table 264
 environment variables 266
 test parameters 262
 output value categories
 database output values 250
 standard output values 249
 text area output values 249
 text output values 249
 XML output values 250
 Output Value Properties dialog box 258
 output values
 ActiveX controls 780
 creating for object properties 255
 creating for text 270
 database 277, 279
 definition 247
 editing 254
 for database cells, creating 277
 for tables and databases 278
 for XML elements/attributes, creating 280

output values (*continued*)

- standard 255
- storing in action, component, or test parameters 252
- storing in Data Table 253
- storing in environment variables 254
- text 268–276
- text area 270
- viewing 254
- viewing results 580
- XML, specifying 280

outputting

- database values 277
- property values 255
- text values 270
- values 247–284
- XML values 280

P

Page and Frame Options dialog box 641

Page Checkpoint Properties dialog box 748, 750

page checkpoints

- editing page property values 752
- filtering hypertext links 756–757
- filtering image sources 757–759
- HTML verification 753–754

pages, checking 747

panes

- changing focus, shortcut key 22
- Debug Viewer 19
- Test 16

parameter definition, default 287, 290

Parameter Options button 288

Parameter Options dialog box 213

Parameter reserved object 685

parameter types

- action parameters 204
- component parameters 204
- Data Table parameters 215–221
- environment variable parameters 222–232
- random number parameters 233–234
- test parameters 204

parameter value, defining 285

parameter values for action calls 379

parameterization example 235

parameterization icon 209, 289

parameterized values, viewing in test results 578

parameterizing

- methods 205
- tests, example 235
- using the Data Driver 241
- values 203–240

parameters

- action 364
- action guidelines 394
- environment variables, user-defined 687–692
- in the Expert View 857
- setting for actions 370
- specifying for tests and components 682

passing data between actions 345

passing parameters

- to a WinRunner function 940
- to a WinRunner test 935

Password command line option 595

Password Encoder dialog box 416

password, encoding 416

PathFinder.Locate, statement 617

pausing run sessions 532

percentages, setting custom format 407

performance, improving 1010

planning tests 87–88

Pop-up window trigger 428

post-recovery test run options 419

Post-Recovery Test Run Options screen 445

power users, advanced features for 1003–1013

previewing test results 556

Print button 20

printing

- test results 555
- tests and components 99

priority

- setting for recovery scenarios 459

ProductDir, environment variable 228

ProductName, environment variable 228

ProductVer, environment variable 228

- programmatic descriptions 66, 878–887
for description objects 883
for WebElement objects 886
in statement 880
 using the With statement 882
using the Index property 886
using variables 880
- programming 1004
 adding steps 465
 comments 498
 conditional statements 484
 in Expert View 851–905
 in Keyword View 465
 in VBScript 873
 loop statements 488
 sending messages to test results 496
 Step Generator 466, 467–484
- project (Quality Center)
 connecting to 946–950
 disconnecting from 949
 opening tests in 952
 saving tests to 951–952
- Project command line option 595
- properties
 adding test object properties 71
 CreationTime 800
 default 33–48, 49–81
 Index 798
 Location 799
 run-time objects 899
 setting for action calls 375
 setting for actions 366
 viewing for recovery scenarios 452,
 459
 viewing for steps in Keyword View
 326
- property collection. *See* programmatic descriptions
- property output values, specifying ??–260
- property values
 outputting 255–??, 255
 waiting for 499
- Q**
- QA engineer
 role in Business Process Testing 974
- QA Tester 975
 role in Business Process Testing 974
- Quality Center 943–971
 associated library files 909
 capturing a snapshot for a component
 669
 connecting to project 598, 946–950
 Connectivity Add-in 950
 Data Table 408
 disconnecting from 949
 environment variable files 226
 managing the testing process 8
 opening tests in 952
 reporting defaults 597
 reporting defaults automatically 602
 reporting defaults manually 598
 running QuickTest tests remotely 968
 saving tests to a project 951–952
 shared object repository 828
 using QuickTest with 8
 version control for 959
- Quality Center Connection button 20
- Quality Center Connection dialog box 599,
 947
- query file, for a database checkpoint
 creating 139, 412
 working with ODBC / Microsoft
 Query 412
- QuickTest
 automation object model 919
 introduction 3–8
 overview 11–30
 window 14
 Action toolbar 14, 21
 Data Table 14
 Debug toolbar 14, 21
 File toolbar 14, 20
 menu bar 14
 status bar 14
 Test pane 14
 Testing toolbar 14, 21
 title bar 14

QuickTest Automation Object Model
Reference 927

R

random number parameters 233–234
Readme xv
Readme, QuickTest Professional xv
Record and Run Settings dialog box 700
 environment variables 709
 Web tab 703
 Windows Applications tab 705
Record button 21
recording
 analog 101
 components 991
 low-level 101, 1004
 on Web sites 739
 status, options 846
 tests 88–92, 699–711
 time, improving 1010
recovery
 associating scenarios with tests 457
 copying scenarios 455
 deleting scenarios 454
 disabling scenarios 460
 files 423
 modifying scenarios 454
 operations 419
 removing scenarios from tests 460
 saving scenarios 450
 scenarios 419
 setting default scenarios 462
 setting scenario priority 459
 viewing scenario properties 452, 459
recovery operation
 Close application process 438
 Function call 438
 Keyboard or mouse operation 438
 Restart Microsoft Windows 438
Recovery Operation - Click Button or Press Key screen 440
Recovery Operation - Close Processes screen 442
Recovery Operation - Function screen 443
Recovery Operation screen 438

Recovery Operations screen 437
Recovery Scenario Manager Dialog Box 423
Recovery Scenario Wizard 426
 Click Button or Press Key screen 440
 Close Processes screen 442
 Completing the Recovery Scenario Wizard screen 449
 Function screen 443
 Name and Description screen 448
Post-Recovery Test Run Options screen 445
Recovery Operation screen 438
Recovery Operations screen 437
Select Object screen 431
Select Processes screen 435
Select Test Run Error screen 434
Select Trigger Event screen 428
Set Object Properties and Values screen 433
Specify Pop-up Window Conditions screen 430
Recursive command line option 595
redirection of server 1006
registering methods 912
 guidelines for 917
 using the RegisterUserFunc statement 914
RegisterUserFunc statement 912
regular expressions 290–301
 backslash (\) 295
 defining 293
 for constants 287
 for property values 291
 in checkpoints 292
 using in the Expert View 871
remote access to QuickTest 968
Remote Agent 969
removing actions 385
renaming actions 389
replacing text in Expert View 869
report. *See* Test Results window
reporting defects
 automatically 597
 manually 598
reports, filter 904
reserved objects 908

- Restart Microsoft Windows operation 438
result set 135
ResultDir, environment variable 228
results
 viewing 541
results display, customizing 606
Results Remover Utility
 running from the command line 592
results schema 606
reusable actions 344
roles in Business Process Testing 974
Run button 21
Run dialog box 513
run options, in the Options dialog box 625
run properties, setting for action calls 377
running
 components 993
 single actions 343
running tests 699–711
 from a Quality Center project 957
 on Web sites 739
 viewing results 546
 WinRunner tests 932–936
running tests and components 511–528
 from a step 517–518
 Run dialog box 513
 to update expected results 519
 Update Run dialog box 521
 using optional steps 525–526
run-time
 Data Table 396
 viewing results 581
 objects 899
 settings, adding and removing 727
- S**
- sample application, Mercury Tours xvi, 9
Save button 20
Save dialog box 96
Save QuickTest Test dialog box 92, 96
Save Test to Quality Center Project dialog box 951
saving
 recovery scenarios 450
 tests 96
 to a Quality Center project 951–952
 tests with locked resources 114
ScenarioId, environment variable 228
scenarios
 associating with tests 457
 copying recovery 455
 deleting recovery 454
 disabling recovery 460
 modifying recovery 454
 recovery 419
 removing recovery from tests 460
 saving recovery 450
 setting default recovery 462
 setting recovery priority 459
 viewing recovery properties 452, 459
Schema Validation dialog box, XML
 checkpoint 197
schema, for results 606
screen shot. *See* snapshot
scripts
 customizing 713
 general options 714
 highlighting script elements 717
 script window customization 715
Section 508, Web Content Accessibility Guidelines 86, 760
Select a Keyword dialog box 863
Select Action dialog box 355, 358
Select Object screen 431
Select Processes screen 435
Select Test Run Error screen 434
Select Trigger Event screen 428
server
 Quality Center, disconnecting from 949
 redirections 1006
 server-side connections 1006
Server command line option 595
session IDs 1006
Set Object Properties and Values screen 433
Set statement, in the Expert View 874

Setting object 724
SetTOProperty method 66
SGML 1007
shared object repository 815, 822
 choosing a file 831
shared object repository mode
 with Quality Center 828
sharing action values
 using Dictionary objects 383
 using environment variables 383
 via the global Data Table 382
Sheet menu commands, Data Table 404
Shortcut Key Reference Card xv
Shortcut Key Reference Card, QuickTest Professional xv
shortcut keys, in Keyword View 308
shortcuts
 for menu items 22
 in Expert View 719
Silent command line option 596
Smart Identification
 analyzing information 587
 configuring 803
 disabling during test runs 667
 enabling from the Object Identification dialog box 801, 803
Smart Identification Properties dialog box 808
snapshot, capturing for a component 669
Specifications for Data Table 402
Specify Pop-up Window Conditions screen 430
Specify SQL statement screen, for creating database checkpoints 138
Split Action button 21
Split Action dialog box 363
splitting actions 362
Spy. *See* Object Spy
standard checkpoints
 analyzing results 559
 specifying timeout 129, 755
standard event-recording configuration 837–838
standard output values 249
 creating 255
 specifying 258
Start Transaction button 21
Start Transaction dialog box 507
starting QuickTest 12
statement completion 716, 858
statements, using in Keyword View 314
status bar, QuickTest window 14
Step commands 530
Step Generator 466, 467–484
steps
 adding after block 313
 adding to Keyword View 310
 deleting from Keyword View 322
 dragging and dropping in hierarchy 321
 inserting 467–484
 modifying in Keyword View 314
 moving in hierarchy 321
 viewing properties in Keyword View 326
steps, optional 525–526
Stop button 21
Subject Matter Expert (SME) 974
Summary column, Keyword View 308
support, online xvi
synchronization points
 creating 499
 inserting 500
synchronization timeout
 setting 666
synchronizing tests 499
 modifying timeout values 503
 synchronization point 499
 waiting for objects to appear 503
 waiting for specified property values 499
syntax errors, VBScript 877
SystemTempDir, environment variable 228
SystemUtil.Run method 887

T

table checkpoints
 analyzing results 560
 modifying 149
 specifying cell identification settings 147–148
 specifying cells 144
 specifying expected data 145–146
 specifying value type 146–147

Table Output Value Properties dialog box 278

tables, checking 133–135, 140–149

technical support. *See* customer support

templates
 for actions 391
 for components 987
 guidelines for business components 990
 guidelines for components 988

test batches, running 527–528

Test command line option 596

test database, maintaining 921

test flow 347

test objects
 identifying 33–48
 managing 49–81
 modifying names of 65
 property values, retrieving and setting 898

Test pane 14, 16
 changing focus, shortcut key 22
 Expert View tab 17
 Keyword View tab 16

test parameters 204, 211–214
 setting options 213
 storing output values 252, 262
 using 685

test results
 deleting with command line options 592
 deleting with Test Results Deletion Tool 590
 enabling and filtering 904
 output values 580
 parameterized values 578

test results (*continued*)

reporting defects 597
 reporting defects automatically 602
 reporting defects manually 598
 run-time Data Table 581
 sending messages to 496
 viewing 541
 viewing for any test 551–553
 viewing WinRunner steps 603

Test Results button 20

Test Results Deletion Tool 590

test results display, customizing 606

Test Results toolbar, Test Results window 546

Test Results tree 545

Test Results window 542
 Test Results toolbar 546
 test results tree 545

Test run error trigger 428

test run time, improving 1010

test scripts. *See* scripts

test set 958

Test Settings button 20

Test Settings dialog box 654
 Environment tab 685–692
 Generate Script option 926
 Parameters tab 682
 Properties tab 659
 Recovery tab 694
 Resources tab 674
 Run tab 665
 Web tab 692

Test toolbar, QuickTest window 14, 21

test versions in QuickTest 959

TestDir, environment variable 228

TestDirector. *See* Quality Center

testing in Expert View 851–905

testing options
 restoring 726
 retrieving 725
 run-time 727
 setting 724
 setting for all tests 611–650
 setting for an individual test or component 651
 within a test script 723–727

- testing process 5
 - analyzing test results 7
 - creating tests 5
 - running tests 7
- Testing Tool Specialist 975
- TestIteration, environment variable 228
- TestName, environment variable 228
- tests
 - associating recovery scenarios with 457
 - checking databases 133–139
 - debugging 529–539
 - deleting results 590
 - diagram 342, 353
 - disabling recovery scenarios 460
 - enhancing 93
 - local 344
 - managing 95–99
 - opening 95
 - parameterizing, example 235
 - pausing runs 532
 - planning 87–88
 - previewing results 556
 - printing 99
 - printing results 555
 - programming 465
 - recording 88–92, 699–711
 - removing recovery scenarios from 460
 - running 511–528, 699–711
 - running from a step 517–518
 - running using optional steps 525–526
 - saving 96
 - saving to a Quality Center project 951–952
 - setting default recovery scenarios for 462
 - test results 541
 - unzipping 98
 - updating 519
 - zipping 98
- text
 - finding in Expert View 867
 - replacing in Expert View 869
- Text Area Checkpoint Properties dialog box 159
- text area checkpoints, analyzing results 563
- Text Area Output Value Properties dialog box 272
- text area output values 249
 - creating 270
 - specifying 272
- Text Checkpoint Properties dialog box 159
- text checkpoints
 - analyzing results 563
 - configuring the text selection 161
 - modifying 168
 - specifying the checked text 163
 - specifying the text after 166–167
 - specifying the text before 164–165
 - specifying timeout 167
 - types 151
- TEXT function in Data Table worksheet 412
- Text Output Value Properties dialog box 272
- text output values 249
 - specifying 272
- text values, outputting 270
- text, checking 151–168
 - using standard checkpoints 155
 - using text area checkpoints 156
 - using text checkpoints 153
- See also* text checkpoints
- timeout
 - setting 666
 - specifying for standard checkpoint 129, 755
 - specifying for text checkpoints 167
- times, setting custom format 407
- timing transactions 504
- title bar, QuickTest window 14
- toolbars, QuickTest window
 - Action 21
 - Debug 14, 21
 - File 14, 20
 - Testing 14, 21
- transactions 504
 - defining 504
 - ending 508
 - inserting 507
 - measuring 504
- Tree View. *See* Keyword View

trigger

- Application crash 428
- events 419
- Object state 428
- Pop-up window 428
- test run error 428

TSL functions, calling from QuickTest
937–941

Tutorial xv

Tutorial, QuickTest Professional xv
typographical conventions xviii

U

Unicode 3, 86, 852

unregistering methods, using the
 UnregisterUserFunc statement 915

UnregisterUserFunc statement 912

UntilDate command line option 597

unzipping tests 98

Update Run dialog box 521

updates, documentation xvii

updating tests and components 519

UpdatingActiveScreen, environment
 variable 228

UpdatingCheckpoints, environment variable
 229

URL_ENV variable 711

User command line option 597

User's Guide xv

user-defined

- functions 907–917
- methods 912
- properties, accessing 900
- test objects, mapping 812

UserName, environment variable 229

V

Value column, Keyword View 308

Value Configuration Options dialog box
 209, 289

VALUE function in Data Table worksheet
 412

values

- configuring 285
 - outputting 247–284
 - parameterizing 203–240
- variables

environment 685–692

See also environment variables, user-defined

VBScript

- auto-expand syntax 717
- documentation 889
- library files 908
- syntax 873
- syntax errors 877

version control

- adding tests to 960
- checking tests in to 960, 962
- checking tests out of 960

version manager 959

viewing test results 541

- checkpoints 557
- filtering results 549
- for any test 551–553
- previewing test results 556
- printing test results 555
- Test Results window 542

Viewlink objects 739

Virtual Object Manager 338

Virtual Object wizard 334–337

virtual objects 329–339

- defining 333–337
- removing 338–339

Visual Basic objects 773–778

- checking 776
- recording and running tests on 775
- using objects and methods 778

VuserId, environment variable 229**W**

W3C Web Content Accessibility Guidelines
 86, 760

Wait statement 503

waiting for objects 499

Index

- WaitProperty statement 499
 - Web browsers, supported 739
 - Web content accessibility 86, 760
 - checkpoints
 - automatically adding 760
 - in test results 763
 - manually adding 761–763
 - setting preferences 760
 - checkpoints in test results 573
 - Web content, dynamic 1005
 - Web Event Recording Configuration dialog box 838
 - Web Page Appearance dialog box 623
 - web page/frame XML checkpoint 183
 - Web settings
 - Advanced Web Options dialog box 644
 - Options dialog box 639
 - Page and Frame Options dialog box 641
 - Web site, Mercury xvii
 - Web sites, recording and running tests 739
 - Web tab, Record and Run Settings dialog box 703
 - WebElement objects, programmatic descriptions 886
 - Web-event-recording configuration 835–850
 - customizing 839–847
 - standard 837–838
 - What's New in QuickTest Professional xv
 - While statement, in the Expert View 894
 - Windows API 902
 - Windows applications
 - adding 707
 - editing 707
 - settings 628
 - Windows Applications settings, Advanced Web Options dialog box 631
 - Windows Applications tab, Record and Run Settings dialog box 705
 - Windows command line options 592
-
- WinRunner
 - calling tests from QuickTest 932–936
 - calling TSL functions from QuickTest 937–941
 - function arguments, passing parameters from QuickTest 940
 - tests, passing parameters from QuickTest 935
 - viewing WinRunner steps in test results 603
 - working with 931–941
 - With Generation Results window 494
 - With statements 491–496
 - "With" Generation Results window 494
 - entering manually 896
 - generating automatically, while recording 492
 - generating for existing actions 494
 - in the Expert View 491
 - removing 495
 - WORKDIR_ENV variable 711
 - workflow in Business Process Testing 976
 - working test 960
 - worksheet functions in the Data Table 412

X

- XML
 - checking 181–202
 - checkpoint results, attribute details 567
 - checkpoint results, checkpoint summary 566
- checkpoints
 - Add Schema dialog box 200
 - analyzing results 201, 564
 - Edit Schema dialog box 200
 - Element Value dialog box 196
 - elements 193
 - for files 187
 - for web page/frame 183
 - modifying 201
 - Schema Validation dialog box 197
 - values 194

XML (*continued*)

- objects and methods 202
- output value results, analyzing 582
- output value results, attribute details
 - 585
- XML Checkpoint from File dialog box 187
- XML Checkpoint Properties dialog box 190
- XML Checkpoint Results window 565
- XML Output Value Properties dialog box 282
- XML Output Value Results window 584
- XML output values 250
- XML values, outputting 280

Z

- zipping tests 98

Index

Introduction to QuickTest Professional 8.0

Student Handbook

Introduction to QuickTest Professional 8.0 Self-Paced Training

© Copyright 1994 - 2004 by Mercury Corporation.

This workbook, and the accompanying software and other documentation, is protected by U.S. and international copyright laws, and may be used only in accordance with the accompanying license agreement. Features of the software, and of other products and services of Mercury Corporation, may be covered by one or more of the following patents: U.S. Patent Nos. 5,701,139; 5,657,438; 5,511,185; 5,870,559; 5,958,008; 5,974,572; 6,138,157; 6,144,962; 6,205,122; 6,237,006; 6,341,310; 6,360,332; 6,470,383; 6,449,739; 6,477,483; 6,549,944; 6,560,564; 6,564,342 and 6,587,969. Other patents pending. All rights reserved.

Mercury Interactive, the Mercury Interactive logo, WinRunner, XRunner, FastTrack, LoadRunner, LoadRunner TestCenter, Test-Director, TestSuite, WebTest, Astra, Astra SiteManager, Astra SiteTest, Astra QuickTest, QuickTest, Open Test Architecture, POPs on Demand, TestRunner, Topaz, Topaz ActiveAgent, Topaz Delta, Topaz Observer, Topaz Prism, Twinlook, ActiveTest, Active-Watch, SiteRunner, Freshwater Software, SiteScope, SiteSeer and Global SiteReliance are registered trademarks of Mercury Interactive Corporation or its wholly-owned subsidiaries, Freshwater Software, Inc. and Mercury Interactive (Israel) Ltd. in the United States and/or other countries.

ActionTracker, ActiveScreen, ActiveTune, ActiveTest SecureCheck, Astra FastTrack, Astra LoadTest, Change Viewer, Conduct, ContentCheck, Dynamic Scan, FastScan, LinkDoctor, ProTune, RapidTest, SiteReliance, TestCenter, Topaz AIMS, Topaz Console, Topaz Diagnostics, Topaz Open DataSource, Topaz Rent-a-POP, Topaz WeatherMap, TurboLoad, Visual Testing, Visual Web Display and WebTrace are trademarks of Mercury Interactive Corporation or its wholly-owned subsidiaries, Freshwater Software, Inc. and Mercury Interactive (Israel) Ltd. in the United States and/or other countries. The absence of a trademark from this list does not constitute a waiver of Mercury Interactive's intellectual property rights concerning that trademark.

All other company, brand and product names are registered trademarks or trademarks of their respective holders. Mercury Interactive Corporation disclaims any responsibility for specifying which marks are owned by which companies or which organizations.

Mercury

379 N. Whisman Avenue Mountain View, CA 94089 USA

Tel: (650) 603-4300

© 1994 - 2004 Mercury, All rights reserved

QTP80INT-CBT-STUDENT-01A

OVERVIEW

The Introduction to QuickTest Professional 8.0 Self-Paced Training focuses on the basic features and functions used in creating automated tests using QuickTest Professional.

This course is intended for all new users of QuickTest Professional, those who have had limited use of the tool from tutorials and testers who have no programming background.

Upon completion of this training you should be able to demonstrate a basic level of product proficiency.

QUICKTEST PROFESSIONAL Self-Paced Training

Intended Audience: Non-technical and new users of QuickTest, who need to automate manual testing and verification in a short amount of time.

Prerequisites:

- Knowledge of HTML
- Working knowledge of Windows applications, Web sites, browsers and testing

Software Prerequisites:

- QuickTest Professional

Planning Your Test	Chapter 2
Exercise: Evaluating a Test Case.....	2-2
Test Case 1 Review.....	2-3
Test Case 1 Review Answers.....	2-4
Exercise: Learning the AUT	2-5
Recording and Running a Test.....	Chapter 3
Exercise: Record and Playback.....	3-2
Test Results Review.....	3-8
Test Results Review Answers.....	3-9
Understanding Objects in QuickTest	Chapter 4
Understanding Objects Review	4-2
Understanding Objects Review Answers	4-3
Exercise: Identifying Objects.....	4-4
Test Objects Review	4-7
Test Objects Review Answers	4-8
Exercise: Changing the Object Name	4-9
Creating Synchronization Steps	Chapter 5
Synchronization Review	5-2
Synchronization Review Answers	5-4
Exercise: Analyzing a Failed Test	5-5
Exercise: Inserting a Synchronization Point.....	5-7
Using Standard Checkpoints	Chapter 6
Checkpoints Review	6-2
Checkpoints Review Answers	6-3
Exercise: Add a Standard Checkpoint	6-4
Test Results Review.....	6-6
Test Results Review Answers.....	6-7
Adding Parameters	Chapter 7
Parameter Types Review	7-2

Parameter Types Review Answers	7-4
Exercise: Create and Run a Test.....	7-5
Exercise: Create an Output Parameter	7-9
Recovery Manager.....	Chapter 8
Recovery Manager Review.....	8-2
Recovery Manager Review Answers.....	8-3
Exercise: Parameterized Test.....	8-4
Exercise: Recovery Scenario	8-6

1

Planning Your Test

Objectives

This chapter contains two exercises:

- Evaluating a Test Case
- Learning the Application Under Test

Exercise: Evaluating a Test Case

In this exercise you will test your understanding of a useful test case. First, read Test Case 1, then answer the questions in Test Case 1 Review.

Test Case 1

Test Name: InsertOrder
Test Objective: Create a new order with the Flight Reservation application.
Description: Test the functionality of the process to create a new flight reservation.
Application: \QTP\samples\flight\app\flight4a.exe
Windows Versions: Windows 2000 or Windows NT
Test Requirements:

1. Verify that "Insert Done..." appears in the status bar of the Flight Reservation window.
 2. Verify that an order number is displayed in the Order No box of the Flight Reservation window.
 3. Verify that the application accepts different combinations of from and to cities.
 4. Verify the state of the check boxes in the Open Order window.
 - Initial conditions: all enabled.
 - When Customer Name or Flight Date is checked: Order No. is disabled; the other 2 are enabled.
 - When Order No. is checked: Order No. is enable; the other 2 are disabled.
 5. Verify that "10" is the maximum number of tickets accepted for a flight reservation.
 6. After clearing the window of all data, verify that the order just created can be opened and displayed in the Flight Reservation window.
-

Test Case 1 Review

1. These are some characteristics of a useful test case. Are these characteristics included in Test Case 1? If yes, fill in the answers; if no, write “not included”.

Purpose _____

Navigation _____

Verification _____

Sample Data _____

2. List one example of a test objective and two examples of a test requirement.

3. What is the importance of learning the application under test before creating automated test scripts?

Test Case 1 Review Answers

1. Are these characteristics included in the sample Test Case?

Purpose: Test objective stated.

Navigation: User actions are not included.

Verification: Test requirements listed.

Sample Data: Sample data was not included.

2. List one example of a test objective and two examples of a test requirement.

Objective: Create a new order with the flight reservation application.

Requirement: Verify that insert done appears in the status bar of the flight reservation window.

Requirement: Verify that an order number is displayed in the order number box on the flight reservation window.

3. You can record a basic test with the correct user actions in a short amount of time.

Exercise: Learning the AUT

Before you can create an effective automated test, you need to learn the application under test: Mercury Flight Reservations.

In this exercise, you will launch Mercury Flight Reservations and perform a simple exercise. Taking the time to do this now will help you feel comfortable creating automated tests in subsequent chapters.

Part 1: Create a New Order

1. From the START menu, select **PROGRAMS > QUICKTEST PROFESSIONAL > SAMPLE APPLICATIONS > FLIGHT**. The login screen appears.
2. Log in using the **AGENT NAME: jojo** with a **PASSWORD: mercury**.
3. Click **OK**. The application will launch the **FLIGHT RESERVATION** window as shown in Figure 1-1.

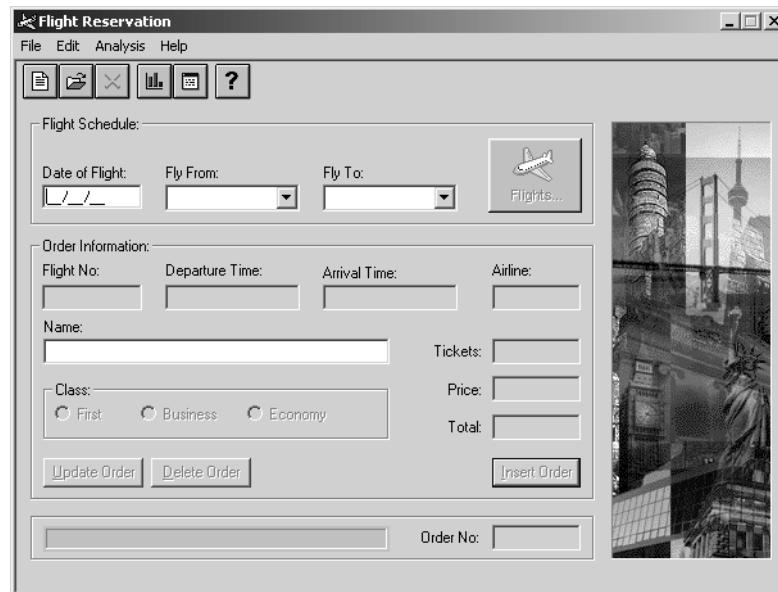


Figure 1-1

4. Enter the following data in the flight reservation fields: date of travel 11/01/10, flying from Paris to San Francisco.
5. Click the **FLIGHTS** button to display the flight schedule.
6. The first flight on the schedule is highlighted. Click **OK** to accept this flight and close the flight schedule window.
7. In the NAME field, type your name.
8. In the Class field, choose **BUSINESS**.
9. Click **INSERT ORDER**. Notice how the application processes your reservation and displays an **INSERT DONE** message when the function is completed.
10. Your order number should appear in the bottom right corner of the **FLIGHT RESERVATION** window. Write the order number assigned to your reservation in this space: _____.

Part 2: Retrieve an Order

1. From the FILE menu, select **OPEN > ORDER**. The OPEN ORDER screen appears.
2. Put a check in the ORDER No. field then type your order number.
3. Click **OK**. Your order appears on your screen.
4. Change the date of travel to tomorrow's date.
5. Click **UPDATE ORDER**.

Notice how the application processes your update and displays an UPDATE DONE message when the function is completed. This is illustrated in Figure 1-2.

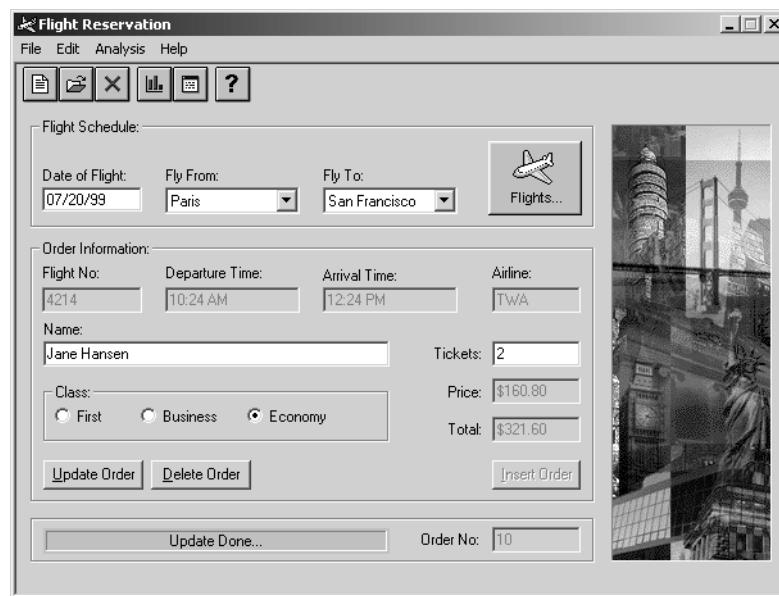


Figure 1-2

Part 3: Delete an Order

1. If your flight reservation order is not displayed on your screen, reopen it from the FILE menu, enter your order number and click **OK**. Your order will appear on your screen.
2. Click the **DELETE ORDER** button.
3. When the delete confirmation message appears, (Figure 1-3) click **YES**. Your order is now removed from the database.

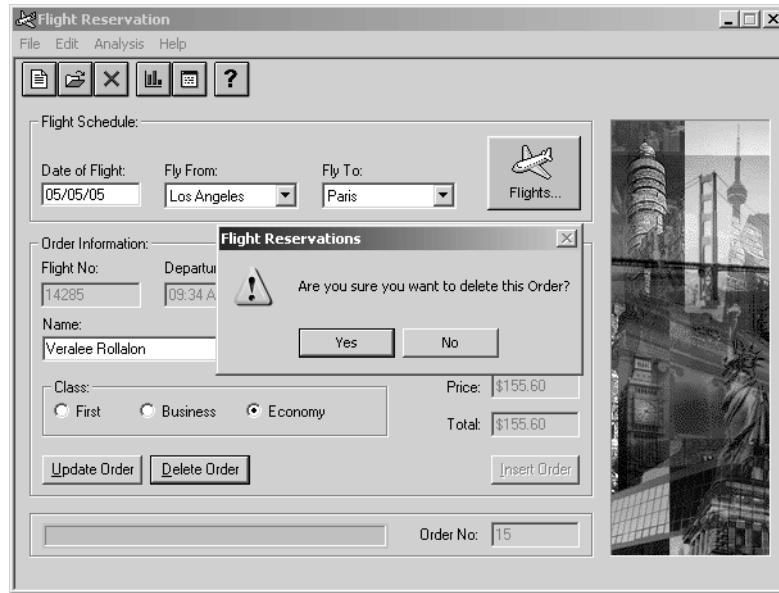


Figure 1-3

2

Recording and Running a Test

Objectives

This chapter contains one exercise with four parts:

- Record and Playback
 - Set-up the test environment.
 - Record a test.
 - Run a test.
 - Review the test results.

Exercise: Record and Playback

In this exercise, you will create a basic test by recording user actions specified in the Mercury Flight Reservation test case. The test should run without any errors.

Remember that QuickTest must be opened before you launch the Mercury Flight Reservation application. This allows QuickTest to establish internal hooks that will capture user actions from the application under test.

Part 1: Set-up the Test Environment

1. Open QuickTest Professional from the START menu. The ADD-IN MANAGER appears.
2. Deselect any add-ins that appear in the ADD-IN MANAGER. There are no add-ins required for creating test cases during these training exercises.
3. Click **OK**. QuickTest Professional launch windows open on your screen.

Note: If the QuickTest WELCOME window appears, deselect the option to ALWAYS SHOW ON START UP.

4. Open the Mercury Flight Reservation application. The login window appears.
5. In the AGENT NAME field type: jojo. In the PASSWORD field type: mercury. The Mercury Flight Reservation launch windows open on your screen.
6. Re-open QuickTest Professional then click **FILE > NEW TEST** from the QuickTest menubar.

7. Click **RECORD**. The RECORD AND RUN SETTINGS window opens (Figure 2-1).
8. Note: Additional environment tabs will appear in this dialog depending on which Add-ins you have loaded. In the example below the Web tab is available because the Web Add-in has been loaded.

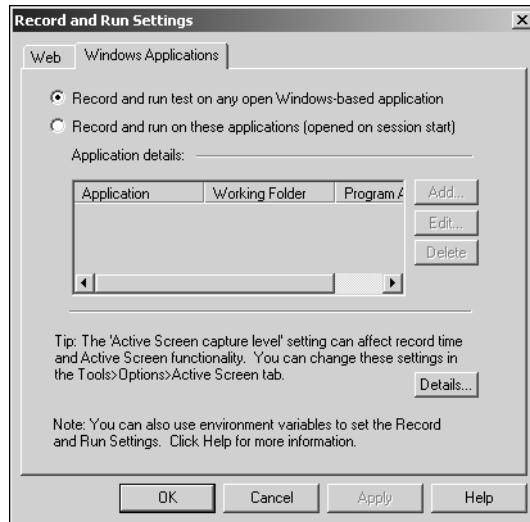


Figure 2-1

9. From the **WINDOWS APPLICATIONS** tab select RECORD AND RUN TEST ON ANY OPEN WINDOWS APPLICATION.
10. Click **OK** to return to the main QuickTest window.

Part 2: Record the Test

1. Begin recording user actions by re-displaying the flight reservation screen. (You may need to click the Flight icon on your windows taskbar). Notice that the word RECORDING flashes in red at the bottom right side of the screen as shown in Figure 2-2.

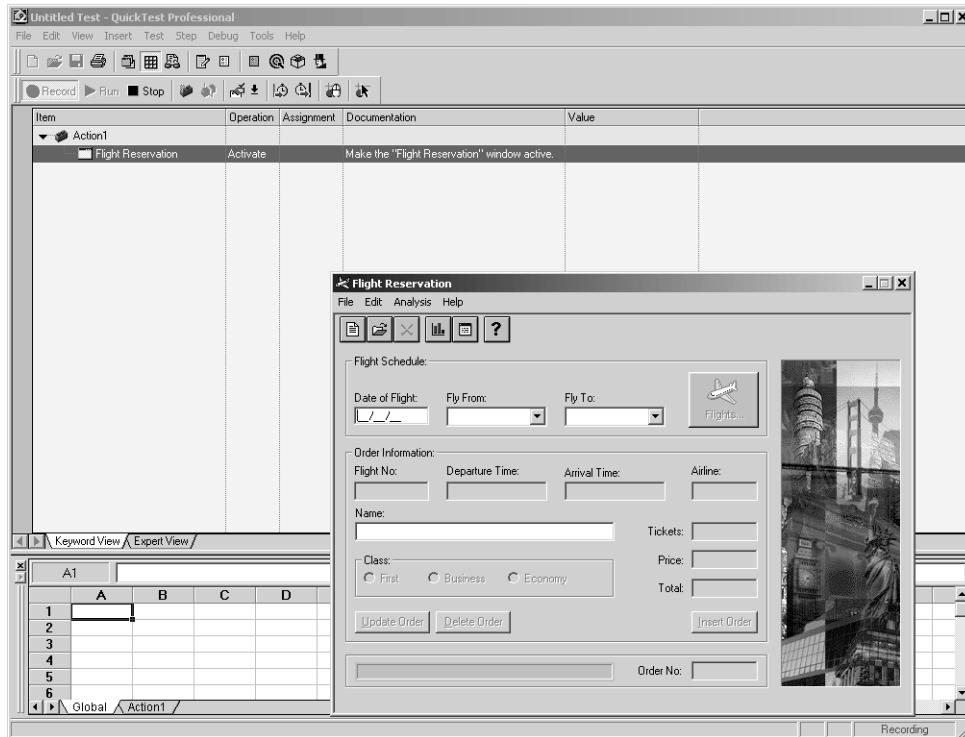


Figure 2-2

2. Record the user actions that will create a new flight reservation. Make up your own data or use the following information: date of travel 12/12/09, flying from Frankfurt to London.

3. Click the **FLIGHTS** button and accept the first flight on the schedule. Click **OK**.
4. Type your name in the NAME field and choose FIRST class.
5. Click **INSERT ORDER**. Watch to ensure that the message **INSERT DONE...** appears in the progress bar on the flight reservation screen.
6. Click **NEW ORDER** button. 

Note: Clicking the **NEW ORDER** button at this step ensures that the screen appears exactly as it did at the start of the test. This is referred to as resetting the end condition to match the initial condition of the test.

7. From the QuickTest menu, click **STOP** to end the test.
8. Click **FILE > SAVE** to store your test as <your initials>_CreateOrder.

Part 3: Run the Test

1. If the test created in Part 2: Record the Test is not already displayed on your screen, find and open it by clicking **FILE > OPEN TEST** from the QuickTest menu bar.
2. Click **RUN**. In the **RUN RESULTS** location tab, (Figure 3.) choose the option to save your test results to a temporary file. Click **OK** to close the window.

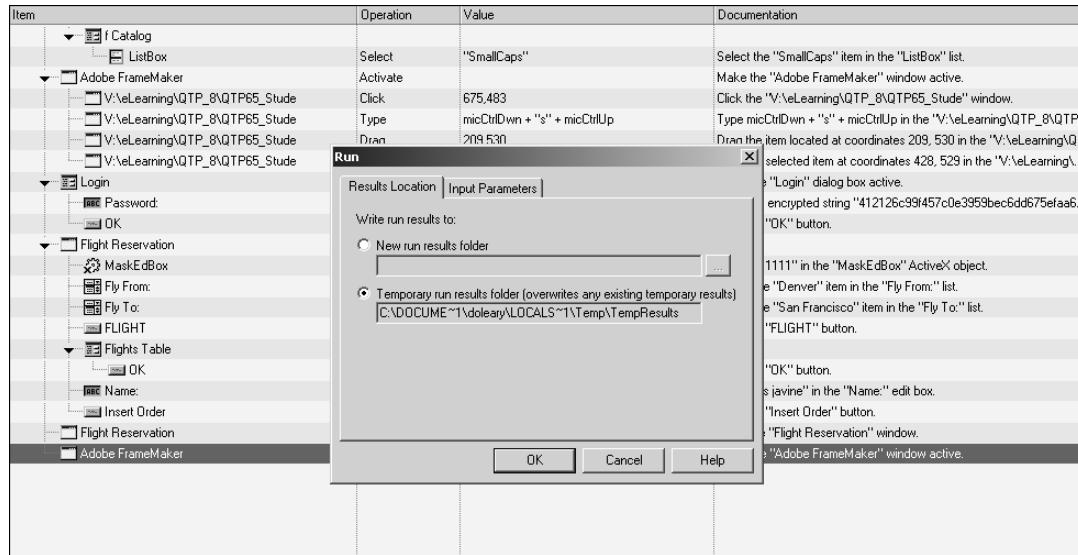


Figure 2-3

3. Open the Mercury Flight Reservations application. The test should automatically begin repeating the actions you recorded earlier.

4. When the test run is complete, the test results will appear on your screen (Figure 2-4).

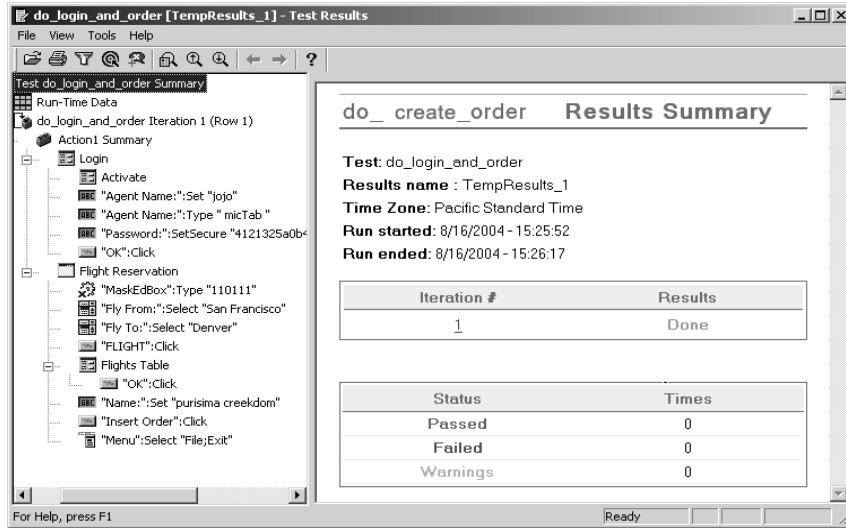


Figure 2-4

Note: If the RESULTS SUMMARY window is not automatically displayed when the test run ends, click TEST > RESULTS from the menu bar.

Test Results Review

Click **TEST > RESULTS** to see a summary of your test run.

1. How many times did your test run?
-

2. Did your test succeed or fail?
-

3. Write the command to display all of the steps in the icon view of the test results.
-

Close the TEST RESULTS window. The main QuickTest window appears.

4. Click once on the **AGENT NAME** step in the KEYWORD VIEW. What field is highlighted in the ACTIVE SCREEN view?
-

5. Click once on the **FLY TO:** step in the KEYWORD VIEW. What happened in the ACTIVE SCREEN when you selected this step?
-

Test Results Review Answers

1. How many times did your test run?

The test ran one time as indicated by the number of iterations in the summary.

2. Did your test succeed or fail?

The test was designed to succeed. If your test failed, try repeating the steps listed in the lab exercise.

3. Write the command to display all of the steps in the icon view of the test results.

VIEW > EXPAND ALL

Close the TEST RESULTS window. The main QuickTest window appears.

4. Click once on the **AGENT NAME** step in the KEYWORD VIEW. What field is highlighted in the ACTIVE SCREEN view?

The Agent Name field is highlighted.

5. Click once on the **PASSWORD** step in the KEYWORD VIEW. What happened in the ACTIVE SCREEN when you selected this step?

The same screen appears but the Password field is now highlighted.

3

Understanding Objects in QuickTest

Objectives

This chapter contains three exercises:

- Understanding objects
- Identifying objects
 - Create a new reservation
 - Investigate the objects that are recorded by QuickTest.
- Using the Object Repository
 - Use the object repository to change the default logical name given to an object.

Understanding Objects Review

Answer the following questions to test your understanding of the ways that QuickTest uses objects to record and playback automated tests.

1. Circle each item on this list that represents an object class.
 - a) Test
 - b) Data entry
 - c) Drop-down List
 - d) Error message

2. The only way to distinguish one object from the other of the same class is by the difference in object characteristics.

TRUE	FALSE
------	-------

3. After learning the class and properties of an object, QTP assigns a name to the object that is referred to as the:
 - a) logical name
 - b) object class
 - c) object property
 - d) value
4. QuickTest Professional stores recorded object properties in the:
 - a) Object Spy
 - b) Object Repository
 - c) Property Value
 - d) Test Results

Understanding Objects Review Answers

1. The correct answer is Drop-down list. An object is a graphic user element in an application, such as a button or a list or an edit box.
2. The correct answer is True. Specific characteristics of an object within QuickTest are called object properties.
3. After learning the class and properties of an object, QTP assigns a name to the object. This is known as the object's logical name.
4. Recorded object properties are stored in QTP's Object Repository.

Exercise: Identifying Objects

In this exercise, you will create a new test and then investigate the objects that are recorded by QuickTest.

Part 1: Create a new test

1. Open QuickTest Professional.
2. Open the Mercury Flight application. START→PROGRAMS→QUICKTEST PROFESSIONAL→SAMPLE APPLICATION→FLIGHT
3. Login using the following information: Agent Name: jojo Password: mercury. The Flight Reservation window should appear on your screen.
4. From the QuickTest toolbar, click the Record button  . The RECORD AND RUN SETTINGS window should open. If it does not open automatically, select TEST→RECORD AND RUN SETTINGS.
5. Choose RECORD ON ANY OPEN WINDOWS-BASED APPLICATION then click **OK**. RECORDING appears in red in the lower right of the QuickTest window.
6. Re-display the Flight Reservation window (minimized in your taskbar) and enter the following flight data: Date: 11-11-09, Flying from Denver to Seattle.
7. Click on the **FLIGHT** button then click **OK** to accept the first flight schedule.
8. Enter the following personal data: Name: Sam Smith, No. of Tickets: 2, Seating: First Class.
9. Click **INSERT ORDER**.
10. When **INSERT DONE...** appears in the progress bar, click the **NEW ORDER** button  to clear the data from the flight reservations window. This action also resets the original test conditions.
11. Click **STOP** to end the test.

Part 2: Save the Test

1. Choose FILE >SAVE.
2. Save the test as <your initials>_ObjectReview.
3. Click **SAVE**. As shown in Figure 3-1, the name of the test now appears in the title of the QuickTest window.

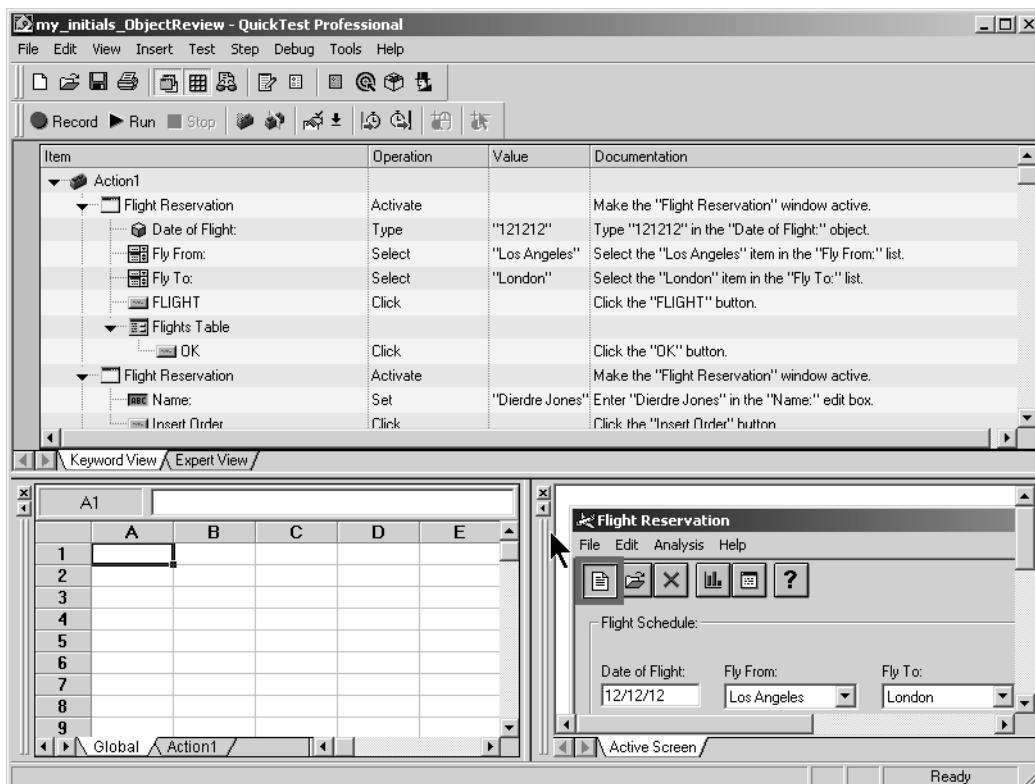


Figure 3-1

Part 3: Run the Test

1. From the QuickTest menu bar, click  .
2. Click **OK** to use the TEMPORARY TEST RESULTS directory for this test run.
3. Observe playback of the test.
4. This test was designed to succeed. If your test fails, open the TEST RESULTS to analyze and resolve the problem.

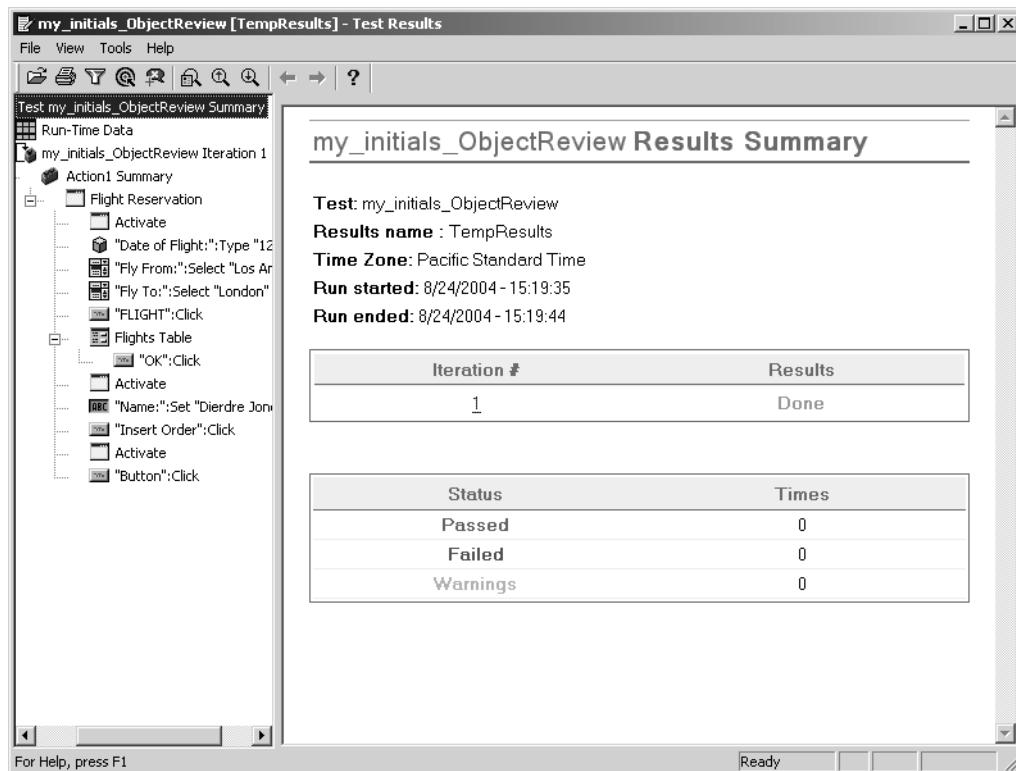


Figure 3-2

Test Objects Review

Open the object repository from the main QuickTest window by selecting TOOLS→OBJECT REPOSITORY and answer the following questions:

1. How many objects can you identify?

2. How many different classes of objects can you identify?

3. How many objects of the class BUTTON are shown?

Test Objects Review Answers

How many objects can you identify?

10 - Flight reservation, date of flight, tickets, name, flyfrom, flyto, insert order, flight, button, flights table.

4. How many different classes of objects can you identify?

7 - window, winbutton, winobject, winedit, winobject, wincombobox, and dialog.

5. How many objects of the class BUTTON are shown?

4 - insert order, flight, button, and OK.

Exercise: Changing the Object Name

In this exercise, you will change the logical name that QuickTest assigned to an object in your test. Although this concept was not specifically covered in the CBT, you should be able to follow the detailed steps in this exercise for a successful result.

Begin by opening the test you created in “Part 1: Create a new test” on page 4.

1. In the KEYWORD VIEW, use your mouse to highlight the last step in the test as shown in Figure 3-3.

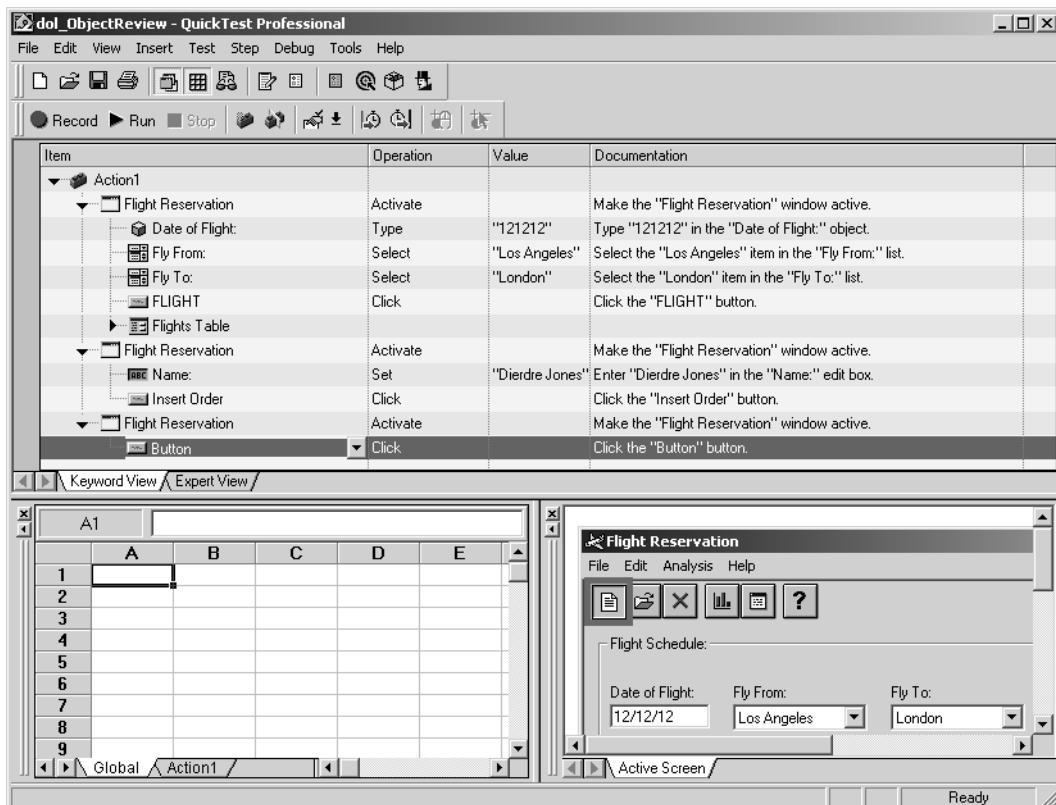


Figure 3-3

2. Notice that the name of the object is **BUTTON** and the documentation column describes the action as **CLICK THE “BUTTON” BUTTON**. To assign a more descriptive name, right-click to open the step menu.
3. Select **OBJECT PROPERTIES** from the step menu.
4. Click the **REPOSITORY** button on the OBJECT PROPERTIES window.

5. Right-click on the **BUTTON** object to open the menu (Figure 3-4).

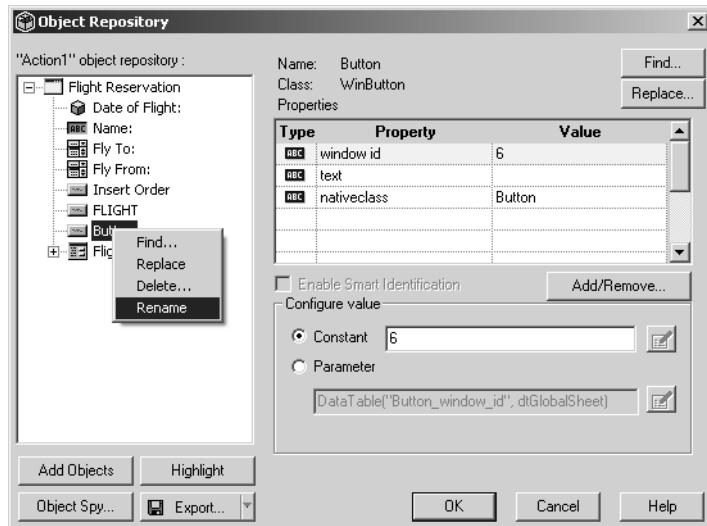


Figure 3-4

6. Type a new name that describes the button such as "New Order".
 7. Click **ENTER** or click **OK** twice to save the change and close the object repository window.

The new button name should appear in the **ITEM** column and the **DOCUMENTATION** column in the **KEYWORD VIEW** as shown in Figure 3-5.

Item	Operation	Value	Documentation
Action1			
Flight Reservation	Activate		Make the "Flight Reservation" window active.
Flight Reservation	Activate		Make the "Flight Reservation" window active.
Flight Reservation	Activate		Make the "Flight Reservation" window active.
New Order	Click		Click the "New Order" button.

Figure 3-5

4

Creating Synchronization Steps

Objectives

This chapter contains three exercises:

- Synchronization Review
- Analyzing a Failed Test
- Inserting a Synchronization Point

Synchronization Review

Answer the following questions to test your understanding of the ways that QuickTest uses synchronization steps for automated tests.

1. Circle each function that synchronization performs in a test.
 - Updates test data.
 - Provides time for an object to process before moving on to the next step.
 - Provides time for an environment variable to run.
 - Sets an environment variable for an object.

 2. Synchronization is enabled only during recording.

TRUE	FALSE
------	-------

 3. Write the command to insert a synchronization point in a test.
-

4. Circle each application object that might require synchronization.
 - Window frame
 - Edit box
 - Insert button
 - Popup message

 5. Open the test called “SYNCH_EXAMPLE” from the directory “Ch04_synch_example” in “C:\MTesting”. *Note: You created the “C:\MTesting” directory when following the steps in the SetupInstructions document.* Review the test and write the name of the step that represents the synchronization point.
-

Synchronization Review Answers

1. Circle each function that synchronization performs in a test.

The correct answer is: Provides time for an object to process before moving on to the next step.

2. Synchronization is enabled only during recording.

The correct answer is - TRUE.

3. Write the command to insert a synchronization point in a test.

Insert > Step > Synchronization Point

4. Circle each application object that might require synchronization.

The correct answer is: Insert Button and Popup Message.

5. The synchronization point in the SYNCH EXAMPLE test is named INSERT DONE TEXT (Figure 4-1).

Item	Operation	Documentation
Action1		
Flight Reservation		
MaskEditBox	Type	Type "070411" in the "MaskEditBox" ActiveX object.
Fly From:	Select	Select the "Los Angeles" item in the "Fly From:" list.
Fly To:	Select	Select the "Denver" item in the "Fly To:" list.
FLIGHT	Click	Click the "FLIGHT" button.
Flights Table		
From	Select	Select the "2922 LAX 03:04 PM DEN 04:02 PM SW \$127.40" item in the "From" list.
OK	Click	Click the "OK" button.
Name:	Set	Enter "Mary Limone" in the "Name:" edit box.
First	Set	Select the "First" radio button.
Insert Order	Click	Click the "Insert Order" button.
Flight Reservation	Activate	Make the "Flight Reservation" window active.
Flight Reservation	Move	Move the "Flight Reservation" window to the screen coordinates, 540, 243.
Insert Done Text	WaitProperty	Check whether the value of the "text" property achieves the value "Insert Done..." within 20000 milliseconds.

Figure 4-1

Exercise: Analyzing a Failed Test

In this exercise you will observe a failed test by changing the default time-out setting.

1. From QuickTest Professional, open the test called Create_Order.
2. Launch the Flight Reservation application but do not log in. This test begins with a log in script.
3. Click **RUN** to playback the test.
4. Save the test results to a temporary directory.
5. Review the test summary by clicking **TEST > RESULTS**. The test should have succeeded without any errors.
6. Close the TEST RESULTS window. The Create_Order test should be displayed on your screen.
7. Click **TEST > SETTINGS** to open the TEST SETTINGS window.
8. Click the **RUN** tab and change the OBJECT SYNCHRONIZATION TIMEOUT value from 20000 to 5000 milliseconds as shown in Figure 4-2. This tells QuickTest to wait just 5 seconds before continuing to the next step in the test.

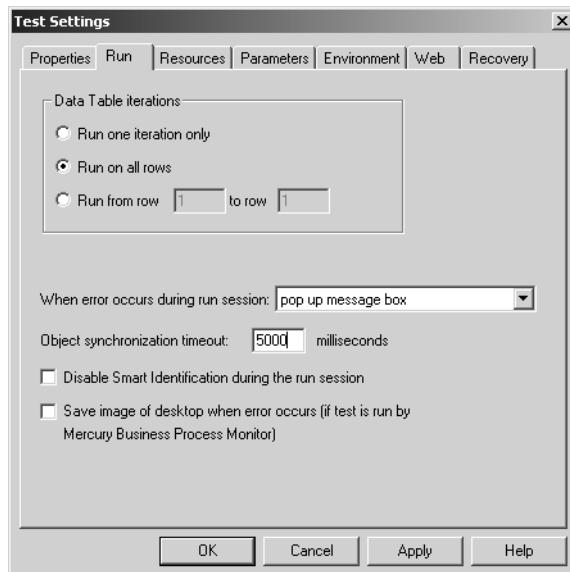


Figure 4-2

9. Run the test again. The test should have failed because the objects in the application were not ready when QuickTest arrived at that point. For example, Figure 4-3 illustrates the error message that was generated when QuickTest tried to execute the File > Exit command before the Insert Order command was completed.

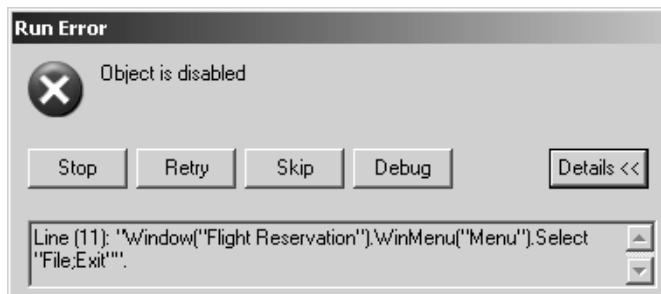


Figure 4-3

What would you do to correct this problem? One solution would be to reset the timeout value to 20000. However, if you know that your application has points that will require synchronization, you can record this in the basic test by inserting a synchronization point.

Exercise: Inserting a Synchronization Point

In this exercise you will create a new test with a synchronization requirement by recording user actions from the Mercury Flights application.

1. Open QuickTest Professional.
2. Open the Mercury Flight Reservation application.
3. Log in to the Flight Reservation application as Agent Name: jojo Password: mercury. The Flight Reservation window should be displayed with no data entered.
4. Click **RECORD** to start the test. Record user actions to create a new flight reservation. Make up your own data or use the following information: date of travel 02/12/10, flying from Denver to Los Angeles.
5. Click the **FLIGHTS** button and accept the first flight on the schedule. Click **OK**.
6. Type your name in the NAME field and choose FIRST class.
7. Click **INSERT ORDER**. At this point, you want to make sure QuickTest waits for the order to process completely before moving to the next step in the test. The visual cue that the order has processed completely appears in the progress bar (Figure 4-4).

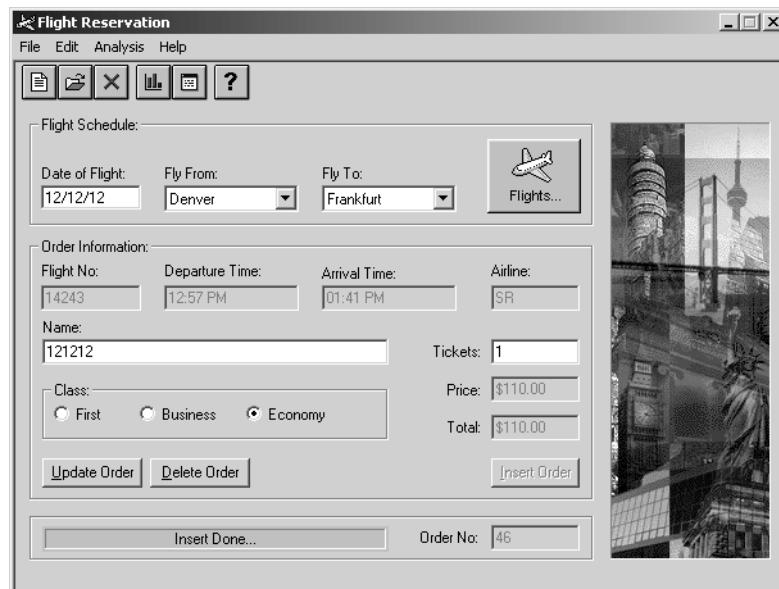
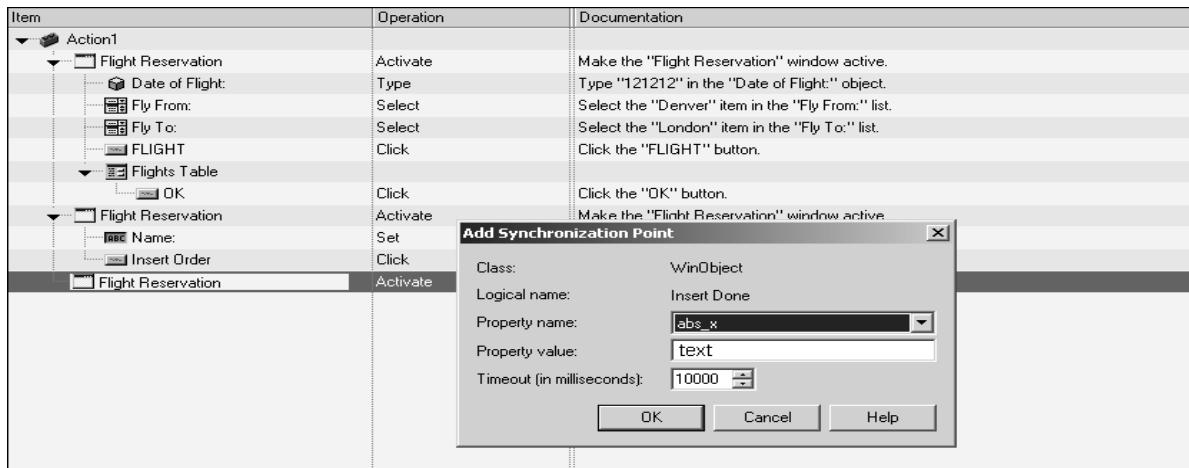


Figure 4-4

8. From the QTP menu bar, select **INSERT >STEP > SYNCHRONIZATION POINT**. The cursor becomes a pointed finger.
9. Click on the words “**INSERT DONE...**” in the progress bar with the pointed finger.
10. In the **OBJECT SELECTION-SYNCHRONIZATION POINT** window, click **OK**. The **ADD SYNCHRONIZATION POINT** window opens (Figure 4-5).

**Figure 4-5**

11. Type the word: **text** in the PROPERTY VALUE field. Keep the default timeout of 10000 milliseconds.
12. Click **OK** to close the synchronization window.
13. Finish the test by selecting **FILE > NEW ORDER** to reset the initial condition.
14. Click **RUN** to replay your test.
15. View the test results to ensure that your test was executed successfully.

5

Using Standard Checkpoints

Objectives

This chapter contains three exercises:

- Checkpoint Review
- Add a Standard Checkpoint
- Analyze Test Results

Checkpoints Review

1. What is the function of a checkpoint in QuickTest?

2. Where is the status of a checkpoint (passed or failed) located?

3. What causes a checkpoint to fail?

Checkpoints Review Answers

1. A checkpoint is a specialized step in QuickTest that compares two values and reports the result.
2. The status of a checkpoint (passed or failed) is located in the Test Results Summary.
3. If the two values match, the checkpoint passes; if not, the checkpoint fails.

Exercise: Add a Standard Checkpoint

In this exercise you will create a basic test with one exception - in this test you will insert a standard checkpoint during the recording phase.

Part 1: Create a New Test and Checkpoint

1. Open QuickTest Professional.
2. Open the Mercury Flight application. START→PROGRAMS→QUICKTEST PROFESSIONAL→SAMPLE APPLICATION→FLIGHT
3. Login using the following information: Agent Name: jojo Password: mercury. The Flight Reservation window should appear on your screen with no data entered.
4. Click **RECORD** to start the test. Record user actions to create a new flight reservation. Make up your own data or use the following information: date of travel 02/09/09, flying from Zurich to Portland.
5. Click the **FLIGHTS** button then click **OK** to accept the first flight schedule.
6. Type your name in the NAME field and choose FIRST class.
7. Click **INSERT ORDER**. Your verification that the order has been inserted correctly is when the text **INSERT DONE...** appears in the progress bar; QuickTest's verification will be a standard checkpoint at this step in the test.

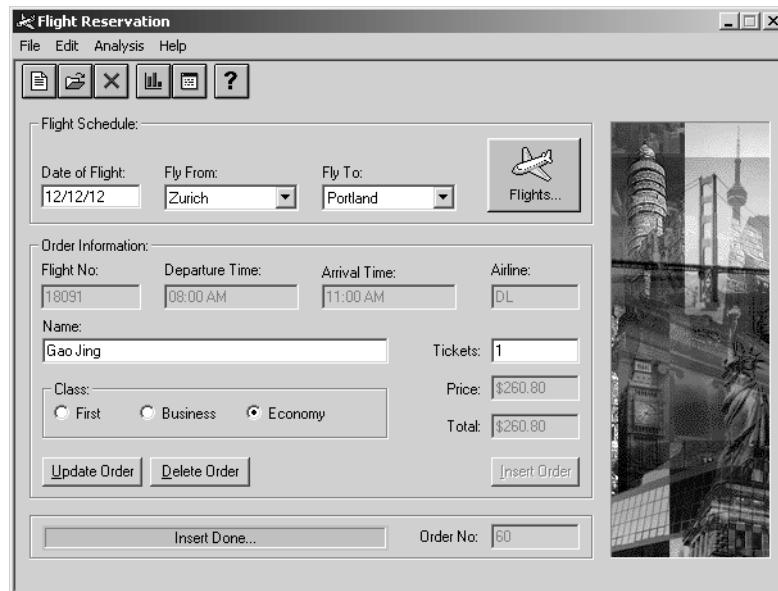


Figure 5-1

8. From the QTP menu bar, select **INSERT >CHECKPOINT > STANDARD CHECKPOINT**. The cursor becomes a pointed finger.
9. Click on the words “**INSERT DONE...**” in the progress bar with the pointed finger.
10. In the **OBJECT SELECTION-CHECKPOINT PROPERTIES** window, click **OK**. The **CHECKPOINT PROPERTIES** window opens (Figure 5-2).

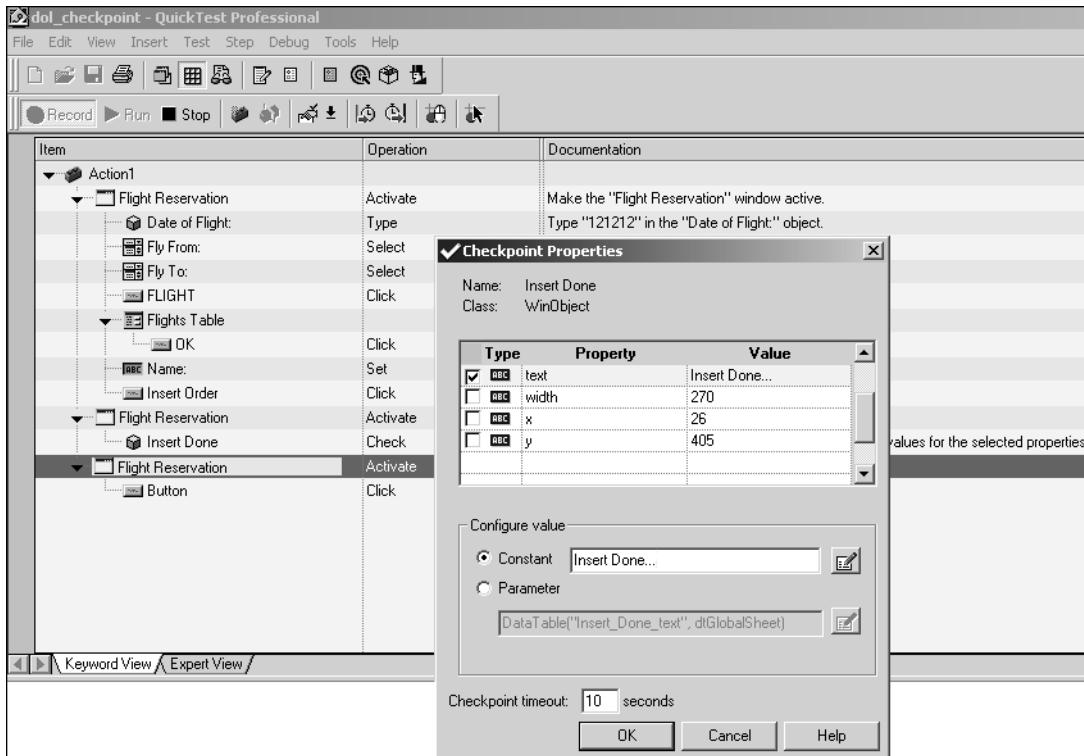


Figure 5-2

11. The property you want to use as a verification is the text **INSERT DONE...**. Deselect all other checkboxes and select the **text** property. **INSERT DONE...** should appear in the **CONSTANT** field as shown in Figure 5-2.
12. Click **OK** to close the **CHECKPOINT PROPERTIES** window.
13. In the Flight Reservation application, click **FILE > NEW ORDER** to set the end condition of the test to match the conditions that existed when the test began.
14. Stop the test.
15. Save the test.

Test Results Review

Run the test. Review the results of the test (**TEST > RESULTS**) to answer the following questions.

1. Did your test pass or fail? Where did you locate this information?

2. Did your checkpoint pass or fail? Where did you locate this information?

3. Display the details of your checkpoint step. What command did you use to perform this task?

Test Results Review Answers

1. This test was designed to run successfully. The word PASSED should appear in green text in the ITERATION RESULTS table as shown in Figure 5-3

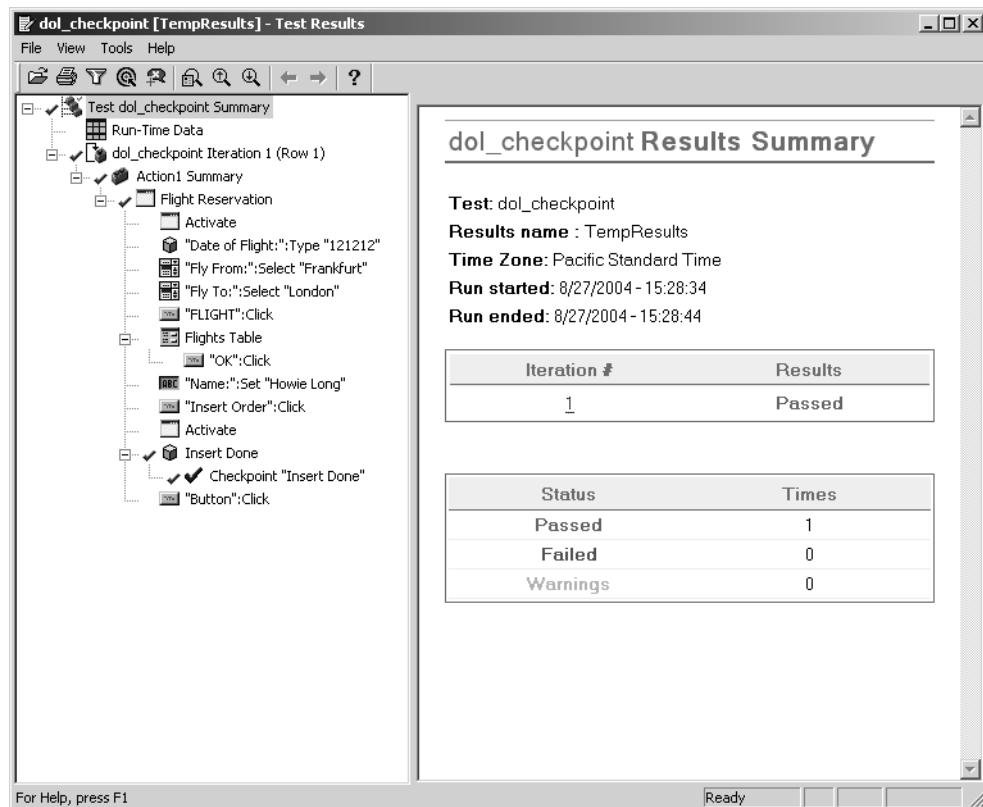


Figure 5-3

2. This checkpoint was designed to run successfully. Double-click the checkpoint step to display the details as shown in Figure 5-4

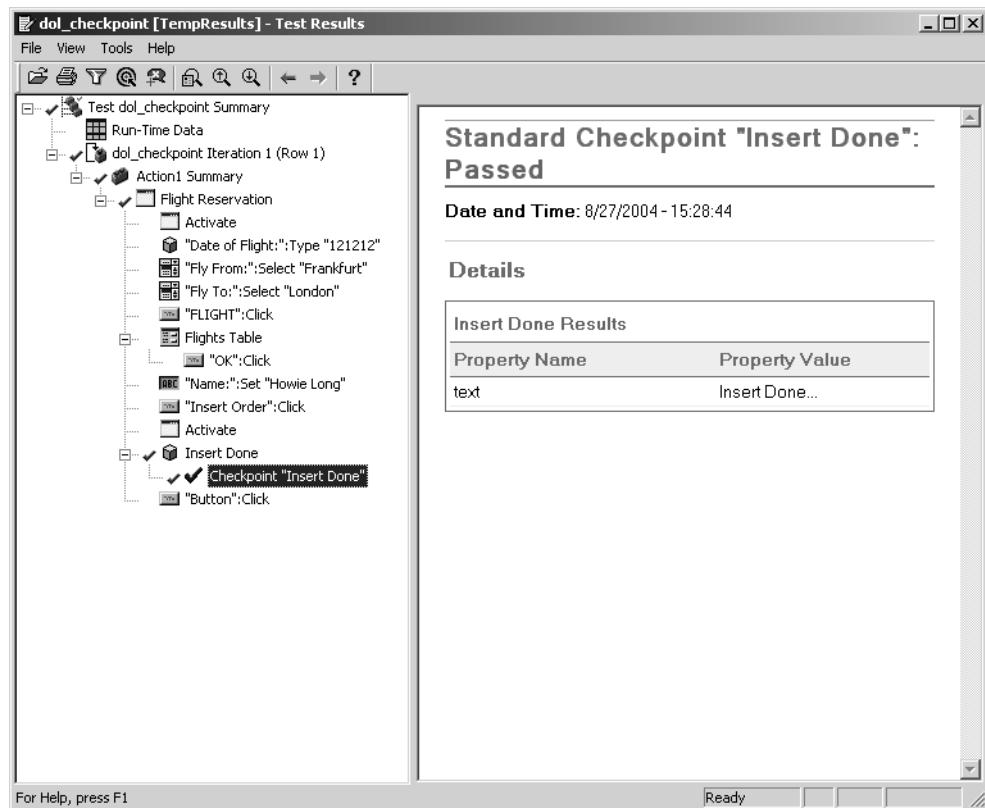


Figure 5-4

6

Adding Parameters

Objectives

This chapter contains three exercises:

- Parameter Types Review
- Create and Run a Test
 - Create a new test
 - Analyze the test results
- Add an Output Parameter

Parameter Types Review

1. Draw a line to connect the parameter type in the left column to the appropriate usage in the right column.

Parameter Type	Usage
Random Number	Insert a number generated earlier in the test run into the ORDER NUMBER field each time the test runs.
Data Table - Input	Insert a different department code into the DEPARTMENT NUMBER field each time the test runs.
Environment	Insert an action from a business process test in Mercury Quality Center.
Data Table - Output	Insert a number from 1-10 into the NUMBER OF ORDERS field each time the test runs.
Component	Insert the server name into the HOST ID field each time the test is run.

Table 6-1.

2. A data-driven test performs the following functions: (circle all that apply).
 - a) Runs multiple test iterations using different input values.
 - b) Uses a database to create a new test.
 - c) Connects multiple test runs.
 - d) Reports new data for a test.

3. List the differences between the Design-Time data table and the Run-Time data table.

Parameter Types Review Answers

1.

Parameter Type	Usage
Random Number	Insert a number from 1-10 into the NUMBER OF ORDERS field each time the test runs.
Data Table - Input	Insert a different department code into the DEPARTMENT NUMBER field each time the test runs.
Environment	Insert the server name into the HOST ID field each time the test is run.
Data Table - Output	Insert a number generated earlier in the test run into the ORDER NUMBER field each time the test runs.
Component	Insert an action from a business process test in Mercury Quality Center.

2. A data-driven test runs multiple test iterations using different input values.
3. Some differences include:

Design-Time Data Table:	Run Time Data Table:
<u>Viewed in the main QuickTest window</u>	<u>Viewed in the Test Results window</u>
<u>Created prior to the execution of the test.</u>	<u>Generated after a test is executed.</u>
<u>Represents data from an external source.</u>	<u>Represents a “live” version of the Design-Time table after the test run.</u>

Exercise: Create and Run a Test

In this exercise you will create a basic test that requires parameterization to run successfully.

Part 1: Create a New Test and Checkpoint

1. Open QuickTest Professional.
2. Open the Mercury Flight application. START→PROGRAMS→QUICKTEST PROFESSIONAL→SAMPLE APPLICATION→FLIGHT
3. Login using the following information: Agent Name: jojo Password: mercury. The Flight Reservation window should appear on your screen with no data entered.
4. Click **RECORD** to start the test. Record user actions to create a new flight reservation. Make up your own data or use the following information: date of travel 05/25/11, flying from Frankfurt to Los Angeles.
5. Click the **FLIGHTS** button then click **OK** to accept the first flight schedule.
6. Type your name in the NAME field and choose FIRST class.
7. Click **INSERT ORDER**. Wait until INSERT DONE...appears in the progress bar as illustrated in Figure 6-1.

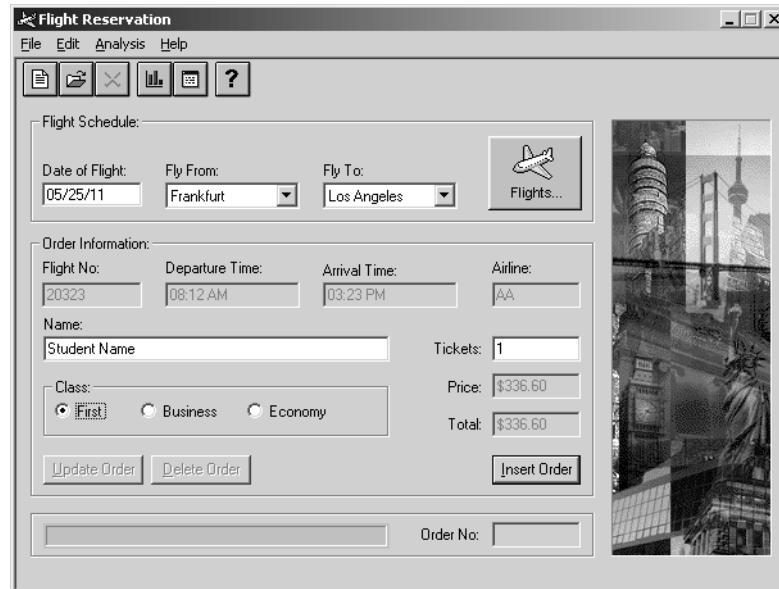


Figure 6-1

8. In QuickTest Pro insert a synchronization point on the INSERT ORDER step:
- In QuickTest Pro select **INSERT > STEP > SYNCHRONIZATION POINT**. Your cursor changes to a pointing hand and the Flights window is redisplayed.
 - Single-click on the **INSERT DONE...** text in the progress bar on the Flights window. The **OBJECT SELECTION** window opens.
 - Verify that **WINOBJECT: INSERT DONE** is selected, then click **OK**. The **ADD SYNCHRONIZATION POINT** window opens.
 - In the **PROPERTY TYPE** field, choose **TEXT**.
 - In the **PROPERTY VALUE** field, type “Insert Done...” including the quotation marks and the dots.
 - Click **OK**. The **INSERT DONE** step now appears as a synchronization point in your test (Figure 6-2).

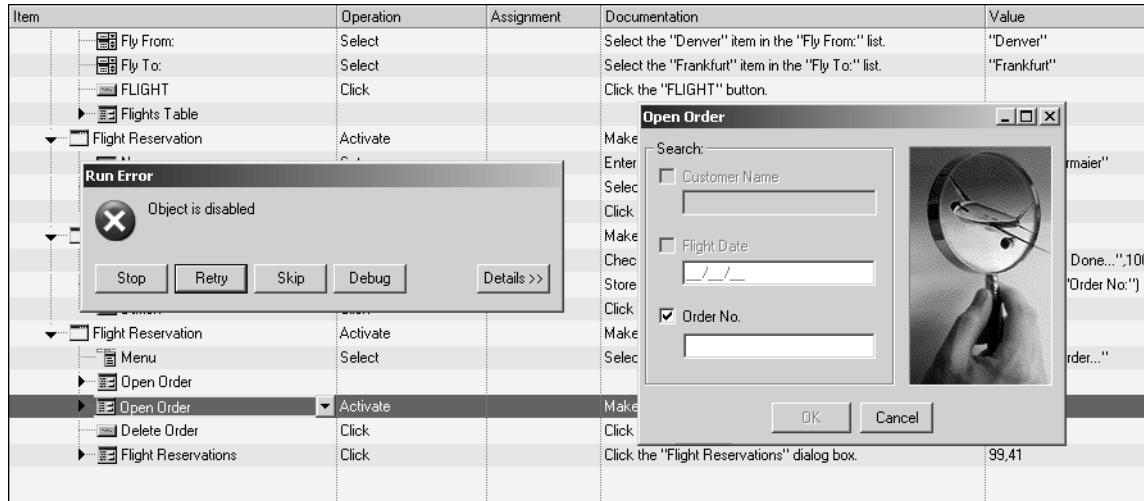
Item	Operation	Documentation
Action1		
Flight Reservation	Activate	Make the "Flight Reservation" window active.
Date of Flight:	Type	Type "121209" in the "Date of Flight:" object.
Fly From:	Select	Select the "Frankfurt" item in the "Fly From:" list.
Fly To:	Select	Select the "Los Angeles" item in the "Fly To:" list.
FLIGHT	Click	Click the "FLIGHT" button.
Flights Table		
Flight Reservation	Activate	Make the "Flight Reservation" window active.
Name:	Set	Enter "Jane Jenkins" in the "Name:" edit box.
Tickets:	SetSelection	Select the text from character 0 to character 1 within the "Tickets:" edit box.
Tickets:	Set	Enter "2" in the "Tickets:" edit box.
First	Set	Select the "First" radio button.
Flight Reservation	Move	Move the "Flight Reservation" window to the screen coordinates, 258, 401.
Flight Reservation	Move	Move the "Flight Reservation" window to the screen coordinates, 227, 205.
Insert Order	Click	Click the "Insert Order" button.
Insert Done	WaitProperty	Check whether the value of the "abs_x" property achieves the value "Insert Done..." within 10000 milliseconds.

Figure 6-2

9. Insert a checkpoint to validate the ORDER NUMBER number field:
 - a) Click **INSERT > CHECKPOINT > STANDARD CHECKPOINT**. The cursor changes to a pointing hand.
 - b) Single-click on the **ORDER NUMBER** field on the FLIGHTS window.
 - c) Make sure that **WINEDIT: ORDER NO.** is highlighted in the OBJECT SELECTION window, then click **OK**. The Checkpoint properties window opens.
 - d) Make sure that only the ENABLED field has a checkmark, then click **OK**. The Checkpoint is inserted into the text.
10. In the Windows Mercury Flight Application click **FILE > OPEN ORDER**.
11. Check the box next to the ORDER No. field.
12. Type the order number of the reservation you just created. The order should reappear on your screen. Click **OK**.
13. Change the number of tickets from 1 to 2.
14. Click **UPDATE ORDER**.
15. Click **FILE > NEW ORDER** to reset the original conditions of the test.
16. Click **STOP** to end the recording.

Part 2: Run the Test and Analyze the Error

- Click **RUN**. Your test should encounter an error as illustrated in Figure 6-3.



The screenshot shows a test log window with a 'Run Error' dialog box overlaid. The error message in the dialog box says 'Object is disabled'. Below the dialog, the test log table lists several steps:

Item	Operation	Assignment	Documentation	Value
Fly From:	Select		Select the "Denver" item in the "Fly From:" list.	"Denver"
Fly To:	Select		Select the "Frankfurt" item in the "Fly To:" list.	"Frankfurt"
FLIGHT	Click		Click the 'FLIGHT' button.	
Flights Table				
Flight Reservation	Activate			
Run Error				
Flight Reservation	Activate			
Menu	Select			
Open Order	Activate			
Open Order	Activate			
Delete Order	Click			
Flight Reservations	Click		Click the "Flight Reservations" dialog box.	99,41

A separate 'Open Order' dialog box is shown, containing fields for Customer Name, Flight Date, and Order No., with a magnifying glass icon.

Figure 6-3

In order to perform the open order step, the test is searching for an ORDER NO. that was deleted when you recorded the test. This order was assigned to the previous reservation and does not appear in this test iteration. Instead of searching for a constant order number, your test should be using whatever order number is generated and inserted into the test iteration.

What should you do to make your test run successfully?

One solution is to create an output parameter that instructs QuickTest to use the order number generated by the application under test.

Exercise: Create an Output Parameter

Part 1: Capture an Output Value

1. Highlight the step **ORDER NUMBER** step in the Keyword View. This is the step that was created during the checkpoint part of the previous exercise.
2. Move your cursor to the Active Screen (lower right) and right-click on the ORDER No. field
3. Choose **INSERT OUTPUT VALUE** from the ORDER No. field menu. The **OBJECT SELECTION -OUTPUT VALUE PROPERTIES** window opens.
4. Make sure that **WINEDIT: ORDER No.** is highlighted then click **OK**. The **OUTPUT VALUE PROPERTIES** window opens.
5. Check the box next to **TEXT**.
6. Select the radio button for **AFTER CURRENT STEP** located at the bottom of the **OUTPUT VALUE PROPERTIES** window.
7. Click **OK** to close the properties window. A column called **ORDER_No_TEXT_OUT** should appear in the lower left portion of the QuickTest window (Figure 6-5).

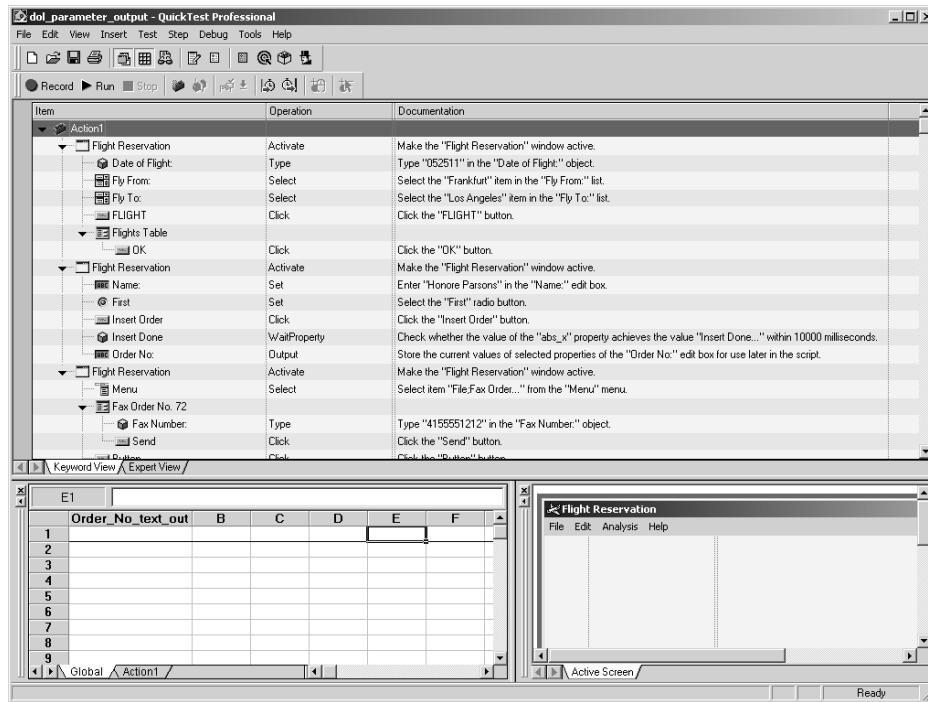


Figure 6-4

Part 2: Run the Test and Analyze the Error

1. Click **RUN**. Your test should encounter an error and fail.
2. Can you determine why your test is continuing to fail?

The test continues to fail because the ORDER NO. window has no value (input). How could you fix the problem? The solution is to modify the ORDER NO. field to accept the value that was generated and stored in the output parameter data table. This is described in “Part 3: Create an Input Parameter” .

Part 3: Create an Input Parameter

1. Highlight the **EDIT** step under the parent step titled **OPEN ORDER** then click once in the **VALUE** column.
2. Click the parameter icon <#> to open the **VALUE CONFIGURATION OPTIONS** window.
3. Click the **PARAMETER** radio button and select **DATA TABLE** from the drop-down list. Make sure that the output parameter data table called **ORDER NUMBER TEXT OUT** appears in the **NAME** field (Figure 6-5).

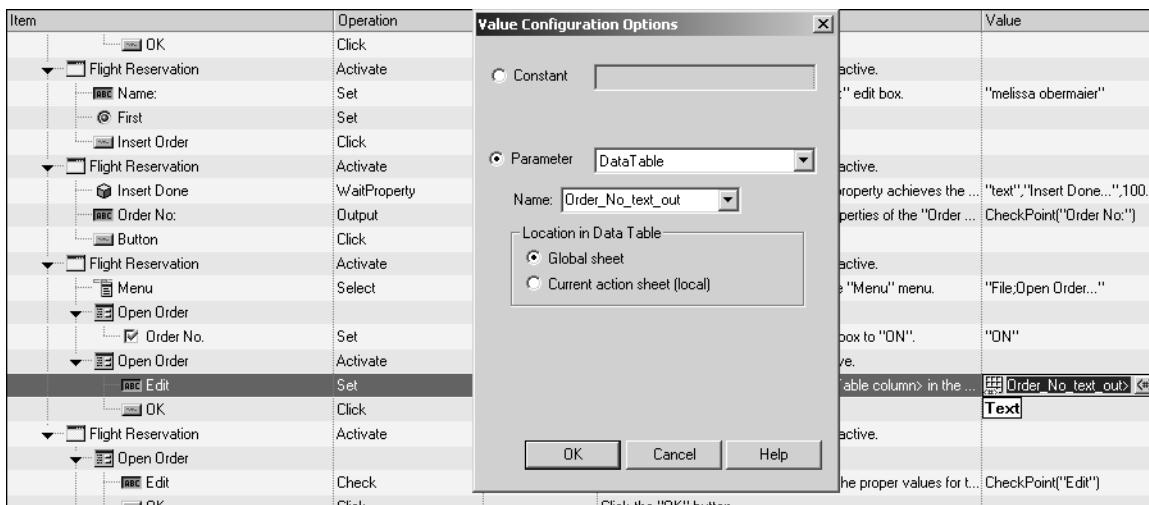


Figure 6-5

4. Click **OK**.
5. Save the test as <your initials>_parameter_example.

Part 4: Run the Test and Celebrate your Success

1. Click **RUN**. Your test should succeed.

7

Recovery Manager

Objectives

This chapter contains three exercises:

- Recovery Manager Review
- Create a New Test
 - Add an input parameter for multiple test iterations
 - Analyze the failed test results
- Create a Recovery Scenario
 - Add two recovery operations to the test
 - Analyze the successful test results

Recovery Manager Review

1. List four types of trigger events that would interrupt your test run.

2. Define the term Recovery Operation.

3. List one example of a Post-Recovery Test Run option.

Recovery Manager Review Answers

1. A checkpoint is a specialized step in QuickTest that compares two values and reports the result.
2. The status of a checkpoint (passed or failed) is located in the TEST RESULTS SUMMARY.
3. A post-recovery test run option is how QuickTest should proceed once the recovery operations have been performed, and from which point in the test QuickTest should continue, if at all. Examples are: restart a test from the beginning, skip the test step entirely and continue with the next step in the test, restart the test, repeat the step again.

Exercise: Parameterized Test

In this exercise you will create a test to validate multiple login names using a data table as a parameter for multiple test iterations.

Part 1: Create a New Test

1. Open QuickTest Professional.
2. Click **RECORD** on the QuickTest Professional menu bar. The RECORD AND RUN SETTINGS window opens. Choose to record on any open windows application.
3. Open the Mercury Flight application. START→PROGRAMS→QUICKTEST PROFESSIONAL→SAMPLE APPLICATION→FLIGHT.
4. Login using the following information: Agent Name: jojo Password: mercury. The Flight Reservation window should appear on your screen with no data entered.
5. Click **FILE > EXIT** to close the Flight application.
6. Click **STOP** on the QuickTest Professional menu bar.

Save the test as <your initials> login_check. Your test should look like the example in Figure 7-1.

The screenshot shows the QuickTest Professional interface with the title bar 'login_check - QuickTest Professional'. The main window displays a table of recorded steps:

Item	Operation	Assignment	Documentation	Value
Action1				
SystemUtil	Run		Open the "C:\Program Files\Mercury Interactive\Flight Reservation\Flight Reservation.exe" application.	"C:\Program Files\Mercury...
Login				
Agent Name:	Set		Enter <the value of the specified Data Table column>.	DataTable("login_name", ...)
Password:	SetSecure		Enter the encrypted string "413e1173fd3ba2ef66c4..."	"413e1173fd3ba2ef66c4...
OK	Click		Click the "OK" button.	
Flight Reservation				
Menu	Select		Select item "File:Exit" from the "Menu" menu.	"File:Exit"

Figure 7-1

Part 2: Add a Data Table Parameter

1. Highlight the AGENT NAME step then click once in the VALUE column.
2. Click the parameter icon <#> to open the VALUE CONFIGURATION OPTIONS window.
3. Click the **PARAMETER** radio button and select **DATA TABLE** from the drop-down list.
4. Type a description for the input parameters, such as Login_Names, in the NAME field.
5. Click **OK**. The heading you created should appear in the first data table column.
6. Add the following 5 names to your data table: bob32, bob0404, bob, bbsmith, bsmith. A similar table is shown in Figure 7-2.

The screenshot shows the QuickTest Professional interface. The main window displays a test script with several steps. One step, 'Action1', contains a 'Login' section with 'Agent Name' set to 'DataTable("login_name")'. Below the script is a data table titled 'A4' with the heading 'helen'. The data table contains the following rows:

	login_name	B	C	D	E	F	G	H	I	J	K	L	M	N
1	jojo													
2	joann													
3	jo													
4	helen													
5														
6														
7														
8														
9														

Figure 7-2

7. Click the **SAVE** icon to retain the changes to your test.

Exercise: Recovery Scenario

Part 1: Create a Recovery Scenario

1. If the test you created is not already displayed on your screen, find and open it by clicking **FILE > OPEN TEST** from the QuickTest menu bar.
 2. Click **RUN**. In the **RUN RESULTS** location tab, choose the option to save your test results to a temporary file. Click **OK** to close the window.
 3. Your test should fail during the third iteration. What error message appeared that stopped the test from continuing?
-
4. To anticipate and correct this error, click **TOOLS > RECOVERY SCENARIO** from the QuickTest menu bar. The Recovery Manager window opens.
 5. Click the **NEW SCENARIO WIZARD** icon as shown in **FIGURE 7-3**. The **RECOVERY SCENARIO WIZARD** opens.

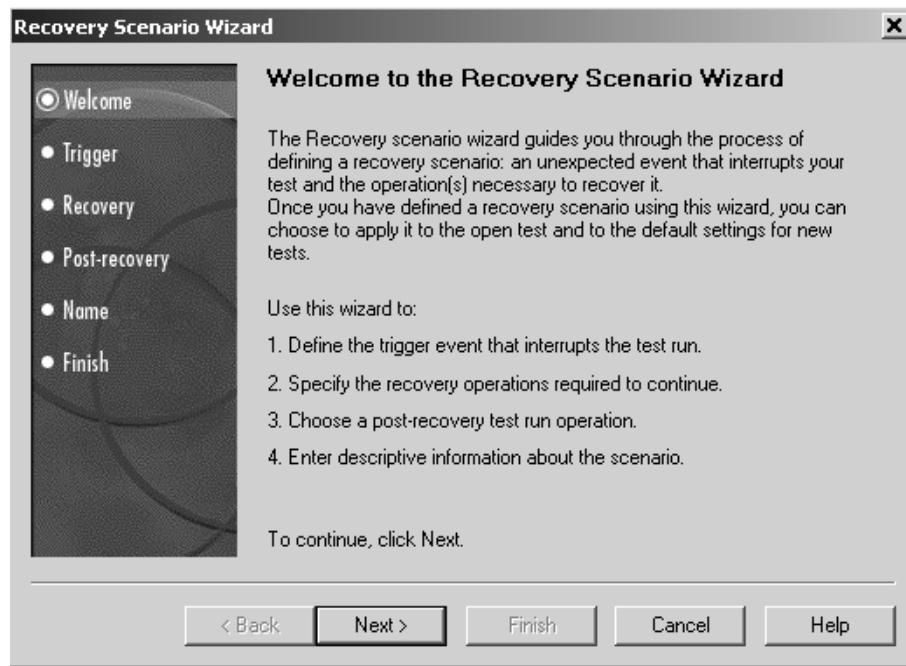


Figure 7-3

6. Review the information on the WELCOME screen then click **NEXT**.

7. Select the trigger event that caused the error (**Pop-up window**), then click **NEXT**.
8. To identify the name of the window where the error message appears, click the **POINTING HAND** icon. The application popup window appears on the screen.
9. Move the **POINTING HAND** to the application popup error window then click once. The name of the window appears in the **WINDOW TITLE:** field in the Recovery Scenario Wizard (Figure 7-4.)

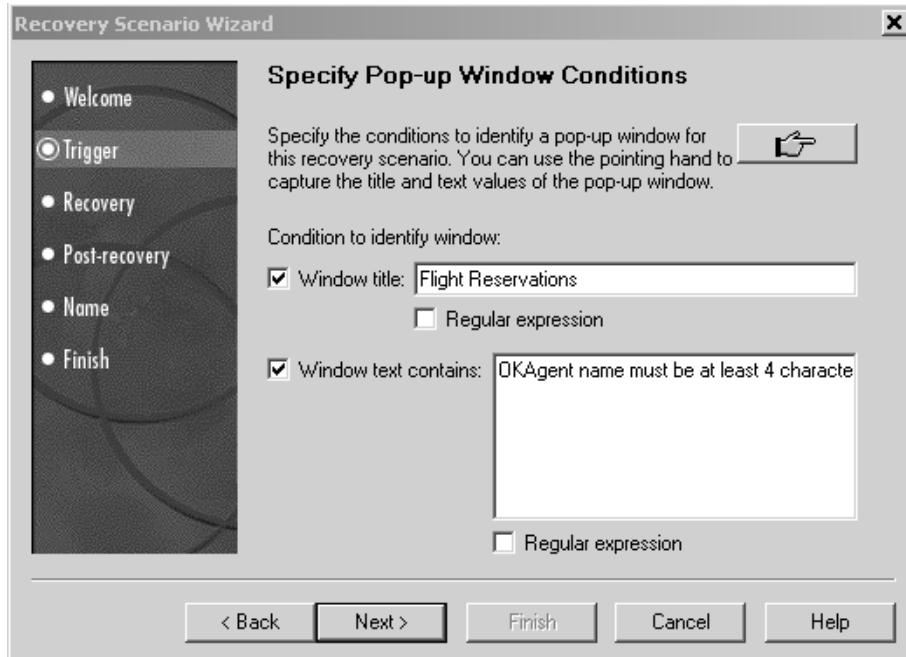


Figure 7-4

10. Click **NEXT** to continue. The RECOVERY screen appears.
11. You do not need to make any additions to this screen. Click **NEXT**. The RECOVERY OPERATION screen appears.
12. Choose the operation type: **KEYBOARD OR MOUSE OPERATION**. Click **NEXT**.
13. Choose the radio button: **CLICK BUTTON WITH LABEL:**.
14. Click on the pointing hand to activate it.
15. With the pointing hand click on the **OK** button in the application popup error window.

16. Click **NEXT**. The RECOVERY OPERATIONS screen opens.
17. De-select the option to **ADD ANOTHER RECOVERY SCENARIO** then click **NEXT**.
18. In the TEST OPTIONS list, choose the **PROCEED TO NEXT STEP** radio button. This enables the test to move to the next step after taking action to resolve the popup window error. Click **NEXT**.
19. Type a name for your scenario. Click **NEXT**. The FINISH screen opens.
20. Check the box to **ADD SCENARIO TO CURRENT TEST**. Click **FINISH**. The name of your scenario should now appear in the Recovery Manager window, as shown in Figure 7-5.



Figure 7-5

21. Click the **SAVE** icon on the Recovery Manager window to store the scenario you created as a .qrs file.
22. Click **CLOSE**.
23. Stop the test run.

Part 2: Run the Test and Analyze the Results

If you analyzed the application error thoroughly, you may already know that a second recovery scenario is required to make the test run successfully.

If you haven't found the second problem that needs to be resolved, run the test again. Even if the test calls the recovery scenario to resolve the popup window, the test still fails.

The second recovery scenario you need to create is the operation to click the CANCEL button on the login window. Try this on your own, or use the steps provided in Part 3: Create a Second Recovery Scenario

Part 3: Create a Second Recovery Scenario

1. If the test you created is not already displayed on your screen, find and open it by clicking **FILE > OPEN TEST** from the QuickTest menu bar.
2. Click **RUN**. In the RUN RESULTS location tab, choose the option to save your test results to a temporary file. Click **OK** to close the window.
3. Your test should fail during the third iteration. What error message appeared that stopped the test from continuing?

4. To anticipate and correct this error, click **TOOLS > RECOVERY SCENARIO** from the QuickTest menu bar. The Recovery Manager window opens.
5. Click the **NEW SCENARIO WIZARD** icon. Review the information on the WELCOME screen then click **NEXT**.
6. Select the trigger event that caused the error (Popup window), then click **NEXT**.
7. To identify the name of the window where the error message appears, click the POINTING HAND icon. The application popup window appears on the screen.
1. Move the POINTING HAND to the application login window then click once. The name of the window appears in the WINDOW TITLE field in the Recovery Scenario Wizard.
2. Click **NEXT** to continue. The RECOVERY screen appears.

3. You do not need to make any additions to this screen. Click **NEXT**. The second recovery scenario screen appears.
4. Define the action that QuickTest should take if this error occurs during a test run by clicking the POINTING HAND icon. The application window appears on the screen.
5. Move the POINTING HAND to the application window then click once on the **CANCEL** button. The name of the button appears in the **CLICK BUTTON WITH LABEL** field in the Recovery Scenario Wizard.
6. Click **NEXT**. The **RECOVERY OPERATIONS** screen opens.
7. De-select the option to add another recovery scenario then click **NEXT**.
8. In the **TEST OPTIONS** list, choose **PROCEED TO THE NEXT STEP**. This enables the test to move to the next step after taking action to resolve the popup window error. Click **NEXT**.
9. Type a new name for this second scenario. Click **NEXT**. The **FINISH** screen opens.
10. Check the box to **ADD SCENARIO TO CURRENT TEST**. Click **FINISH**.

The name of your scenario should now appear in the Recovery Manager window.

Part 4: Check the Test Settings and Run the Test

1. To make sure that both scenarios are linked to your test, click **TEST > SETTINGS** from the QuickTest menu bar. The TEST SETTINGS window opens.
2. Click the **RECOVERY** tab.
3. Check to make sure that two scenarios appear in the Recovery tab as shown in Figure 7-6. Use the + or X icons to add or delete a scenario.

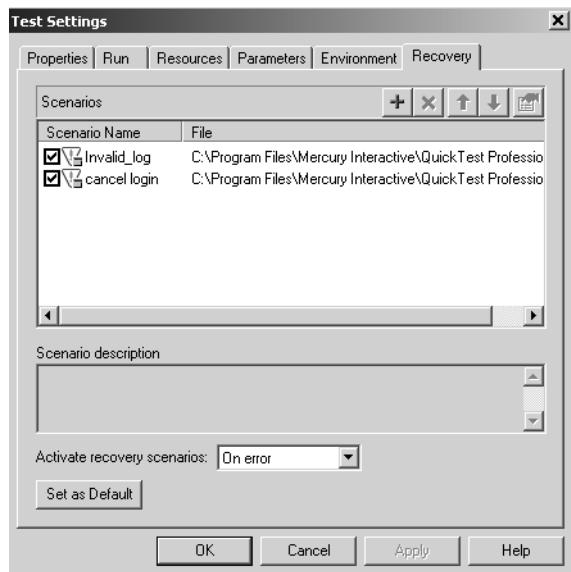


Figure 7-6

4. Click **OK**.
5. Click **RUN** to playback the test. The test should succeed.

QTP80INT-CBT-STUDENT-01A

QUICKTEST Pro 8.0

V 1.0

Table of Contents

1.0	Introduction to QuickTest.....	1
1.1	Over view of Quick Test Pro 8.0	1
1.1.1	QuickTest Pro 8.0 Environment Support	1
1.1.2	QuickTest Pro 8.0 Configurations	2
2.0	Record and Playback	3
2.1	Create and Execute Basic Scripts	3
2.1.1	Recording Tests	3
2.1.2	Running a Test.....	5
2.2	Understand Recording Levels.....	5
2.2.1	Standard Recording	6
2.2.2	Analog Recording	6
2.2.3	Low Level Recording	6
2.3	Understand QuickTest Results	6
3.0	How QuickTest identifies objects	10
3.1	Object Identification.....	10
3.1.1	Object Identification While Recording.....	10
3.1.2	Object Identification During Test Run	10
3.2	Object Repository Introduction.....	10
3.2.1	Identifying the Object	11
3.2.2	Viewing the Object's Properties	11
3.3	Use the Object Spy	12
3.3.1	To view object properties:	12
3.3.2	To view object methods:	13
4.0	Synchronization.....	15
4.1	Synchronizing Your Tests	15
4.2	Options to Synchronize Tests	15
4.2.1	4.2.1 Inserting Synchronization Point	15
4.2.2	4.2.2 Adding Exist and Wait Statements	16
4.2.3	Global synchronization Settings.....	16
4.3	Transactions.....	16
4.3.1	Inserting Transactions.....	16
4.3.2	Ending Transactions	17
5.0	Checkpoints	18
5.1	About Checkpoints	18
5.2	Adding Checkpoints to a test	18
5.2.1	To add checkpoints while recording:	18
	From Menu bar.....	18
5.2.2	To add a checkpoint while editing your test:	18
5.3	Types of Checkpoints	18
5.3.1	QuickTest Professional Checkpoint Types	18
5.3.2	Creating a Standard Checkpoint.....	19
5.3.3	Creating a Text Checkpoint	19
5.4	Use regular expressions	19
5.4.1	To define a constant property value as a regular expression:	20
5.4.2	To parameterize a property value using regular expressions:	20
5.4.3	To define a regular expression in an object checkpoint:	20
5.4.4	Common options to create regular expressions	20
6.0	Creating Tests with Multiple Actions	22
6.1	Benefits of Test Modularity	22

6.2	Creating Tests with Multiple Actions	22
6.2.1	Creating New Actions	22
6.2.2	Inserting Existing Actions	23
6.2.3	Nesting Actions	23
6.2.4	Splitting Actions	24
6.3	Miscellaneous	24
6.3.1	Setting Action Properties	24
6.3.2	Sharing Action Information	24
6.3.3	Exiting an Action	24
6.3.4	Removing Actions from a Test	24
6.3.5	Renaming Actions	24
6.3.6	Action Template	24
7.0	Data Driving a Test	26
7.1	Parameterize tests	26
7.1.1	Parameterize test Manually	26
7.1.2	DataTable Parameters	26
7.1.3	Using Environment Variable Parameters	26
7.2	Create data-driven tests	27
7.3	Local and Global Data Tables	27
7.3.1	Using the Data Driver to Parameterize Your Test	28
8.0	Working with Data Tables	30
8.1	Introduction	30
8.2	Working with Global and Action Sheets	30
8.3	Editing and Saving Data Table	30
8.4	Importing Data from a Database	31
8.5	Using Formulas in the Data Table	31
8.6	Using Data Table Scripting Methods	32
9.0	Output and Correlation	33
9.1	About Outputting Values	33
9.1.1	9.1.1 Creating Page Output Values	33
9.1.2	9.1.2 Creating Text Output Values	33
9.1.3	9.1.3 Creating Standard Output Values	33
9.1.4	9.1.4 Creating Image Output Values	34
9.1.5	9.1.5 Creating XML Output Values	34
9.1.6	9.1.6 Creating Table Output Values	34
9.1.7	9.1.7 Creating Database Output Values	34
9.2	Capture and Reuse Run Time data	34
9.2.1	9.2.1 Adding a Standard Output Value	34
9.2.2	9.2.2 Creating Image Output Values	34
9.2.3	9.2.3 Creating Table Output Values	35
10.0	Alternatives to Standard Recording	36
10.1	Analog Recording	36
10.1.1	10.1.1 Analog Recording	36
10.1.2	10.1.2 Recording in Analog Mode	36
10.2	Low-Level Recording	37
10.2.1	10.2.1 Recording in Low-Level mode	37
10.3	Configuring Web Event Recording	38
10.3.1	10.3.1 To set Web Event Recording Configuration:	38
10.4	Define a Virtual Object	38
10.4.1	10.4.1 To define a virtual object:	38
11.0	Introduction to the Expert View	41
11.1	Object Model in the Expert View	41
11.2	Using QuickTest Professional's online books	41

12.0 Working in the Expert View	43
12.1 VBScript Language Overview	43
7.1.16 12.1.1 VBScript Data Types.....	43
7.1.17 12.1.2 VBScript Variables	43
7.1.18 12.1.3 VBScript Constants.....	44
7.1.19 12.1.4 VBScript Operators	44
7.1.20 12.1.5 Using Conditional Statements	45
7.1.21 12.1.6 Looping Through Code	46
7.1.22 12.1.7 VBScript Procedures.....	47
12.2 Working with the Data Table Object	48
7.1.23 12.2.1 AddSheet Method	48
7.1.24 12.2.2 DeleteSheet Method	48
7.1.25 12.2.3 Export Method.....	48
7.1.26 12.2.4 ExportSheet Method	48
7.1.27 12.2.5 GetCurrentRow Method	48
7.1.28 12.2.6 GetRowCount Method	48
7.1.29 12.2.7 GetSheet Method.....	49
7.1.30 12.2.8 GetSheetCount Method	49
7.1.31 12.2.9 Import Method	49
7.1.32 12.2.10 ImportSheet Method	49
7.1.33 12.2.11 SetCurrentRow Method	49
7.1.34 12.2.12 SetNextRow Method	49
7.1.35 12.2.13 SetPrevRow Method	49
7.1.36 12.2.14 GlobalSheet Property.....	49
7.1.37 12.2.15 LocalSheet Property	49
7.1.38 12.2.17 RawValue Property	49
7.1.39 12.2.18 Value Property	50
12.3 Working with TextUtil Object.....	50
7.1.40 12.3.1 GetText Method	50
7.1.41 12.3.2 GetTextLocation Method.....	50
12.4 Working with Reporter Objects	50
7.1.42 12.4.1 ReportEvent Method	50
7.1.43 12.4.2 Filter Property	50
7.1.44 12.4.3 ReportPath Property	50
13.0 Object Recognition and Smart Identification	51
13.1 Object Repository Custom Configuration	51
13.2 Introduction to Smart Identification	51
13.2.1 13.2.1 Base filter properties	52
13.2.2 13.2.2 Optional filter properties	52
13.3 Understanding the Smart Identification Process.....	52
13.4 Smart Identification Configuration.....	52
14.0 Enhance Test Cases with Descriptive Programming	54
14.1 Interact with Test Objects not stored in the Object Repository.....	54
14.1.1 14.1.1 Entering Programmatic Description Directly into Test Statements	54
14.1.2 14.1.2 Using Description Objects for Programmatic Descriptions	55
14.1.3 14.1.3 Retrieving ChildObjects	55
14.1.4 14.1.4 Using Programmatic Descriptions for the WebElement Object	56
14.1.5 14.1.5 Using the Index Property in Programmatic Descriptions	56
14.2 Access Dynamic Objects during run-time	56
14.2.1 14.2.1 Retrieving Run-Time Object Properties	56
14.2.2 14.2.2 Activating Run-Time Object Methods	57
15.0 Enhance Test Cases with User-Defined Functions	58
15.1 Utilize external Windows API functions in Test Cases	58
15.1.1 15.1.1 Extern Object	58
15.2 Create QuickTest user-defined functions	60

16.0 Database Verification	62
16.1 Review of database concepts	62
16.1.1 Understanding relational tables	62
16.1.2 About SQL.....	63
16.1.3 Using SQL to interact with a database	63
16.1.4 Connection String.....	63
16.2 How to add a database checkpoint in QuickTest.....	64
17.0 Recovery Manager and Scenarios	70
18.0 Scripting in Real Time Environments	96
18.1 QuickTest Pro Coding Standards & Best Practices.....	96
18.1.1 Introduction:	96
18.2 Naming Conventions.....	96
18.2.1 Local scope variables	96
18.2.2 Global scope variables.....	96
18.2.3 Constants	97
18.2.4 Functions/Actions.....	97
18.2.5 Reusable Actions	97
18.2.6 Scripts	97
18.2.7 Function Libraries.....	97
18.2.8 Object Repository Files	98
18.3 Coding Rules.....	98
18.3.1 Commenting Code	98
18.3.2 Formatting Code	100
18.3.3 Using Shared Object Repository.....	101
18.3.4 Using Relative paths	101
18.3.5 Using Global Variables	101

1.0 Introduction to QuickTest

1.1 Over view of Quick Test Pro 8.0

QuickTest Pro is Mercury Imperative's functional enterprise testing tool.

QuickTest Professional is a fresh approach to automated software and application testing .It is designed to provide a robust application verification solution without the need for advanced technical or programming

QuickTest Pro is similar to Astra QuickTest. The key difference lies in the number of environments that QuickTest Professional supports (e.g. ERP/CRM, Java applets and applications, multiple multimedia environments, etc.).

QuickTest Professional enables you to test standard Windows applications, Web objects, ActiveX controls, Visual Basic applications, and multimedia objects on Web pages.

We Can Use QuickTest add-ins for a number of special environments (such as Java, Oracle, SAP solutions, .NET Windows and Web Forms, Siebel, PeopleSoft, Web services, and terminal emulator applications).

1.1.1 QuickTest Pro 8.0 Environment Support

Windows Applications (MFC) • Visual Basic • Java • ActiveX	Enterprise Applications • SAP • Oracle • PeopleSoft • Siebel	Web Technologies HTML • DHTML • JavaScript
Browsers IE • Netscape • AOL	Emerging Technologies • .Net Winforms, Webforms, Web services • J2EE Web services • XML, WSDL, UDDI	Terminal Emulators • 3270 • 5250 • VT100
Server Technologies • Oracle • Microsoft • IBM • BEA • ODBC • COM/COM+	Multimedia • RealAudio/RealVideo • Windows Media Player • Flash	Languages • European • Japanese • Chinese (traditional and simplified) • Korean

1.1.2 QuickTest Pro 8.0 Configurations

QuickTest Pro	QuickTest Pro Add-ins
Web Environments IE NS AOL ActiveX XML DHTML HTML	.Net Add-in Winforms, Webforms, Net Controls Java Add-in JDK 1.1-1.4.2
Client/Server Windows Win32/MFC Visual Basic	Terminal Emulator Add-in 3270,5250,vt100 MySAP Add-in SAP GUI, Web, Portals 6.2
Operating Systems Windows 98, 2000, NT, ME, XP	Oracle Add-in 11i PeopleSoft Add-in 8.0-8.8 Siebel Add-in 7.0 & 7.5 Webservices Add-in WSDL,.Net,J2EE

2.0 Record and Playback

2.1 Create and Execute Basic Scripts

2.1.1 Recording Tests

1. Start QuickTest and open a new test.
 - If QuickTest is not currently open, choose Start > Programs > QuickTest Professional > QuickTest Professional.

In the Add-in Manager, confirm that the Web Add-in is selected, and clear all other add-ins. Click OK to close the Add-in Manager and open QuickTest.

Note: While QuickTest loads your selected add-ins, the QuickTest splash screen is displayed. This may take a few seconds. If the Welcome window opens, click Blank Test.

Otherwise, choose File > New, or click the New button .

A blank test opens.

- If QuickTest is already open, check which add-ins are loaded by selecting Help > About QuickTest Professional. If the Web Add-in is not loaded, you must exit and restart QuickTest. When the Add-in Manager opens, select the Web Add-in, and clear all other add-ins.

Choose File > New, or click the New button .

A blank test opens.

Note: If the Add-in Manager does not open when starting QuickTest, choose Tools > Options. In the General tab, select Display Add-in Manager on startup. When you exit and restart QuickTest, the Add-in Manager opens.

2. Start recording.
 - Choose Test > Record or click the Record button The Record and Run Settings dialog box opens.

In the Web tab, select Open the following browser when a record or run session begins.

- Choose a browser from the Type list and confirm that the URL in the Address box is for example . <http://newtours.mercuryinteractive.com>.
- Confirm that Close the browser when the test is closed is selected.
- In the Windows Applications tab, confirm that Record and run on these applications is selected, and that there are no applications listed.

This setting prevents you from inadvertently recording operations performed on various Windows applications (such as e-mail) during a recording session.

- Click OK.

QuickTest begins recording, and your browser opens to the Mercury Tours Web site.

3. Login to the Mercury Tours Web site.

In the User Name and Password boxes, type the name and password you registered with Mercury Tours.

Click Sign-In.

The Flight Finder page opens.

4. Enter flight details.

Change the following selections:

Departing From: New York

On: Dec 29

Arriving In: San Francisco

Returning: Dec 31

Service Class: Business class

Click CONTINUE to accept the other default selections. The Select Flight page opens.

Note: When entering dates while recording this test, do not click the View Calendar button, which opens a Java-based calendar. Your test will not record the date selected using this calendar because you did not load the Java Add-in for this tutorial.

To check which add-ins have been loaded, click Help > About QuickTest Professional. To change the available add-ins for your tests, you must close and reopen QuickTest Professional.

5. Select a flight.

Click CONTINUE to accept the default flight selections. The Book a Flight page opens.

6. Enter required passenger and purchase information.

Enter the required information (fields with red text labels) in the Passengers and Credit Card sections. (You may enter fictitious information.)

In the Billing Address section, select Ticketless Travel.

At the bottom of the page, click SECURE PURCHASE. The Flight Confirmation page opens.

7. Review and complete your booking.

Click BACK TO HOME. The Mercury Tours home page opens.

8. Stop recording.

In QuickTest, click Stop  on the test toolbar to stop the recording process.

You have now reserved an imaginary business class ticket from New York to San Francisco. QuickTest recorded your Web browser operations from the time you clicked the Record button until you clicked the Stop button.

Save your test.

Select File > Save or click the Save button. The Save dialog box opens to the Tests folder.

Create a folder named Tutorial, select it, and click Open.

Type Recording in the File name field.

Confirm that Save Active Screen files is selected.

Click Save. The test name (Recording) is displayed in the title bar of the main QuickTest window.

2.1.2 Running a Test

Here you will run the test you recorded.

- 1) Start QuickTest and open the Recording test.

If QuickTest is not already open, choose Start > Programs > QuickTest Professional > QuickTest Professional.

- If the Welcome window opens, click Open Existing.
- If QuickTest opens without displaying the Welcome window, choose File > Open or click the Open button.

In the Open Test dialog box, locate and select the Recording test, then click Open.

- 2) Confirm that all images are saved to the test results.

QuickTest allows you to determine when to save images to the test results. In this lesson, all images should be saved to the test results.

Choose Tools > Options and select the Run tab. In the Save step screen capture to test results option, select always.

Click OK to close the Options dialog box.

- 3) Start running your test.

Click Run or choose Test > Run. The Run dialog box opens.

Select New run results folder. Accept the default results folder name.

Click OK to close the Run dialog box.

Watch carefully as QuickTest opens your browser and starts running the test. In the browser, you can see QuickTest perform each step you recorded; a yellow arrow in the left margin of the test tree indicates the step that QuickTest is running.

2.2 Understand Recording Levels

2.2.1 Standard Recording

Records the test in terms of GUI objects

2.2.2 Analog Recording

enables you to record the exact mouse and keyboard operations you perform in relation to either the screen or the application window.

2.2.3 Low Level Recording

This mode records at the object level and records all run-time objects as Window or WinObject test objects.

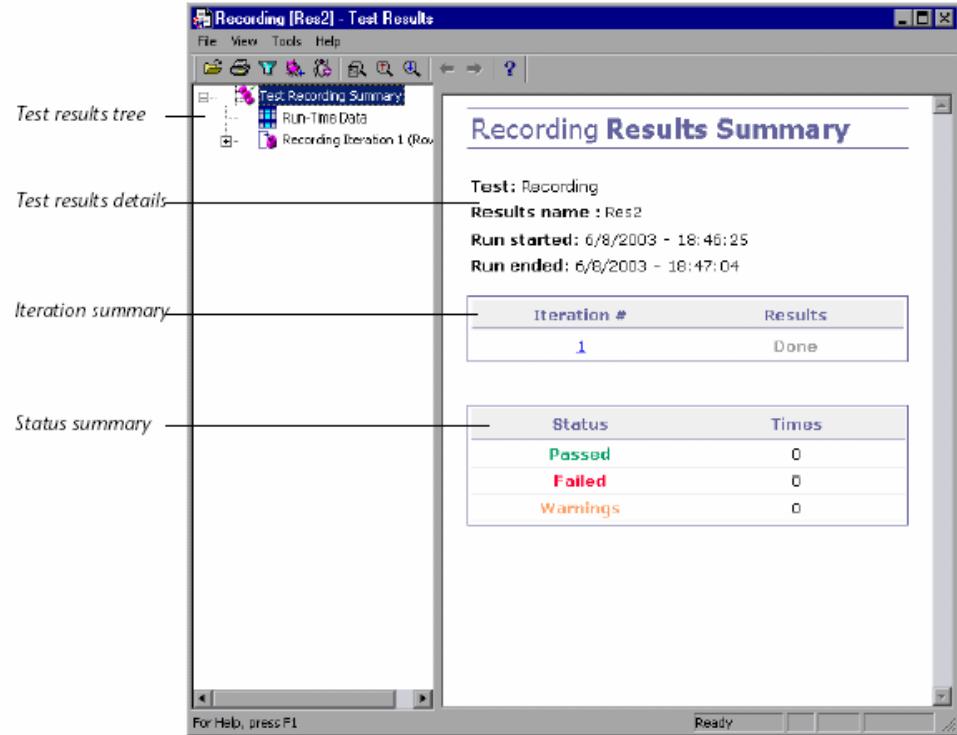
2.3 Understand QuickTest Results

When QuickTest finishes running the test, the Test Results window opens.

Initially, the Test Results window contains two panes for displaying the key elements of your test run.

- The left pane displays the test results tree, an icon-based view of the steps that were performed while the test was running. Similar to the test tree in QuickTest main screen, it is organized according to the Web pages visited during the test run and can be expanded (+) to view each step. The steps performed during the test run are represented by icons in the tree. You can instruct QuickTest to run a test or action more than once using different sets of data in each run. Each test run is called an iteration and each iteration is numbered. (The test you ran had only one iteration.)
- The right pane displays the test results details. The iteration summary table indicates which iterations passed and which failed. The status summary table indicates the number of checkpoints or reports that passed, failed, and raised warnings during the test.

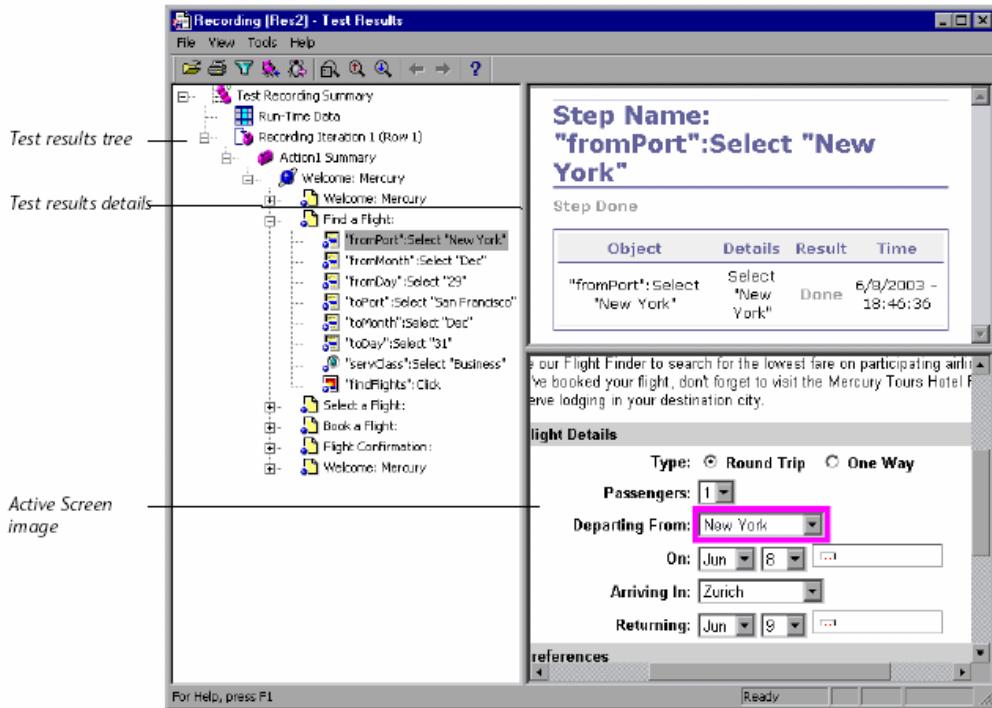
Your test run succeeded because QuickTest was able to navigate the Mercury Tours site just as the test was originally recorded. In this section, you will inspect the steps QuickTest performed when running your test, and how the application window appeared when a step was performed.



1. View the test results for a specific step.

In the test results tree, expand (+) Test Recording Summary > Recording Iteration 1 (Row 1) > Action1 Summary > Welcome Mercury > Find a Flight.

Highlight "fromPort":Select "New York" in the test results tree.



The Test Results window now contains three panes, displaying:

- the test results tree, with one step highlighted
- the test results details of the highlighted step
- the Active Screen, showing a screen capture of the Web page on which the step was performed.

When you click a page in the test results tree, QuickTest displays the corresponding page in the application view. When you click a step (an operation performed on an object) in the test results tree, the corresponding object is highlighted in the application view. In this case, the Departing From text box is highlighted.

2. Close the Test Results window.

Choose File > Exit.

Test Results Tree Symbols

- ✓ indicates a step that succeeded.
- ✗ indicates a step that failed.
- ! 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 test run was stopped before it ended.

3.0 How QuickTest identifies objects

3.1 Object Identification

3.1.1 Object Identification While Recording

Stores Object as Test Object, Determining the class it fits.

For each test object class, QuickTest always learns a list of mandatory properties. Checks whether this description is enough to uniquely identify the object. If it is not, QuickTest adds assistive properties, one by one, to the description, until it has a unique description.

3.1.2 Object Identification During Test Run

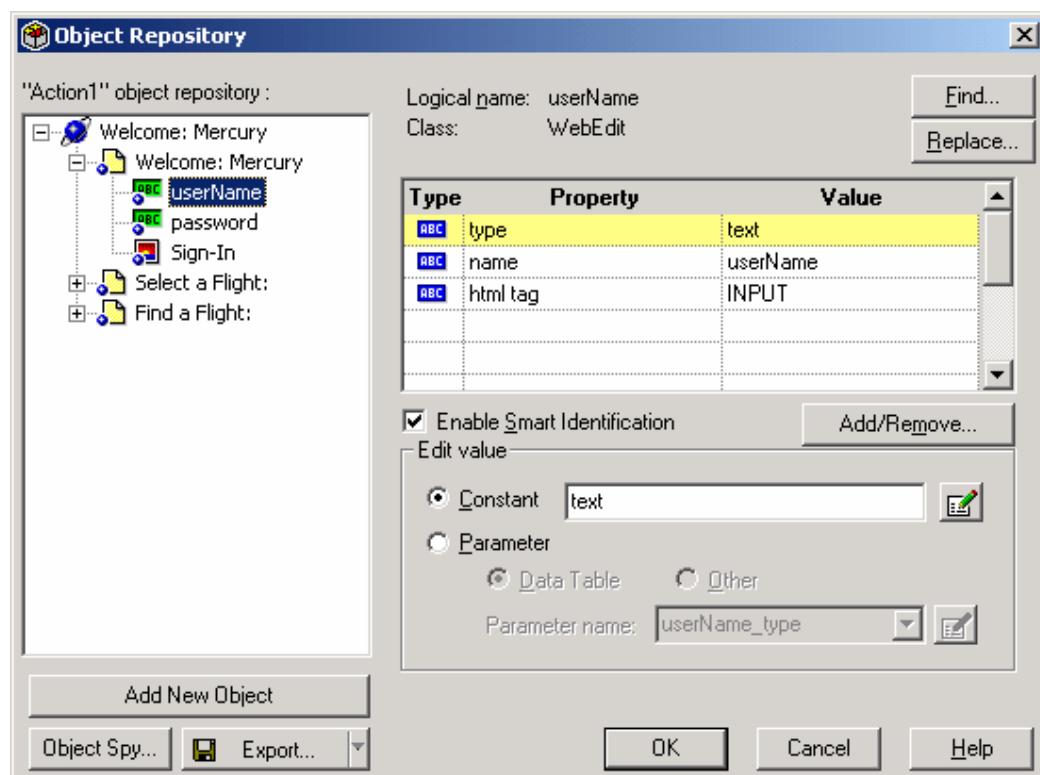
Searches for a run-time object that exactly matches the description of the test object

It expects to find a perfect match for both the mandatory and any assistive properties of test object

Uses Smart Identification mechanism to identify an object, even when the recorded description is no longer accurate.

3.2 Object Repository Introduction

The Object Repository dialog box displays a test tree of all objects in the current action or the entire test.



You can use the Object Repository dialog box to view or modify the properties of any test object in the repository or to add new objects to your repository.

3.2.1 Identifying the Object

The top part of the dialog box displays information about the object:

Information	Description
Logical name	The name that QuickTest assigns to the object.
Class	The class of the object.
Find	Opens the Find dialog box, where you can find a property or value that occurs several times in the same action.
Replace	Opens the Replace dialog box, where you can modify a property or value that occurs several times in the same action.

3.2.2 Viewing the Object's Properties

The default 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
Type	<p>The  icon indicates that the value of the property is currently a constant.</p> <p>The  icon indicates that the value of the property is currently a Data Table parameter.</p> <p>The  icon indicates that the value of the property is currently an environment variable parameter.</p> <p>The  icon indicates that the value of the property is currently a random number parameter.</p>
Property	The name of the property.
Value	The value of the property.
Enable Smart Identification	<p>Indicates whether or not QuickTest uses Smart Identification to identify this object during the test run if it is not able to identify the object using the test object description. Note that this option is available only if Smart Identification properties are defined for the object's class in the Object Identification dialog box.</p> <p>Note: When you select Disable Smart Identification during the test run in the</p>

	Run tab of the Test Settings dialog box, this option is disabled, although the setting is saved. When you clear the Disable Smart Identification during the test run check box, this option returns to its previous on or off setting after the test run.
Add/Remove	Opens the Add/Remove Properties dialog box which lists the properties that can be used to identify the object.

3.3 Use the Object Spy

Using the Object Spy, you can view the properties of any object in an open application. In addition to viewing object properties, the Object Spy also enables you to view both the run-time object methods and the test object methods associated with an object and to view the syntax for a selected method.

3.3.1 To view object properties:

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

Choose Tools > Object Spy to open the Object Spy dialog box and display the Properties tab. Alternatively, click the Object Spy button from the Object Repository dialog box.

In the Object Spy dialog box, click the pointing hand. Both QuickTest and the Object Spy are minimized so that you can point to and click on any object in the open application.

Note: 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. The window comes into the foreground. You can now point and click on the object you want. You can configure the length of time required to bring a window into the foreground in the General tab of the Options dialog box.

If the object on which you want to spy can only be displayed by performing an event (such as a right-click or a mouse-over to display a context menu), hold the Ctrl key. The pointing hand temporarily turns into a standard arrow and you can perform the event. When the object on which you want to spy is displayed, release the Ctrl key. The arrow becomes a pointing hand again.

Click the object for which you want to view properties. The Object Spy returns to focus and displays the object hierarchy tree and the properties of the object that is selected within the tree.

To view the properties of the test object, click the Test Object Properties radio button. To view the properties of the run-time object, click the Run-Time Object Properties radio button.

Tip: You can use the Object property to retrieve the values of the run-time properties displayed in the Object Spy.

You can use the GetTOProperty and SetTOProperty methods to retrieve and set the value of test object properties for test objects in your test. You can use the GetROProperty to retrieve the current property value of the objects in your application during the test run.

If you want to view properties for another object within the displayed tree, click the object in the tree.

If you want to copy an object property or value to the clipboard, click the property or value. The value is displayed in the selected property/value box. Highlight the text in the selected property/value box and use Ctrl + C to copy the text to the clipboard or right-click the highlighted text and choose Copy from the menu.

Note: If the value of a property contains more than one line, the Values cell of the object properties list indicates multi-line value. To view the value, click the Values cell. The selected property/value box displays the value with delimiters indicating the line breaks.

3.3.2 To view object methods:

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

Choose Tools > Object Spy to open the Object Spy dialog box. Alternatively, click the Object Spy button from the Object Repository dialog box.

Click the Methods tab.

Click the pointing hand. Both QuickTest and the Object Spy are minimized so that you can point to any object on the open application.

Note: If the object you want is partially hidden by another window, hold the pointing hand over the partially hidden window for a few seconds. The window comes into the foreground. You can now point and click on the object you want. You can configure this option in the Options dialog box.

If the object on which you want to spy can only be displayed by performing an event (such as a right-click or a mouse-over to display a context menu), hold the Ctrl key. The pointing hand temporarily turns into a standard arrow and you can perform the event. When the object on which you want to spy is displayed, release the Ctrl key. The arrow becomes a pointing hand again.

Click the object for which you want to view the associated methods. The Object Spy returns to focus and displays the object hierarchy tree and the run-time object or test object methods associated with the object that is selected within the tree.

To view the methods of the test object, click the Test Object Methods radio button. To view the methods of the run-time object, click the Run-Time Object Methods radio button.

Tip: You can use the Object property to activate the run-time object methods displayed in the Object Spy.

If you want to view methods for another object within the displayed tree, click the object on the tree.

If you want to copy the syntax of a method to the clipboard, click the method in the list. The syntax is displayed in the selected method syntax box. Highlight the text in the selected method syntax box and

Jagan Mohan Julooru

use Ctrl + C to copy the text to the clipboard, or right-click the highlighted text and choose Copy from the menu.

4.0 Synchronization

4.1 Synchronizing Your Tests

When you run tests, 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 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

4.2 Options to Synchronize Tests

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.
- 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 also increase the default timeout settings in the Test Settings and Options dialog boxes in order to instruct QuickTest to allow more time for certain events to occur.

4.1.1 4.2.1 Inserting Synchronization Point

- Begin recording your test.
- Display the screen or page in your application that contains the object for which you want to insert a synchronization point.
- In QuickTest choose Insert > Step > Synchronization Point. The mouse pointer turns into a pointing hand.
- Click the object in your application for which you want to insert a 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.

- The Property name list contains the test object properties associated with the object. Select the Property name you want to use for the synchronization point.
- Enter the property value for which QuickTest should wait before continuing to the next step in the test.

- 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.
- Click OK. A WaitProperty step is added to your test.

For example, if you insert a synchronization point for the Update Order button, it may look something like this:

```
Window("Flights").WinButton("Update Order").WaitProperty "enabled", 1, 3000
```

4.1.2 4.2.2 Adding Exist and Wait Statements

You can use **Exist** and/or **Wait** statements in the Expert View 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.

Example following statement returns whether Flights Table is displayed:

```
y=Window("Flight Reservation").Dialog("Flights Table").Exist
```

Wait statements instruct QuickTest to wait a specified amount of time before proceeding to the next.

Example following statement waits for 10 seconds:

```
Wait (10)
```

4.1.3 Global synchronization Settings

Modifying Timeout Values

- To modify the maximum amount of time that QuickTest waits for an object to appear, change the Object Synchronization Timeout (Test > Settings > Run tab).
- To modify the amount of time that QuickTest waits for a Web page to load, change the Browser Navigation Timeout (Test > Settings > Web tab).

4.3 Transactions

The time taken by a section of the test to run can be measured by defining **Transaction**.

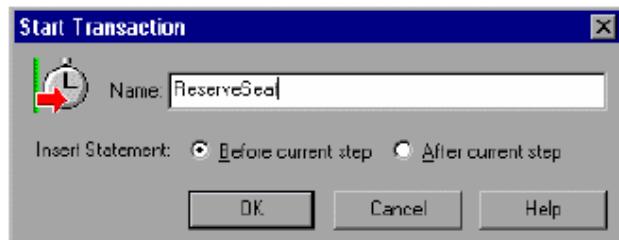
A transaction represents the business process that you are interested in measuring. You define transactions within your test by enclosing the appropriate sections of the test with **start** and **end** transaction statements.

4.3.1 Inserting Transactions

You define the beginning of a transaction in the Start Transaction dialog box.

To insert a transaction:

- In the test tree, click the step where you want the transaction timing to begin. The page is displayed in the Active Screen tab.
- Click the Start Transaction button or choose Insert > Start Transaction. The Start Transaction dialog box opens.



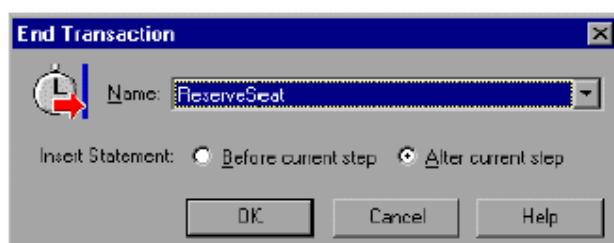
- Enter a meaningful name in the Name box.
- Decide where you want the transaction timing to begin:
 - To insert a transaction before the current step, select Before current step.
 - To insert a transaction after the current step, select After current step.
- Click OK. A tree item with the Start Transaction icon is added to the test tree.

4.3.2 Ending Transactions

You define the end of a transaction in the End Transaction dialog box.

To end a transaction:

- In the test tree, click the step where you want the transaction timing to end. The page opens in the Active Screen.
- Click the End Transaction button or choose Insert > End Transaction. The End Transaction dialog box opens.



- The Name box contains a list of the transaction names you defined in the current test. Select the name of the transaction you want to end.
- Decide where to insert the end of the transaction:
 - To insert a transaction before the current step, select Before current step.
 - To insert a transaction after the current step, select After current step.
- Click OK. A tree item with the End Transaction icon is added to the test tree.

5.0 Checkpoints

5.1 About Checkpoints

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 Web site or application is functioning correctly.

5.2 Adding Checkpoints to a test

There are several ways to add checkpoints to your tests.

5.2.1 To add checkpoints while recording:

We can add checkpoints while recording the test. Use the commands on the **Insert** menu, or click the arrow beside the **Insert Checkpoint** button on the Test toolbar. This displays a menu of checkpoint options that are relevant to the selected step in the test tree.

From Menu bar

Use the commands on the Insert menu, or click the arrow beside the Insert Checkpoint button on the Test toolbar. This displays a menu of checkpoint options that are relevant to the selected step in the test tree.

5.1.2 To add a checkpoint while editing your test:

5.2.2.1 From Test Tree

Right-click the step in the test tree where you want to add the checkpoint and choose Insert Standard Checkpoint.

5.2.2.2 From the Active Screen

Right-click any object in the Active Screen and choose Insert Standard Checkpoint. This option can be used to create checkpoints for any object in the Active Screen (even if the object is not part of any step in your test tree).

5.3 Types of Checkpoints

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 Web site or application is functioning correctly.

5.3.1 QuickTest Professional Checkpoint Types

Checkpoint Type	Description
Standard Checkpoint	Checks values of an object's properties
Image Checkpoint	Checks the property values of an image
Table Checkpoint	Checks information in a table
Page checkpoint	Checks the characteristics of a Web page

Text / Text Area Checkpoint	Checks that a text string is displayed in the appropriate place in a Web page or application window
Bitmap Checkpoint	Checks an area of a Web page or application after capturing it as a bitmap
Database Checkpoint	Checks the contents of databases accessed by an application or Web site
Accessibility Checkpoint	Identifies areas of a Web site to check for Section 508 compliancy
XML Checkpoint	Checks the data content of XML documents

5.3.2 Creating a Standard Checkpoint

- Click the Insert Checkpoint toolbar button  or choose Insert > Checkpoint > Standard Checkpoint.
- Click the object you want to check. The Select an Object dialog box opens.
- Select the item you want to check from the displayed object tree.
- Click OK. The Checkpoint Properties dialog box opens.
- Specify the settings for the checkpoint.
- Click OK to close the dialog box.

5.3.3 Creating a Text Checkpoint

- Display the page, window, or screen containing the text you want to check.
- Choose Insert > Checkpoint > Text Checkpoint,  or click the arrow next to the Insert Checkpoint button  and Choose Text Checkpoint.
- The QuickTest window is minimized and the mouse pointer turns into a pointing hand.
- Click the text string for which you want to create the checkpoint. If the area you defined is associated with more than one object, the Object Selection–Text Checkpoint Properties dialog box opens. Select the required object
- The Text Checkpoint Properties dialog box opens.
- Specify the checkpoint settings.
- Click OK to close the dialog box.

5.4 Use regular expressions

Regular expressions enable QuickTest to identify objects and text strings with varying values.

You can use regular expressions when:

- Defining the property values of an object
- Parameterize a step
- Creating checkpoints with varying values

For example, if a window titlebar's name changes according to a file name, you can use a regular expression to identify a window whose titlebar has the specified product name, followed by a hyphen, and then any other text.

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. When one of these special characters is preceded by a backslash (\), QuickTest searches for the literal character.

5.4.1 To define a constant property value as a regular expression:

- Open the Object Properties dialog box for the object either from test tree or from Active Screen or from Object Repository
- In the Property column, select the property you want to set as a regular expression.
- In the Edit value section, click Constant.
- Click the Edit Constant Value Options button. The Constant Value Options dialog box opens.
- Select the Regular Expression check box.
- In the Value box, enter the regular expression syntax for the string.
- Click OK to close the Constant Value Options dialog box.
- Click OK to save and close the Object Properties

5.4.2 To parameterize a property value using regular expressions:

Covered in Data Driving a Test

5.4.3 To define a regular expression in an object checkpoint:

- Display the page, window, or screen containing the text you want to check.
- Choose Insert > Checkpoint > Text Checkpoint, or click the arrow next to the Insert Checkpoint button and choose Text Checkpoint.
- Click the text string for which you want to create the checkpoint.
- Select the object for which you are creating the checkpoint. The Text Checkpoint Properties dialog box opens.
- In the Edit value section, click Constant.
- Click the Edit Constant Value Options button. The Constant Value Options dialog box opens.
- Select the Regular Expression check box.
- In the Value box, enter the regular expression syntax for the string.
- Click OK to close the Constant Value Options dialog box.
- Click OK to save and close Text Checkpoint Properties dialog box.

5.4.4 Common options to create regular expressions.

.	Matching Any Single Character
[xy]	Matching Any Single Character in a List
[^xy]	Matching Any Single Character Not in a List
[x-y]	Matching Any Single Character within a Range

*	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
\w	Matching Any AlphaNumeric Character Including the Underscore
\W	Matching Any Non-AlphaNumeric Character

You can combine regular expression operators in a single expression to achieve the exact search criteria you need.

6.0 Creating Tests with Multiple Actions

6.1 Benefits of Test Modularity

- Makes code reusable.
- Scripts are easy to maintain.
- Scripts are efficient.
- Saves development time.

6.2 Creating Tests with Multiple Actions

You can divide your test into multiple actions by creating new actions or by inserting existing actions. There are three kinds of actions:

- **Non-reusable action**—an action that can be used only in the test in which it was created, and only once.
- **Reusable action**—an action that can be called multiple times by the test in which it was created (the local test) as well as by other tests.
- **External action**—a reusable action created in another test. External actions are read-only in the calling test. They can be modified only in the test in which they were created.

6.2.1 Creating New Actions

You can add new actions to your test during a recording session or while designing your test.

You can add the action as a top-level action, or you can add the action as a sub-action (or nested action) of an existing action in your test.

To create a new action in your test:

If you want to insert the action within an existing action, click the step after which you want to insert the new action.

- Choose Insert > New Action or click the New Action button. The Insert New Action dialog box opens.
- Type a new action name or accept the default name.
- If you wish, add a description of the action. You can also add an action description at a later time in the Action Properties dialog box.
- Select Reusable Action if you want to make the action reusable. You can also set or modify this setting at a later time in the Action Properties dialog box.
- Decide where to insert the action and select At the end of the test or After the current step.
- Click OK.

A new action is added to your test and is displayed at the bottom of the test tree or after the current step. You can move your action to another location in your test by dragging it to the desired location.

6.2.2 Inserting Existing Actions

You can insert an existing action by inserting a copy of the action into your test, or by inserting a call to the original action.

6.2.2.1 Inserting Copies Actions

When you insert a copy of an action into a test, the action is copied in its entirety, including checkpoints, parameterization, and the corresponding action tab in the Data Table. The action is inserted into the test as an independent, non-reusable action

Once the action is copied into your test, you can add to, delete from, or modify the action just as you would with any other recorded 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 inserted action. You can insert copies of both reusable and non-reusable actions.

Steps to insert a copy of an action:

- Choose Insert > Copy of Action, right-click the action and select Insert Copy of Action, or right-click any step and select Action > Insert Copy. The Insert Copy of Action dialog box opens.
- Type a meaningful name for the action in the New action name box and give action description
- Specify where to insert the action : At the end of the test or After the current step.
- Click OK. The action is inserted into the test as an independent, nonreusable action.

6.2.2.2 Inserting Call to Actions

You can insert a call (link) to a reusable action that resides in your current test (local action), or in any other test (external action).

When you insert a call to an external action, the action is inserted in read-only format. You can view the components of the action in the action tree, but you cannot modify them.

Steps to insert a call to an action:

- Choose Insert > Call to Action, right-click the action and select Insert Call to Action, or right-click any step and select Action > Insert Call. The Insert Call to Action dialog box opens.
- In the Select an action box, select the action you want to insert from the list.
- Specify where to insert the action : At the end of the test or After the current step.
- Click OK. The action is inserted into the test as a call to the original action

6.2.3 Nesting Actions

Sometimes you may want to run an action within an action. This is called *nesting*.

Nesting actions Help you maintain the modularity of your test. Enable you to run one action or another based on the results of a conditional statement.

6.2.4 Splitting Actions

You can split an existing action into two sibling actions or into parent-child nested actions.

You cannot split an action and the option is disabled

- When an external action is selected
- When the first line of the action is selected
- While recording a test
- While running a test
- When you are working with a read-only test

6.3 Miscellaneous

6.3.1 Setting Action Properties

The Action Properties dialog box enables you to modify an action name, add or modify an action description, and set an action as reusable.

6.3.2 Sharing Action Information

There are several ways to share or pass values from one action to other actions:

- Store values from one action in the global Data Table and use these values as Data Table parameters in other actions.
- Set a value from one action as a user-defined environment variable and then use the environment variable in other actions.
- Add values to a Dictionary object in one action and retrieve the values in other actions.

6.3.3 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.

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.

6.3.4 Removing Actions from a Test

We can remove Non-reusable actions, External Actions, Reusable Actions, or Calls to External or Reusable actions.

6.3.5 Renaming Actions

You can rename actions from the Tree View or from the Expert View.

6.3.6 Action Template

If you want to include one or more statements in every new action in your test, you can create an action template.

Steps to create an action template:

- Create a text file containing the comments, function calls, and other statements that you want to include in your action template.
- Save the text file as *ActionTemplate.mst* in your <QuickTest Installation Folder>\dat folder.

7.0 Data Driving a Test

7.1 Parameterize tests

You can use the parameter feature in QuickTest to enhance your tests by parameterizing values in the test. A parameter is a variable that is assigned a value from various data sources or generators.

Two different groups of parameters

- Data Table parameters - parameter is taken from data table.
- Other parameter types - parameter values are taken from random number, external user-defined file or environment variables etc.

7.1.1 Parameterize test Manually

- You can parameterize a step recorded in your test or a checkpoint added to your test.
- You can parameterize steps and checkpoints manually by opening the appropriate dialog box.
- You can parameterize a step in your test tree while recording or editing your test.

When you parameterize a step, you can parameterize an object property, a method argument, or both.

7.1.2 DataTable 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 that runs several times using the data you supply. In each *iteration*, QuickTest substitutes the constant value with a different value from the Data Table.

7.1.3 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,

There are three types of environment variables:

7.1.3.1 User-Defined Internal

User defined internal variables are the variables that you define within the test. They are saved with the test and accessible only within the test in which they were defined.

Steps to add or modify environment variable parameters:

- In the **Edit value** section of the Object Properties, Object Repository, Method Arguments, or Checkpoint Properties dialog box, click **Other** as the type of parameter you want to use.
- Click the **Edit Parameter Options** button next to the parameter type box. The Parameter Options dialog box opens.

- Select **Environment** in the **Parameter Types** box.
- Accept the default name or enter a new name to add a new user-defined internal environment parameter, or select an existing environment variable name from the **Name** box. If you select an existing internal parameter, you can modify the value.
- If you created a new parameter or selected an existing user-defined-internal parameter, enter the value for the parameter in the **Value** box.
- Select the **Regular Expression** check box if needed and Enter the regular expression
- Click **OK** to save your changes and close the dialog box.

7.1.3.2 User-Defined External

User defined external variables are the variable that you pre-defined in the active external environment variables file. You can create as many files as you want and select an appropriate file for each test. Note that external environment variable values are designated as read-only within the test.

To define these variables create an external environment variables file

To select the active external environment-variables file:

- Choose Test > Settings to open the Test Settings dialog box.
- Click the Environment tab.
- Select the Load variables and values from external file (reloaded each test run) check box.
- Use the browse button or enter the full path of the external environment- variables file you want to use with your test.

7.1.3.2 Build-in

Built-in variables, such as Test path and Operating system. They are accessible from all tests, and are designated as read-only.

7.2 Create data-driven tests

QuickTest Pro enables you to create and run tests, which are driven by data stored in table.

When you test your application, you may want to check how it performs the same operations with multiple sets of data. For example, suppose you want to check how your application responds to ten separate sets of data. You could record ten separate tests, each with its own set of data. Alternatively, you could create a data-driven test with a loop that runs ten times. In each of the ten iterations, the test is driven by a different set of data.

7.3 Local and Global Data Tables

Each action also has its own sheet in the Data Table so that you can insert data that applies only to that action. When there are parameters 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.

The Global sheet contains the data that replaces parameters in each iteration of the test. 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.

7.3.1 Using the Data Driver to Parameterize Your Test

The Data Driver enables you to quickly parameterize several (or all) objects, methods, and/or checkpoints containing the same constant value within a given action. Similar to a 'Find and Replace All' operation versus a step-by-step 'Find and Replace' process, you can choose to replace all occurrences of a selected constant value with a parameter. QuickTest can also show you each occurrence of the constant so that you can decide whether or not to parameterize the value.

To parameterize a value using the Data Driver:

- Display the action you want to parameterize.
- Choose Tools > Data Driver. The Data Driver scans the test for constants (this may take a few moments) and then the Data Driver opens.

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 only the following constants:

The argument value of each Set method

The argument value of each Select method

The value of the second argument (Value) of each SetTObjectProperty method

- 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 OK. The constant is added to the list.

- 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.

If you selected Step-by-step parameterization, click Next. The Parameterize the Selected Step screen opens.

If you selected Parameterize all, the Parameter details area is enabled in the Data Driver main screen. Select your parameterization preferences the same way that you would for an individual step.

- In the Step to parameterize area, the first step with an object property, method argument, 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.

By default, the value is parameterized as a Data Table variable. Accept the default parameterization settings or set the parameterization options you want to apply to this step.

- 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.

- 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 previous for each relevant step.
If there are no remaining steps containing the selected value, the Finished screen opens.
- 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.

8.0 Working with Data Tables

8.1 Introduction

Purpose: Insert data table parameters into the test to run it several times on different sets of data.

Iteration: Run the test on one set of data

Design-Time Data Table: This is the sheet used to store the data for the test, which is displayed at the bottom of the screen while creating and editing the test. It has the same characteristics of Microsoft excel spreadsheet that means can insert formulas within the cells.

Run-Time Data Table: This is the sheet created by quick test to hold live version of data table when you run your test. It is displayed in the Test Results window.

8.2 Working with Global and Action Sheets

There are two types of sheets within the Data Table—**Global** and **Action**. You can access the different sheets by clicking the appropriate tabs below the Data Table.

- Global Sheet data is available to all actions in your test, which is mainly used to pass parameters from one action to another.
- Action Sheet data available to only one action in your test.

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.

8.3 Editing and Saving Data Table

By default, QuickTest automatically saves the test's Data Table as an .xls file in the test folder. You can save the Data Table in another location and instruct the test to use this Data Table when running a test.

Attach External Data Table: Specify a name and location for the data table in the resources tab of the test setting dialog box.

Usage:

1. 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.

2. 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.

Note: 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 actions in the test.

Edit information in the table by typing directly into the table.

Import data into Data Table from Microsoft Excel 95, Excel 97, Excel 2000, tabbed text files (.txt), or ASCII format by using **File → Import from File** Option.

Note: Microsoft Excel 2002 is not supported.

8.4 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.

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. Once 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.
 - **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.

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.

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.

8.5 Using Formulas in the Data Table

This enables to create contextually relevant data during the test run.

Also formulas can be used as part of a checkpoint to check that objects created on-the-fly (dynamically generated) or other variable objects in the Web page or application have the values expected for a given context.

Note: When ever you compare the values, must be of the same type, i.e. integers, strings, etc.

To do this, you can use the **TEXT** and **VALUE** functions to convert values from one type to another as follows:

- **TEXT(value)** returns the textual equivalent of a numeric value, so that, for example, `TEXT(8.2)="8.2"`.
- **VALUE(string)** returns the numeric value of a string, so that, for example, `VALUE("8.2")=8.2`.

8.6 Using Data Table Scripting Methods

QuickTest provides several Data Table methods that enable you to retrieve information about the run-time Data Table and to set the value of cells in the run-time Data Table.

From a programming perspective, the Data Table is made up of three types of objects—`DataTable`, `Sheet` (sheet), and `Parameter` (column). Each object has several methods and properties that you can use to retrieve or set values.

9.0 Output and Correlation

9.1 About Outputting Values

An *output value* is a step in which one or more values are captured at a specific point in your test or 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.

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: You can insert an output value by using commands on the Insert menu or by clicking the arrow beside the **Insert Checkpoint** button  on the Test toolbar. This displays a menu of options that are relevant to the selected step in the test tree.

7.1.1 9.1.1 Creating Page Output Values

You can create a page output value from a Web page property value. When you run the test, QuickTest retrieves the current value of the property and stores it in the run-time Data Table as an output value.

For example, 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 make an output value to store the number of links on the page during each test run or iteration.

7.1.2 9.1.2 Creating Text Output Values

You can create a text output value from a text string. When you run the test, QuickTest retrieves the current value of the text string and enters it in the run-time Data Table as an output value.

QuickTest allows you to create text output values by adding one of the following to your test:

- **Text Output Value**—enables you to output the text displayed in a screen or Web page, according to specified criteria. It is supported for all environments.
- **Standard Output Value**—enables you to output an object's text property. This is the preferred way of outputting the text displayed in many Windows applications.
- **Text Area Output Value**—enables you to output the text string displayed within a defined area of a Windows-application screen, according to specified criteria. It is supported for Standard Windows, Visual Basic, and ActiveX environments.

Note: Text Area output values are only supported by the following operating systems: Windows NT, Windows 2000, and Windows XP.

7.1.3 9.1.3 Creating Standard Output Values

You can create a standard output value from an object property value. When you run the test, QuickTest retrieves the current value of the property and enters it in the run-time Data Table as an output value.

7.1.4 9.1.4 Creating Image Output Values

You can create an image output value from the property value of a Web image. When you run the test, QuickTest retrieves the current value of the property and enters it in the run-time Data Table as an output value.

7.1.5 9.1.5 Creating XML Output Values

You can create an XML output value from an element value or attribute of an XML file or Web page/frame. When you run the test, QuickTest retrieves the current value of the element or attribute and enters it in the run-time Data Table as an output value.

When you run your test, you can view summary results of the XML output value in the Test Results window. You can also view detailed results by opening the XML Output Value Results window.

7.1.6 9.1.6 Creating Table Output Values

You can create a table output value from the contents of a table cell. When you run the test, QuickTest retrieves the current value of a table cell and enters it in the run-time Data Table as an output value.

7.1.7 9.1.7 Creating Database Output Values

You can create a database output value from the contents of a database cell. When you run the test, QuickTest retrieves the current value of a database cell and enters it in the run-time Data Table as an output value.

9.2 Capture and Reuse Run Time data

7.1.8 9.2.1 Adding a Standard Output Value

You can create a standard output value from an object property value. When you run the test, QuickTest retrieves the current value of the property and enters it in the run-time Data Table as an output value.

To create a standard output value while recording:

- Choose Insert > Output Value > Standard Output Value .
The mouse pointer turns into a pointing hand.
- Click the object in your application.
If the location you clicked is associated with more than one object, the Select an Object dialog box opens.
- Select the object for which you want to specify an output value.
- Click OK.
The Output Value Properties dialog box opens.
- Specify the settings for the output value.
- Click OK to close the Output Value Properties dialog box.
- An output value step is added to your test tree.

7.1.9 9.2.2 Creating Image Output Values

You can create an image output value from the property value of a Web image. When you run the test, QuickTest retrieves the current value of the property and enters it in the run-time Data Table as an output value.

To create an image output value while recording:

- Choose Insert > Output Value > Standard Output Value .
The mouse pointer turns into a pointing hand.
- Click the image.
- If the location you clicked is associated with more than one object, the Select an Object dialog box opens.
- Select the Image item you want to specify for an output value.
- Click OK.
The Image Output Value Properties dialog box opens.
- Specify the settings for the output value.
- Click OK to close the Image Output Value Properties dialog box.
- An output value step is added to your test.

7.1.10 9.2.3 Creating Table Output Values

You can create a table output value from the contents of a table cell. When you run the test, QuickTest retrieves the current value of a table cell and enters it in the run-time Data Table as an output value.

To create a table output value while recording:

- Choose Insert > Output Value > Standard Output Value. The mouse pointer turns into a pointing hand.
- Click the table. If the location you clicked is associated with more than one object, the Select an Object dialog box opens.
Select a Table item and click OK.
- The Output Value Properties dialog box opens.
- Specify the settings for the output value.
- Click OK to close the Output Value Properties dialog box.
- An output value step is added to your test tree.

Example: Consider the following problem for flight reservation application:

Create a new reservation and verify the reservation details.

Hint: you need order number to verify the reservation details.

Steps to solve the problem:

1. Create two actions → Create New Order and Open Order.
2. Create an output value for the order number field in Create New Order Action.
3. Parameterize Order Number in Open Order Action.

10.0 Alternatives to Standard Recording

10.1 Analog Recording

7.1.11 10.1.1 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.

7.1.12 10.1.2 Recording in Analog Mode

- Click the Record button  to begin recording.
- Click the Analog Recording button  or choose Test > Analog Recording. The Analog Recording Settings dialog box opens.
- Select from the following options:
Record relative to the screen—records mouse movement or keyboard input relative to the coordinates of your screen, regardless of which application(s) are open.

Record relative to the following window—QuickTest records any mouse movement or keyboard input relative to the coordinates of the specified window.

- 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 mode.
- Click Start Analog Record. Perform the operations you want to record in analog mode.
- When you are finished and want to return to normal recording mode, click the Analog Recording button or choose Test > Analog Recording to turn off the option.

Notes:

When you record in analog mode relative to the screen, the test run will fail if your screen resolution or the screen location on which you recorded your analog steps has changed from the time you recorded.

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.

If you chose to Record relative to the screen, QuickTest inserts the RunAnalog step under a Desktop parent item.

For example: Desktop.RunAnalog "Track9"

If you chose to Record relative to the following window, QuickTest inserts the RunAnalog step under a Window parent item. For example:

Window("Microsoft Internet").RunAnalog "Track8"

The track file called by the RunAnalog method contains all your analog data and is stored with the action.

Tip: To stop an analog step in the middle of a test run, click Ctrl + Esc, then click Stop in the test toolbar.

10.2 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 tests in an environment or on an object not recognized by QuickTest or if the exact coordinates of the object are important for your test

7.1.13 10.2.1 Recording in Low-Level mode

- Click the Record button  to begin a recording session.
- Click the Low Level Recording button  or choose Test > Low Level Recording.
- When you are finished and want to return to normal recording mode, click the Low Level Recording button or choose Test > Low Level Recording to turn off the option.

Notes:

In low-level recording and your entire keyboard input and mouse clicks are recorded based on mouse coordinates. When test is run, the cursor retraces the recorded clicks.

Suppose you type the word mercury into a user name edit box and then click the Tab key while in normal recording mode. Your script are displayed as follows:

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

If you perform the same action while in low-level recording mode, the script is recorded as follows

```
Window("Microsoft Internet").WinObject("Internet Explorer_Se").Click 29,297
```

```
Window("Microsoft Internet").WinObject("Internet Explorer_Se").Type "mercury" + micTab
```

10.3 Configuring Web Event Recording

The Web Event Recording Configuration dialog box offers three standard event-configuration levels.

By default, Quick Test uses the Basic recording -configuration level.

If Quick Test does not record all the events you need, you may require a higher event-configuration level.

Level	Description
Basic (Default)	<ul style="list-style-type: none">Always records click events on standard Web objects such as images, buttons, and radio buttons.Always records the submit event within forms.Records click events on other objects with a handler or behavior connected. Records the mouse over event on images and image maps only if the event following the mouse over is performed on the same object and is dependent on the mouse over event.
Medium	Records click events on the <DIV>, , and <TD> HTML tag objects, in addition to the objects recorded in the basic level.
High	Records mouse over, mouse down, and double-click events on objects with handlers or behaviors attached, in addition to the objects recorded in the basic level.

7.1.14 10.3.1 To set Web Event Recording Configuration:

- Choose Tools > Web Event Recording Configuration. The Web Event Recording Configuration dialog box opens. Use the slider to select your preferred standard event-recording configuration.
- Click OK.

10.4 Define a Virtual Object

You can teach Quick Test to recognize any area of your application as an object by defining it as a virtual object.

Virtual objects enable you to record and run tests on objects that are not normally recognized by Quick Test.

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 logical name.

7.1.15 10.4.1 To define a virtual object:

- With Quick Test open (but not in record mode), open your Web site or application and display the object containing the area you want to define as a virtual object.
- In Quick Test, choose Tools > Virtual Objects > New Virtual Object. The Virtual Object Wizard opens. Click Next.
- Select a standard class to which you want to map your virtual object.

- Click Mark Object.
The Quick Test window and the Virtual Object Wizard are minimized. 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. Click Next.
- Click an object in the object tree to assign it as the parent of the virtual object.
- In the Identify object-using box, select how you want Quick Test to identify and map the virtual object.

If you want Quick Test to identify all occurrences of the virtual object, select parent only. Quick Test identifies the virtual object using its direct parent only, regardless of the entire parent hierarchy.

If you want Quick Test to identify the virtual object in one occurrence only, select entire parent hierarchy. Quick Test identifies the virtual object only if it has the exact parent hierarchy. Click Next.

- Specify a name and a collection for the virtual object. Choose from the list of collections or create a new one by entering a new name in the Collection name box.
- To add the virtual object to the Virtual Object Manager and close the wizard, select No and then click Finish.

Jagan Mohan Julooru

11.0 Introduction to the Expert View

11.1 Object Model in the Expert View

The Expert View provides an alternative to the Tree View for testers who are familiar with VBScript. In the Expert View, you can view the recorded test in VBScript and enhance it with programming.

The Expert View displays the steps you executed while recording your test in VBScript. After you record your test, you can increase its power and flexibility by adding recordable and non-recordable VBScript statements. You can add statements that perform operations on objects or retrieve information from your site. For example, you can add a step that checks that an object exists, or you can retrieve the return value of a method.

The objects in QuickTest are divided by environment. QuickTest environments include standard Windows objects, Visual Basic objects, ActiveX objects, Web objects, Multimedia objects, as well as objects from other environments available as external add-ins.

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

In the following example, the user enters mercury in the User Name edit box while recording. The following line is recorded in the Expert View:

```
Browser ("Mercury_Tours"). Page ("Mercury_Tours"). WebEdit ("username").Set "mercury"
```

When running the test, the **Set** method inserts the mercury text into the WebEdit object.

In the following example, the user selects **Paris** from the **Departure City** list box while recording. The following line is recorded in the Expert View:

```
Browser("Mercury Tours").Page("Find Flights").WebList("depart").Select "Paris"
```

When the test runs, the **Select** method selects **Paris** in the WebList object.

11.2 Using QuickTest Professional's online books

QuickTest Professional documentation helps you take full advantage of the capabilities of this powerful, easy-to-use, testing tool.

- QuickTest Professional User's Guide—provides step-by-step instructions on how to use QuickTest to test your applications. It describes many useful features, options, and testing procedures with guidelines and helpful examples.
- QuickTest Professional Installation Guide—explains how to install QuickTest Professional.

- QuickTest Professional Tutorial—teaches you basic QuickTest skills and shows you how to start testing your applications.
- QuickTest Professional Object Model Reference—provides access to the QuickTest Professional VBScript methods, including a description of each object, a list of the methods associated with each object, description syntax, and an example of usage for each object and method.
- QuickTest Professional Shortcut Key Reference Card—provides a list of commands that you can execute using shortcut keys.
- QuickTest Professional Automation Object Model Reference — (available from the QuickTest Professional Start menu program folder and from the QuickTest Professional **Help** menu) 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.

12.0 Working in the Expert View

12.1 VBScript Language Overview

7.1.16 12.1.1 VBScript Data Types

VBScript has only one data type called a Variant. A Variant is a special kind of data type that can contain different kinds of information, depending on how it is used.

Subtype	Description
Empty	Variant is uninitialized. Value is 0 for numeric variables or a zero-length string ("") for string variables.
Null	Variant intentionally contains no valid data.
Boolean	Contains either True or False.
Byte	Contains integer in the range 0 to 255.
Integer	Contains integer in the range -32,768 to 32,767.
Currency	-922,337,203,685,477.5808 to 922,337,203,685,477.5807.
Long	Contains integer in the range -2,147,483,648 to 2,147,483,647.
Single	Contains a single-precision, floating-point number in the range -3.402823E38 to -1.401298E-45 for negative values; 1.401298E-45 to 3.402823E38 for positive values.
Double	Contains a double-precision, floating-point number in the range -1.79769313486232E308 to -4.94065645841247E-324 for negative values; 4.94065645841247E-324 to 1.79769313486232E308 for positive values.
Date (Time)	Contains a number that represents a date between January 1, 100 to December 31, 9999.
String	Contains a variable-length string that can be up to approximately 2 billion characters in length.
Object	Contains an object.
Error	Contains an error number.

7.1.17 12.1.2 VBScript Variables

12.1.2.1 Declaring Variables

You declare variables explicitly in your script using the Dim statement, the Public statement, and the Private statement.

Variables declared with Dim at the script level are available to all procedures within the script. At the procedure level, variables are available only within the procedure.

Public statement variables are available to all procedures in all scripts.

Private statement variables are available only to the script in which they are declared.

12.1.2.2 Variable Naming Restrictions

Variable names follow the standard rules for naming anything in VBScript. A variable name:

- Must begin with an alphabetic character.
- Cannot contain an embedded period.
- Must not exceed 255 characters.
- Must be unique in the scope in which it is declared.

12.1.2.3 Scope and Lifetime of Variables

A variable's scope is determined by where you declare it. When you declare a variable within a procedure, only code within that procedure can access or change the value of that variable. It has local scope and is a procedure-level variable. If you declare a variable outside a procedure, you make it recognizable to all the procedures in your script. This is a script-level variable, and it has script-level scope.

The lifetime of a variable depends on how long it exists. The lifetime of a script-level variable extends from the time it is declared until the time the script is finished running. At procedure level, a variable exists only as long as you are in the procedure. When the procedure exits, the variable is destroyed. Local variables are ideal as temporary storage space when a procedure is executing. You can have local variables of the same name in several different procedures because each is recognized only by the procedure in which it is declared.

7.1.18 12.1.3 VBScript Constants

You create user-defined constants in VBScript using the [Const](#) statement. Using the Const statement, you can create string or numeric constants with meaningful names and assign them literal values. For example:

```
Const MyString = "This is my string."
```

```
Const MyAge = 49
```

7.1.19 12.1.4 VBScript Operators

12.1.4.1 Arithmetic Operators

Description	Symbol
Exponentiation	$^$
Unary negation	$-$
Multiplication	$*$
Division	$/$
Integer division	\backslash
Modulus arithmetic	Mod
Addition	$+$
Subtraction	$-$
String concatenation	$&$

12.1.4.2 Comparison Operators

Description	Symbol
Equality	=
Inequality	<>
Less than	<
Greater than	>
Less than or equal to	<=
Greater than or equal to	>=
Object equivalence	Is

12.1.4.3 Logical Operators

Description	Symbol
Logical negation	Not
Logical conjunction	And
Logical disjunction	Or
Logical exclusion	Xor
Logical equivalence	Eqv
Logical implication	Imp

7.1.20 12.1.5 Using Conditional Statements

The following conditional statements are available in VBScript:

12.1.5.1 If...Then...Else statement.

Example:

```
If value = 0 Then  
    AlertLabel.ForeColor = vbRed  
Else  
    AlertLabel.Forecolor = vbBlack  
End If
```

12.1.5.2 Select Case statement.

Example:

```
Select Case strMyText  
    Case "MasterCard"  
        DisplayMCLogo  
        ValidateMCAccount  
    Case "Visa"  
        DisplayVisaLogo
```

```
    ValidateVisaAccount
Case Else
    DisplayUnknownImage
    PromptAgain
End Select
```

7.1.21 12.1.6 Looping Through Code

The following looping statements are available in VBScript:

12.1.6.1 Do...Loop:

Loops while or until a condition is True.

Repeats a block of statements while a condition is True or until a condition becomes True.

Do [{While | Until} condition]

[statements]

[Exit Do]

[statements]

Loop

Or, you can use this syntax:

Do

[statements]

[Exit Do]

[statements]

Loop [{While | Until} condition]

12.1.6.2 While...Wend:

Loops while a condition is True.

Executes a series of statements as long as a given condition is True.

While condition

Version [statements]

Wend

12.1.6.3 For...Next:

Uses a counter to run statements a specified number of times.

Repeats a group of statements a specified number of times.

For counter = start To end [Step step]

[statements]

[Exit For]

[statements]

Next

For Each...Next: Repeats a group of statements for each item in a collection or each element of an array.

Repeats a group of statements for each element in an array or collection.

For Each element In group

[statements]

[Exit For]

[statements]

Next [element]

7.1.22 12.1.7 VBScript Procedures

12.1.7.1 Sub Procedures

A Sub procedure is a series of VBScript statements (enclosed by Sub and End Sub statements) that perform actions but don't return a value.

Example:

```
Sub ConvertTemp()
    temp = InputBox("Please enter the temperature in degrees F.", 1)
    MsgBox "The temperature is " & Celsius(temp) & " degrees C."
End Sub
```

12.1.7.2 Function Procedures

A Function procedure is a series of VBScript statements enclosed by the Function and End Function statements.

Example:

```
Function Celsius(fDegrees)
    Celsius = (fDegrees - 32) * 5 / 9
End Function
```

The easiest way to create a test is to begin by recording typical business processes that you perform on your application or Web site. Then, to increase your test's power and flexibility, you can add programming statements to the recorded framework. Programming statements can contain:

- *recordable* test object methods: operations that a user can perform on an application or Web site.
- *non-recordable* test object methods: operations that users cannot perform on an application or Web site. You use these methods to retrieve or set information, or to perform operations triggered by an event.
- run-time methods of the object being tested.
- various 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

Note: *Test object* methods are defined in QuickTest; *run-time* methods are defined within the object you are testing, and therefore are retrieved from them.

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.

12.2 Working with the Data Table Object

7.1.23 12.2.1 AddSheet Method

Adds the specified sheet to the run-time Data Table

Syntax: **DataTable.AddSheet(SheetName)**

Example:

Variable=DataTable.AddSheet ("MySheet").AddParameter("Time", "8:00")

7.1.24 12.2.2 DeleteSheet Method

Deletes the specified sheet from the run-time Data Table.

Syntax: **DataTable.DeleteSheet SheetID**

Example: DataTable.DeleteSheet "MySheet"

7.1.25 12.2.3 Export Method

Saves a copy of the run-time Data Table in the specified location.

Syntax: **DataTable.Export(FileName)**

Example: DataTable.Export ("C:\flights.xls")

7.1.26 12.2.4 ExportSheet Method

Exports a specified sheet of the run-time Data Table to the specified file.

Syntax: **DataTable.ExportSheet(FileName, DTSheet)**

Example: DataTable.ExportSheet "C:\name.xls" ,1

7.1.27 12.2.5 Get.CurrentRow Method

Returns the current (active) row in the run-time global data sheet.

7.1.28 12.2.6 GetRowCount Method

Returns the total number of rows in the longest column in the global data sheet or in the specified data sheet of the run-time Data Table.

Example:

rowCount = DataTable.GetSheet("MySheet").GetRowCount

7.1.29 12.2.7 GetSheet Method

Returns the specified sheet from the run-time Data Table.

Example: MyParam=DataTable.GetSheet ("MySheet").AddParameter("Time", "8:00")

7.1.30 12.2.8 GetSheetCount Method

Returns the total number of sheets in the run-time Data Table.

7.1.31 12.2.9 Import Method

Imports the specified Microsoft Excel file to the run-time Data Table.

Syntax: **DataTable.Import(FileName)**

Example: DataTable.Import ("C:\flights.xls")

7.1.32 12.2.10 ImportSheet Method

Imports a sheet of a specified file to a specified sheet in the run-time Data Table.

Syntax: **DataTable.ImportSheet(FileName, SheetSource, SheetDest)**

Example: DataTable.ImportSheet "C:\name.xls", 1, "name"

7.1.33 12.2.11 SetCurrentRow Method

Sets the specified row as the current (active) row in the run-time Data Table.

Example: DataTable.SetCurrentRow (2)

7.1.34 12.2.12 SetNextRow Method

Sets the row after the current (active) row as the new current row in the run-time Data Table.

7.1.35 12.2.13 SetPrevRow Method

Sets the row above the current (active) row as the new current (active) row in the run-time Data Table.

7.1.36 12.2.14 GlobalSheet Property

Returns the Global sheet of the run-time Data Table.

Example:

DataTable.GlobalSheet.AddParameter "Time", "5:45"

7.1.37 12.2.15 LocalSheet Property

Returns the current (active) local sheet of the run-time Data Table.

Example:

MyParam=DataTable.LocalSheet.AddParameter("Time", "5:45")

7.1.38 12.2.17 RawValue Property

Retrieves the *raw value* of the cell in the specified parameter and the current row of the run-time Data Table.

Syntax: **DataTable.RawValue ParameterID [, SheetID]**

SheetID can be the sheet name, index or `dtLocalSheet`, or
`dtGlobalSheet`.

7.1.39 12.2.18 Value Property

Retrieves or sets the value of the cell in the specified parameter and the current row of the run-time Data Table.

Syntax:**DataTable.Value(ParameterID [, SheetID])**

12.3 Working with TextUtil Object.

7.1.40 12.3.1 GetText Method

Returns the text from the specified window handle area.

Syntax

TextUtil.GetText(hWnd [, Left, Top, Right, Bottom])

7.1.41 12.3.2 GetTextLocation Method

Checks whether a specified text string is contained in a specified window area.

Syntax

TextUtil.GetTextLocation(TextToFind, hWnd, Left, Top, Right, Bottom[, MatchWholeWordOnly])

12.4 Working with Reporter Objects

7.1.42 12.4.1 ReportEvent Method

Reports an event to the Test Report.

Syntax

Reporter.ReportEvent EventStatus, ReportStepName, Details [, in]

EventStatus – micPass, micFail, micDone, micWarning

7.1.43 12.4.2 Filter Property

Retrieves or sets the current mode for displaying events in the Test Results.

Syntax

To retrieve mode setting: `CurrentMode = Reporter.Filter`

To set the mode: `Reporter.Filter = NewMode`

Mode - 0 or rfEnableAll, 1 or rfEnableErrorsAndWarnings,

2 or rfEnableErrorsOnly, 3 or rfDisableAll

7.1.44 12.4.3 ReportPath Property

Retrieves the folder path in which the current test's Test Results are stored.

Syntax `Path = Reporter.ReportPath`

13.0 Object Recognition and Smart Identification

13.1 Object Repository Custom Configuration

You use the main screen of the Object Identification dialog box to set mandatory and assistive recording 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.

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

- Choose **Tools > Object Identification**. The Object Identification dialog box opens.
- 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.
- Select the properties you want to include in the Mandatory Properties list and/or clear the properties you want to remove from the list.
- Enter a valid property in 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.
- Click **OK** to close the Add/Remove Properties dialog box. The updated set of mandatory properties is displayed in the **Mandatory Properties** list.
- In the **Assistive Properties** list, click **Add/Remove**. The Add/Remove Properties dialog box for assistive properties opens.
- Select the properties you want to include in the assistive properties list and/or clear the properties you want to remove from the list.
- Enter a valid property in 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.
- Click **OK** to close the Add/Remove Properties dialog box. The properties are displayed in the Assistive Properties list.
- Use the up and down arrows to set your preferred order for the assistive properties. When you record a test 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.

13.2 Introduction to Smart Identification

When QuickTest uses the recorded description to identify an object, it searches for an object that matches every one 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 recorded object description, or if it finds more than one object that fits the description, then QuickTest ignores the recorded 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, and thus, if configured logically, a Smart Identification definition can probably help QuickTest identify an object, if it is present, even when the recorded description fails.

The Smart Identification mechanism uses two types of properties:

13.2.1 13.2.1 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.

13.2.2 13.2.2 Optional filter properties

Other properties that can help identify objects of a particular class as they are unlikely to change on a regular basis, but which can be ignored if they are no longer applicable.

13.3 Understanding the Smart Identification Process

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

1. QuickTest “forgets” the recorded 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 property list.
2. From that list of objects, QuickTest filters out any object 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 recorded description plus the ordinal identifier to identify the object

13.4 Smart Identification Configuration

1. Choose **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.
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.
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.
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 Properties** list until it filters the object candidates down to one object.

14.0 Enhance Test Cases with Descriptive Programming

14.1 Interact with Test Objects not stored in the Object Repository

You can also instruct QuickTest to perform methods on objects without referring to the Object Repository, without referring to the object's logical 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 a method.

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 in order to perform the same operation on several objects with certain identical properties, or in order to perform an operation on an object whose properties match a description that you determine dynamically during the test run.

There are two types of programmatic descriptions.

- You can either list the set of properties and values that describe the object directly in a test statement,
- or you can add a collection of properties and values to a description object, and then enter the description object name in the statement.

14.1.1 14.1.1 Entering Programmatic Description Directly into Test Statements

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

The general syntax is:

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

TestObject—the test object class.

PropertyName:=PropertyValue—the test object property and its value. Each property:=value pair should be separated by commas and quotation marks.

For Example: `Window("Text:=Myfile.txt - Notepad").Move 50, 50`

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

For Example:

```
Set MyWin = Window("Text:=Myfile.txt - Notepad")
MyWin.Move 50, 50
```

14.1.2 14.1.2 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** object in place of a logical name in a test statement.

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

Set MyDescription = Description.Create()

Once 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 test run. This enables you to determine which, and how many properties to include in the object description in a dynamic way during the test run.

Once you have filled the **Properties** collection with a set of **Property** objects (properties and values), you can specify the **Properties** object in place of a logical 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
```

14.1.3 14.1.3 Retrieving ChildObjects

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. In order 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.

Once 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

14.1.4 14.1.4 Using Programmatic Descriptions for the WebElement Object

The **WebElement** object enables you to perform methods on Web objects that may not fit into any other Mercury test object class. The **WebElement** test object is never recorded, but you can use a programmatic description with the **WebElement** object to perform methods on any Web object in your Web site.

For example, when you run the statement below:

```
Browser("Mercury Tours").Page("Mercury Tours").WebElement("Name:=UserName",  
"Index:=0").Click
```

or

```
set WebObjDesc = Description.Create()  
WebObjDesc("Name").Value = "UserName"  
WebObjDesc("Index").Value = "0"  
Browser("Mercury Tours").Page("Mercury Tours").WebElement(WebObjDesc).Click
```

QuickTest clicks on the first Web object in the Mercury Tours page with the name UserName.

14.1.5 14.1.5 Using the Index Property in Programmatic Descriptions

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

Note that *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")
```

14.2 Access Dynamic Objects during run-time

14.2.1 14.2.1 Retrieving Run-Time Object 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
```

14.2.2 14.2.2 Activating Run-Time Object Methods

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

```
Dim MyWebEdit
```

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

15.0 Enhance Test Cases with User-Defined Functions

15.1 Utilize external Windows API functions in Test Cases

15.1.1 15.1.1 Extern Object

Enables you to declare calls to external procedures from an external dynamic-link library(DLL)

Associated Methods

- Declare Method** : Declares references to external procedures in a dynamic-link library (DLL).

Once you use the Declare method for a method, you can use the Extern object to call the declared method.

Syntax

`Extern.Declare(RetType, MethodName, LibName, Alias [, ArgType(s)])`

<i>RetType</i>	String	Data type of the value returned by the method.
<i>MethodName</i>	String	Any valid procedure name.
<i>LibName</i>	String	Name of the DLL or code resource that contains the declared procedure.
<i>Alias</i>	String	Name of the procedure in the DLL or code resource. Note: DLL entry points are case sensitive. Note: If <i>Alias</i> is an empty string, <i>MethodName</i> is used as the Alias.
<i>ArgType(s)</i>	String	A list of data types representing the data types of the arguments that are passed to the procedure when it is called. Note: For out arguments, use the micByRef flag.

Declare Data Types

The following data types can be used for the Extern.Declare *RetType* and *ArgType(s)* arguments.

Type	Description
micVoid	=0 ; void (RetType only)
micInteger	=2 ; int
micLong	=3 ; long
micFloat	=4 ; float

micDouble	=5 ; double
micString	=8 ; CHAR*
micDispatch	=9 ; IDispatch*
micWideString	=18 ; WChar*
micChar	=19 ; char
micUnknown	=20 ; IUnknown
micHwnd	=21 ; HWND
micVPtr	=22 ; void*
micShort	=23 ; short
micWord	=24 ; WORD
micDWord	=25; DWORD
micByte	=26 ; BYTE
micWParam	=27 ; WPARAM
micLParam	=28 ; LPARAM
micLResult	=29 ; LRESULT
micByRef	=0X4000 ; out
micUnsigned	=0X8000 ; unsigned
micUChar	=micChar micUnsigned ; unsigned char
micULong	=micLong micUnsigned ; ; unsigned long
micUShort	=micShort micUnsigned ; ; unsigned short
micUInteger	=micInteger micUnsigned ; ; unsigned int

Example

The following example uses the Extern.Declare and Extern.<declared method> methods to change the title of the Notepad window.

'Declare FindWindow method

```
Extern.Declare micHwnd, "FindWindow", "user32.dll", "FindWindowA", micString, micString
```

```
'Declare SetWindowText method
Extern.Declare micLong, "SetWindowText", "user32.dll", "SetWindowTextA", micHwnd, micString

'Get HWND of the Notepad window
hwnd = Extern.FindWindow("Notepad", vbNullString)

if hwnd = 0 then
    MsgBox "Notepad window not found"
end if

'Change the title of the notepad window
res = Extern.SetWindowText(hwnd, "Textpad")
```

15.2 Create QuickTest user-defined functions

In addition to the test objects, methods, and functions supported by the QuickTest Test Object Model, you can define your own VBScript functions subroutines, classes, modules, etc., and then call them from your test.

Vbscript function libraries can be written in any text editor. We have to save the file with .vbs extension. Syntax for writing vbscript function in vbscript file is as follows.

```
[Public |Default] | Private] Function name [(arglist)]
[statements]
[name = expression]
[Exit Function]
[statements]
[name = expression]
End Function
```

Arguments

Public Indicates that the Function procedure is accessible to all other procedures in all scripts.

Default Used only with the Public keyword in a Class block to indicate that the Function procedure is the default method for the class. An error occurs if more than one Default procedure is specified in a class.

Private Indicates that the Function procedure is accessible only to other procedures in the script where it is declared or if the function is a member of a class, and that the Function procedure is accessible only to other procedures in that class.

name Name of the Function; follows standard variable naming conventions.

arglist List of variables representing arguments that are passed to the Function procedure when it is called. Commas separate multiple variables.

statements Any group of statements to be executed within the body of the Function procedure.

expression Return value of the Function.

The *arglist* argument has the following syntax and parts:

[**ByVal** | **ByRef**] *varname*[*()*]

Arguments

ByVal Indicates that the argument is passed by value.

ByRef Indicates that the argument is passed by reference.

varname Name of the variable representing the argument; follows standard variable naming conventions.

You can call these user defined functions from QuickTest as follows:

- Specify the default library files for all new tests in the Test Settings dialog box (**Test > Settings > Resources tab**).
- call a function contained in a VBScript file directly from any action using the **ExecuteFile** function.

To execute an externally-defined function:

- Create a VBScript file using standard VBScript syntax
- Store the file in any folder that you can access from the machine running your test.
- Add an **ExecuteFile** statement to an action in your test or in an associated library file using the following syntax:

ExecuteFile *FileName*

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

- Use the functions, subroutines, classes, etc., from the specified VBScript file as necessary in your action or within other functions in an associated library file.

16.0 Database Verification

16.1 Review of database concepts

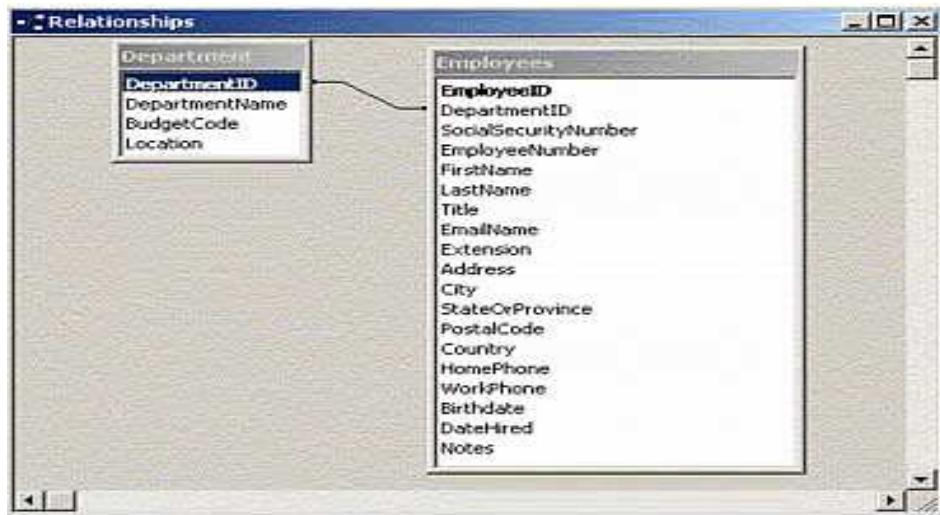
A *relational database* is a structured collection of information that is related to a particular subject or purpose, such as an inventory database or a human resources database. You use databases to manage information. Information, such as product name, cost, and on-hand inventory, is stored in a database.

Within the database, you organize the data into storage containers called tables. *Tables* are made up of columns and rows. Columns represent individual *fields* in a table. Rows represent *records* of data in a table.

Each field in the table contains one piece of information. In an employee table, for example, one column contains the employee name, another contains the employee phone number, and the address, city, state, zip, and salary are all stored in their own columns. Each record represents one set of related information. For example, an employee table might store information about one employee per row. The number of rows in a table represents the total number of table records.

16.1.1 Understanding relational tables

In a database, you can organize data in multiple tables. For example, if you manage a database for the Human Resource department, you might have one table that lists all the employees information and another table that lists all the departments:



Because you have multiple departments for employees, but you would not store the information about the departments in every employee row for several reasons:

- The department information is the same for each employee in a given department, however, repeating the department information for each employee is redundant. Storing redundant data takes up more disk space.
- If the department information changes, you can update one occurrence. All references to that department are updated automatically.

Storing multiple occurrences of the same data is rarely a good thing. Good relational database design separates application entities into their own tables. Key values from one table are often stored in a related table rather than repeating the information. The key value is used to join the data between the tables to return the complete set of data required.

16.1.2 About SQL

SQL (Structured Query Language) is a language that lets you communicate with databases. For example, you can use SQL to retrieve data from a database, add data to a database, delete or update records in a database, change columns in multiple rows, add columns to tables, and add and delete tables.

16.1.3 Using SQL to interact with a database

Unlike other computer languages, SQL is made up of a small number of language elements that let you interact efficiently with a database. Some of the more frequently used elements include these SQL commands:

Command	Description
SELECT	Use to retrieve (query) information in a database.
INSERT	Use to add records to a database.
UPDATE	Use to update information in a database.
DELETE	Use to delete information in a database.

16.1.4 Connection String

For connecting to database we must provide the information to connect to the database. There are two ways to provide this information.

The first way is through the use of a System DSN. A System DSN is a file containing information about a particular database. This information includes the physical location of the database on the computer, what type of database it is (SQL, Access, etc.), and other pertinent information. (*DSN stands for Data Source Name*)

the other way is to use what is called a **DSN-less connection**. This approach uglies up your connection string a bit, because you need to supply all of the important information in the connection string, since there isn't a DSN which holds that information. Access databases are the ones that typically use DSN-less connections.

Here is an example of a DSN-less connection string.

```
objConn.ConnectionString = "DBQ=C:\WebShare\MyDatabase.mdb;DRIVER={MS Access (*.mdb)}"
```

In QuickTest pro You can use database checkpoints in your test to check databases accessed by your Web site or application and to detect defects. You define a query on your database, and then you create a database checkpoint that checks the results of the query.

There are two ways to define a database query:

Use Microsoft Query. You can install Microsoft Query from the *custom installation* of Microsoft Office.

Manually define an SQL statement.

In QuickTest, you create a database checkpoint based on the results of the query you defined on a database. QuickTest captures the current information about the database and saves this information as *expected data*. A database checkpoint is inserted into the test script. This checkpoint is displayed in your test script as a **DbTable.Check CheckPoint** statement.

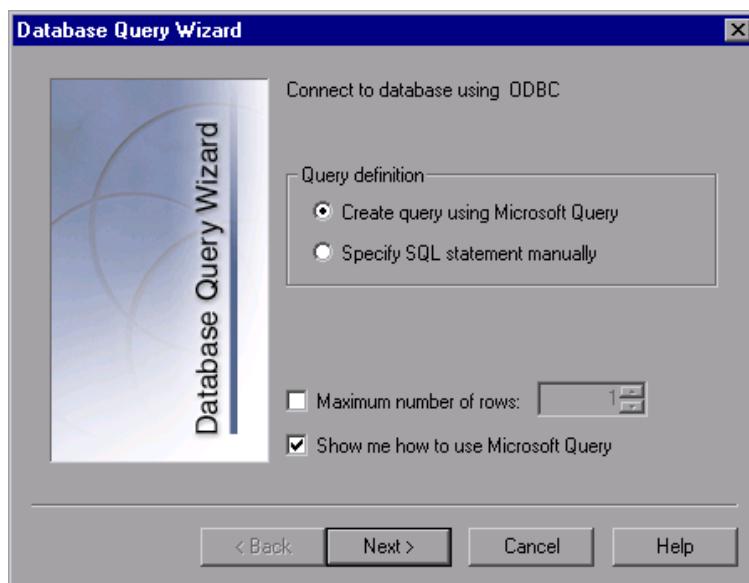
When you run the test, the database checkpoint compares the current state of the database to the expected data defined in the Checkpoint Properties dialog box. If the expected data and the current results do not match, the database checkpoint fails.

16.2 How to add a database checkpoint in QuickTest

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 Choose Insert > Checkpoint > Database Checkpoint. 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. Once 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. For additional information, see step 3.
- **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 If you chose **Create query using Microsoft Query** in the previous step, Microsoft Query opens. Choose a data source and define a query. If you chose **Specify SQL statement** in the previous step, the **Specify SQL statement** screen opens. Specify the connection string and the SQL statement, and click **Finish**.

4 The Checkpoint Properties dialog box opens. Select the checks to perform on the result set. 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.

1.3 Specifying SQL Statements

You can manually specify the database connection string and the SQL statement.

1.3.1 To specify SQL statements:

1 Choose **Specify SQL statement** in the Database Query Wizard screen. 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 Return to step 4 in the previous procedure to continue creating your database checkpoint in QuickTest.

1.4 Creating a Query in Microsoft Query

You can use Microsoft Query to choose a data source and define a query within the data source.

1.4.1 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.

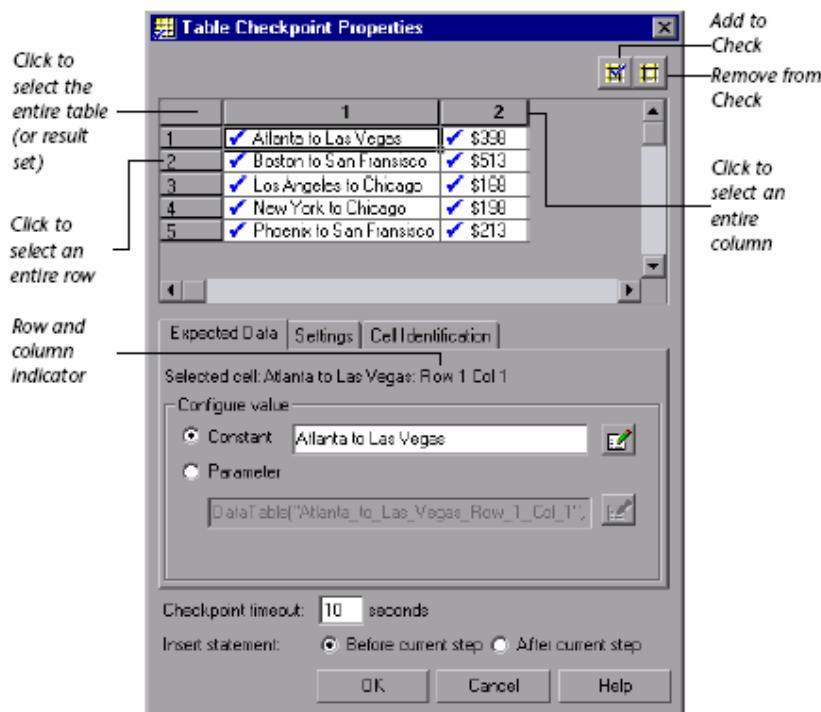
3 When you are done, in the Finish screen of the Query Wizard, select **Exit and return to QuickTest Professional** and click **Finish** to exit Microsoft Query. Alternatively, click **View data or edit query in Microsoft Query** and click **Finish**. After viewing or editing the data, choose

File > Exit and return to QuickTest Professional to close Microsoft Query and return to QuickTest.

4 Return to step 4 on above page to continue creating your database checkpoint in QuickTest.

1.5 Understanding the Table/Database Checkpoint Properties Dialog Box

The Table/Database Checkpoint Properties dialog box enables you to specify which cell contents of your table or 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 top part of the Table/Database Checkpoint Properties dialog box displays the data that was captured for the checkpoint. You use this area to specify which cells you want to check.

Below the expected data, the Database Checkpoint dialog box contains the following three tabs:

- **Expected Data**—Enables you to set each checked cell as a constant or parameterized value. For example, you can instruct QuickTest to use a value from the Data Table as the expected value for a particular cell.
- **Settings**—Enables you to set the criteria for a successful match between the expected and actual 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.

- **Cell Identification**—Enables you to instruct QuickTest how to locate 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 Table/Database Checkpoint Properties dialog box. However, you know that each time you run your test or component, 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 bottom part of the Table/Database 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 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 data to load in a table. Increasing the checkpoint timeout value in this case can ensure that the data has sufficient time to load and therefore enables the checkpoint to pass before the end of the timeout period is reached.

You can see information about the checkpoint timeout, including the time interval used by QuickTest to perform the checkpoint, in the Test Results window.

Note: The **Checkpoint timeout** option is available only when creating a table checkpoint. It is not available when creating a database checkpoint.

- **Insert statement**—Specifies when to perform the checkpoint in the test or component. Choose **Before current step** if you want to check the table or database content before the highlighted step is performed. Choose **After current step** if you want to check the table or database content 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 checkpoint. It is available when adding a new checkpoint to an existing test or component.

1.6 Modifying a Database Checkpoint

You can make the following changes to an existing database checkpoint:

- Modify the SQL query definition.
- Modify the expected data, verification type, or method.

1.6.1 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**.

1.6.2 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.

2 Select Checkpoint Properties. The Checkpoint Properties dialog box opens.

Modify the settings as described “Understanding the Table/Database Checkpoint Properties Dialog Box” above.

17.0 Recovery Manager and Scenarios

You can instruct QuickTest to recover from unexpected events and errors that occur in your testing environment during a run session.

This section describes:

- About Defining and Using Recovery Scenarios
- Deciding When to Use Recovery Scenarios
- Defining Recovery Scenarios
- Understanding the Recovery Scenario Wizard
- Managing Recovery Scenarios
- Setting the Recovery Scenarios List for Your Tests or Components
- Programmatically Controlling the Recovery Mechanism

2.1 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 running tests or components unattended—the test or component is suspended until you perform the operation needed to recover.

The Recovery Scenario Manager provides a wizard that guides you through the process of defining a *recovery scenario*—a definition of an unexpected event and the operation(s) 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 or component.

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 Operation(s)—The operation(s) that need to be performed in order to continue running the test or component. 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 once the recovery operations have been performed, and from which point in the test or component QuickTest should continue, if at all. For example, you may want to restart a test or component from the beginning, or skip a step entirely and continue with the next step in the test or component.

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 it with that test or component. A test or component can have any number of recovery scenarios

associated with it. You can prioritize the scenarios associated with your test or component to ensure that trigger events are recognized and handled in the required order.

When you run a test or component for which you have defined recovery scenarios and an error occurs, QuickTest looks for the defined trigger event(s) 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 or component.

Note: If you choose On error in the Activate recovery scenarios box in the Recovery tab of the Test Settings dialog box or Business Component Settings dialog box, the recovery mechanism does not handle triggers that occur in the last step of a test or component. 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 or component, you can do this by adding an extra step to the end of your test or component.

2.2 Deciding When to Use Recovery Scenarios

If you can predict that a certain event may happen at a specific point in your test or component, it is highly recommended to handle that event directly within your test or component 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 operations are activated only occur after a step returns an error, which can potentially occur several steps after the one that actually caused the error. The alternative, checking for trigger events after every step, may slow performance.

You should use recovery scenarios only for unpredictable events, or events that you cannot synchronize with a specific step in your test or component. For example, a recovery scenario can handle a printer error by clicking the default button in the Printer Error message box. You cannot handle this error directly in your test or component, since you cannot know at what point the network will return the printer error. You could try to handle this event in your test or component by adding an If statement immediately after the step that sent a file to the printer, but if the network takes time to return the printer error, your test or component may have progressed several steps before the error is displayed. Therefore, for this type of event, only a recovery scenario can handle it.

2.3 Defining Recovery Scenarios

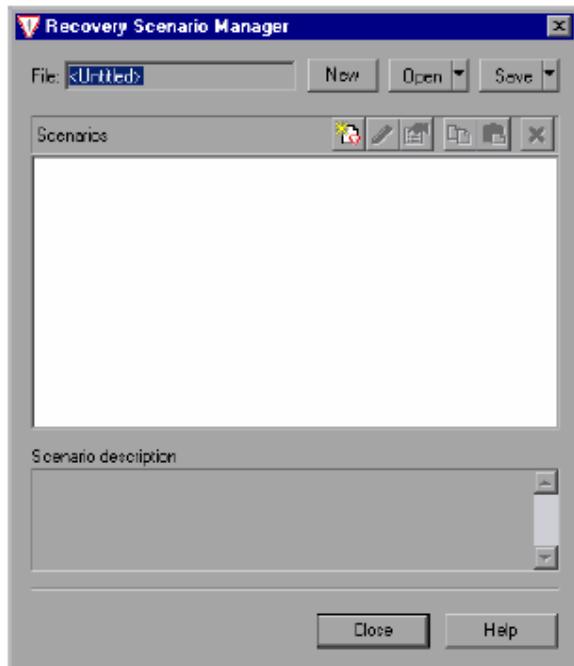
The Recovery Scenario Manager dialog box enables you to create recovery scenarios and save them in recovery files. You create recovery scenarios using the Recovery Scenario Wizard, which leads you through the process of defining each of the stages of the recovery scenario. You then save the recovery scenarios in a recovery file, and associate them with specific tests or components.

2.3.1 Creating a Recovery File

You save your recovery scenarios in a recovery file. A recovery file is a convenient way to organize and store multiple recovery scenarios together. You can create a new recovery file or edit an existing one.

To create a recovery file:

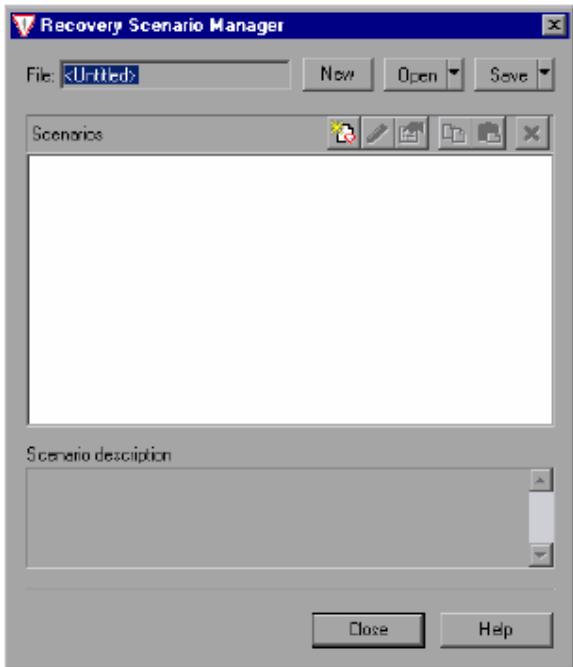
- 1 Choose Tools > Recovery Scenario Manager. The Recovery Scenario Manager dialog box opens.



- 2 By default, the Recovery Scenario Manager dialog box opens with a new recovery file. You can either use this new file, or click the Open button to choose an existing recovery file. Alternatively, you can click the arrow next to the Open button to select a recently-used recovery file from the list. You can now create recovery scenarios using the Recovery Scenario Wizard and save them in your recovery file, as described in the following sections.

2.3.2 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 recovery scenarios. The Recovery Scenario Manager dialog box displays the name of the currently open recovery file, a list of the scenario(s) saved in the recovery file, and a description of each scenario.



2.4 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 five main steps:

- defining the trigger event that interrupts the run session
- specifying the recovery operation(s) 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. **Welcome to the Recovery Scenario Wizard Screen** The Welcome to the Recovery Scenario Wizard screen provides general information about 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.

2.4.1 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 in order to continue the run session. Select this option and click **Next** to continue to the Specify Pop-up Window Conditions screen.
- **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. **Part III • Creating Tests 418 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.

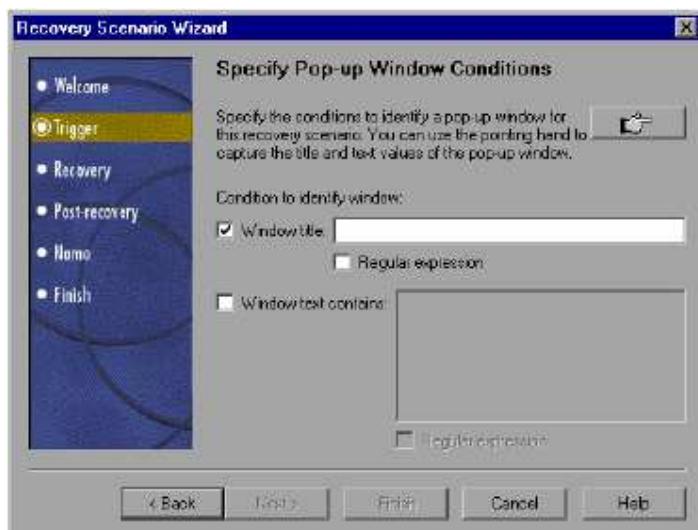
- **Test run error**—QuickTest detects a run error and identifies it by a failed return value from a method. For example, QuickTest may not be able to identify a menu item specified in the method argument, due to the fact that the menu item is not available at a specific point during the run session. QuickTest can detect this run error and activate a defined recovery scenario in order to continue the run session. Select this option and click **Next** to continue to the Select Test Run Error screen.
- **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.

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 replay 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 or component. If you need to recover from an unexpected event or error that may occur in the last step of a test or component, you can do this by adding an extra step to the end of your test or component.

2.4.1.1 Specify Pop-up Window Conditions Screen

If you chose a **Pop-up window** trigger in the Select Trigger Event screen, the Specify Pop-up Window Conditions screen opens.



Click the pointing hand and then click the pop-up window to capture the window title and textual content of the window.

Tip: Hold the CTRL key to change the window focus or perform operations such as a right-click or mouseover to display a context menu. Note that pressing the CTRL key does not enable you to select an application from the Windows taskbar, therefore you must make sure that the window you want to access is not minimized. You can choose whether you want to identify the pop-up window according to its **Window title** and/or **Window text**. You can also 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. Click **Next** to continue to the Recovery Operations screen.

2.4.1.2 Select Object Screen

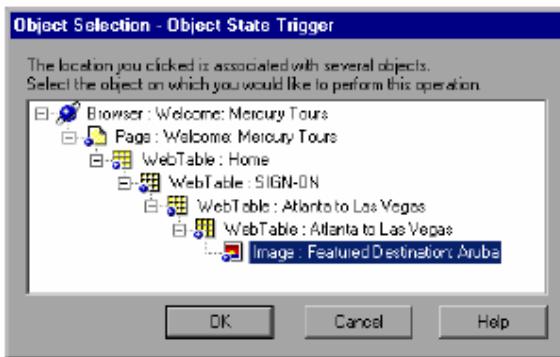
If you chose an **Object state** trigger in the Select Trigger Event screen, the Select Object screen opens.



Click the pointing hand and then click the object whose properties you want to specify.

Tip: Hold the CTRL key to change the window focus or perform operations such as a right-click or mouseover to display a context menu. Note that pressing the CTRL key does not enable you to select an application from the Windows taskbar, therefore you must make sure that the window you want to access is not minimized.

If the location you click is associated with more than one object, the Object Selection–Object State Trigger dialog box opens.

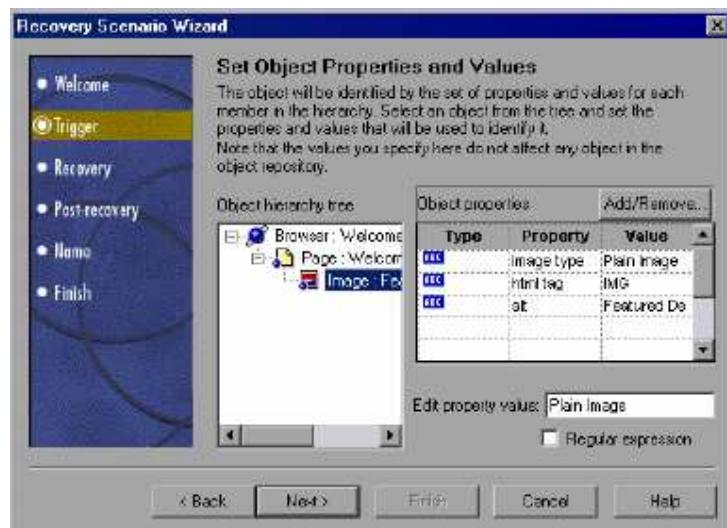


Select the object whose properties you want to specify and click **OK**. The selected object and its parents are displayed in the Select Object screen.

Note: The hierarchical object selection tree also enables you to select an object that QuickTest would not ordinarily record (a non-parent object), such as a web table. Click **Next** to continue to the Set Object Properties and Values screen.

2.4.1.3 Set Object Properties and Values Screen

After you select the object whose properties you want to specify in the Select Object screen, the Set Object Properties and Values screen opens.

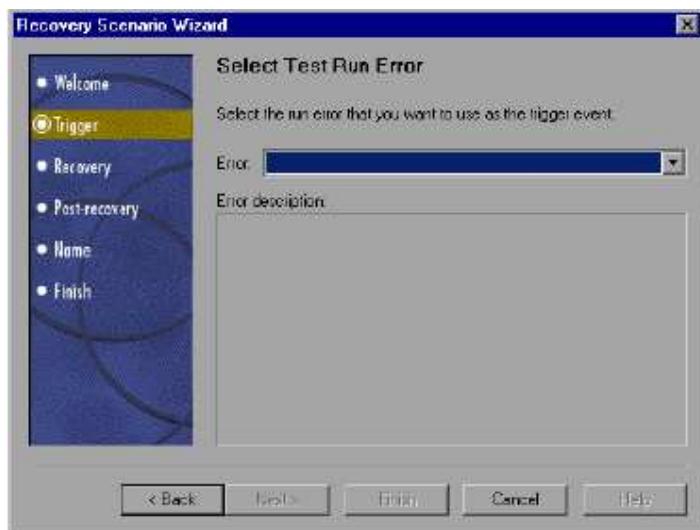


For each object in the hierarchy, in the **Edit property value** box, you can modify the property values used to identify the object. You can also click the **Add/Remove** button to add or remove object properties from the list of property values to check. Note that an object is identified only by its property values, and not by its class.

Select the **Regular expression** check box if you want to use regular expressions in the property value. Click **Next** to continue to the Recovery Operations screen.

2.4.1.4 Select Test Run Error Screen

If you chose a **Test run error** trigger in the Select Trigger Event screen, the Select Test Run Error screen opens.

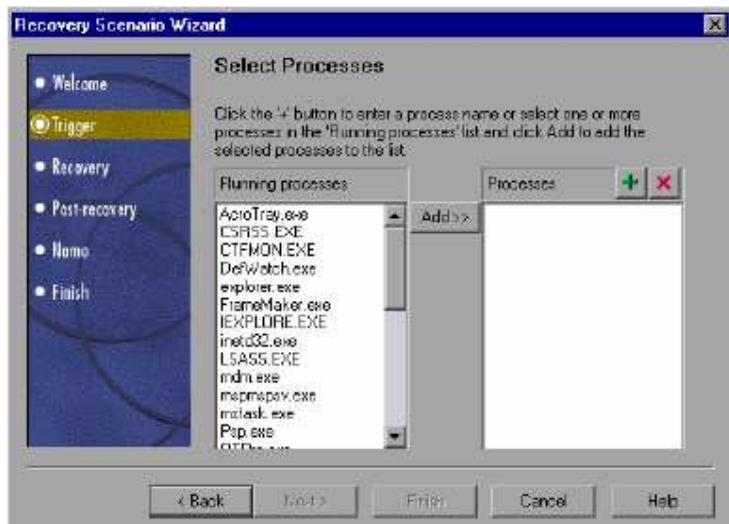


In the **Error** list, choose the run error that you want to use as the trigger event:

- **Any error**—Any error code that is returned by a step.
- **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.

2.4.1.5 Select Processes Screen

If you chose an **Application crash** trigger in the Select Trigger Event screen, 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).

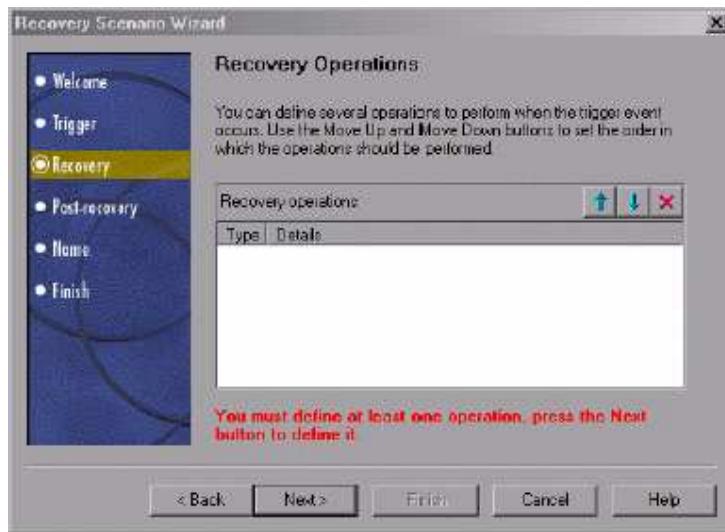
To add a process directly to the **Processes** list, click the **Add New Process** button to enter the name of any process you want to add to the list. To remove a process from the **Processes** list, select it and click the **Remove Process** button.

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.

2.4.2 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.

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.

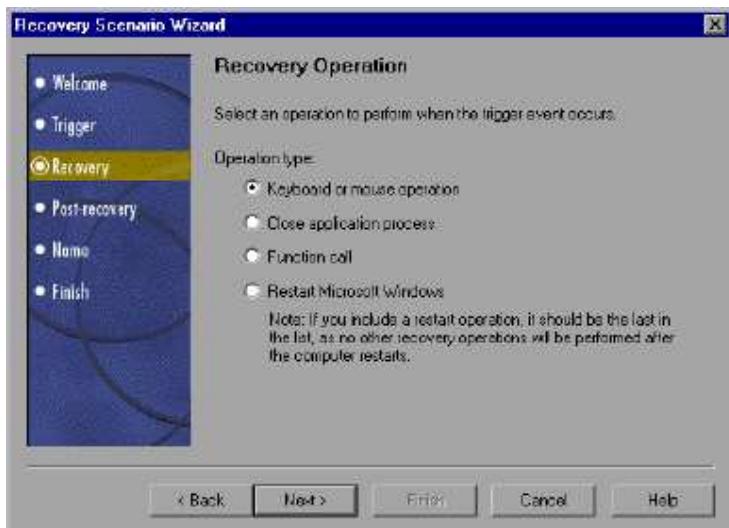
Note: If you define a **Restart Microsoft Windows** recovery operation, it is always inserted as the last recovery operation, and you cannot change its position in the list.

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.

2.4.2.1 Recovery Operation Screen

The Recovery Operation screen enables you to specify the operation(s) 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.

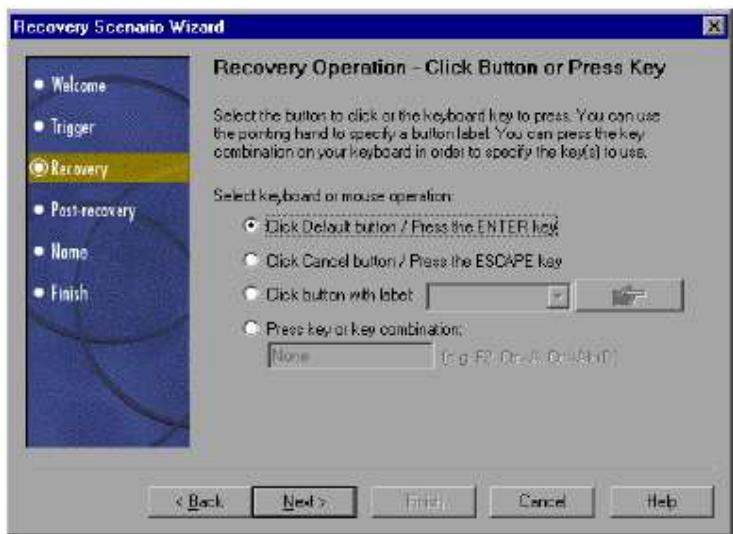
You can define the following types of recovery operations:

- **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.
- **Close application process**—QuickTest closes specified processes. Select this option and click **Next** to continue to the Recovery Operation – Close Processes screen.
- **Function call**—QuickTest calls a VBScript function. Select this option and click **Next** to continue to the Recovery Operation – Function screen.
- **Restart Microsoft Windows**—QuickTest restarts Microsoft Windows. Select this option and click **Next** to continue to the Recovery Operations screen.

Note: If you use the Restart Microsoft Windows recovery operation, you must ensure that any test or component associated with this recovery scenario is saved before you run it. You must also configure the computer on which the test or component is run to auto login on restart.

2.4.2.2 Recovery Operation – Click Button or Press Key Screen

If you chose a **Keyboard or mouse operation** recovery operation in the Recovery Operation screen, 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.

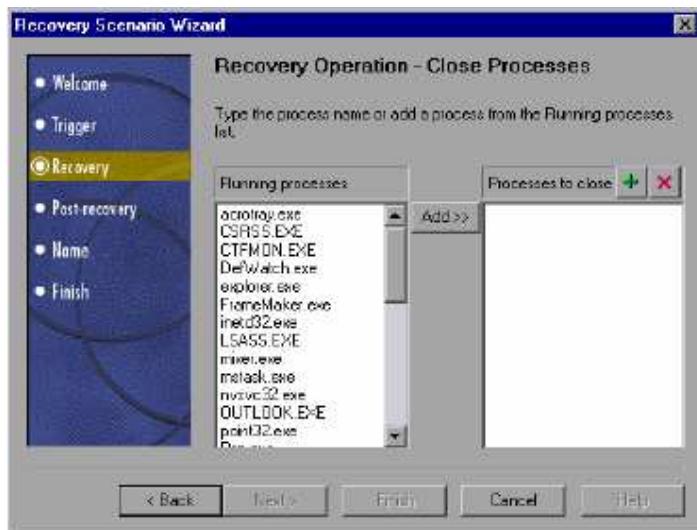
Tip: Hold the CTRL key to change the window focus or perform operations such as a right-click or mouseover to display a context menu. Note that pressing the CTRL key does not enable you to select an application from the Windows taskbar, therefore you must make sure that the window you want to access is not minimized.

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.

2.4.2.3 Recovery Operation – Close Processes Screen

If you chose a **Close application process** recovery operation in the Recovery Operation screen, 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. You can add application processes to the **Processes to close** list by typing them in the **Processes to close** 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 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.

2.4.2.4 Recovery Operation – Function Call Screen

If you chose a **Function call** recovery operation in the Recovery Operation screen, the Recovery Operation – Function Call screen opens. Select a recently specified library file in the **Library file** box. Alternatively, click the browse button to navigate to an existing library file.

Note: QuickTest automatically associates the library file you select with your test or component. Therefore, you do not need to associate the library file with your test or component in the Resources tab of the Test Settings dialog box or Business Component Settings dialog box, respectively.

After you select a library file, choose one of the following options:

- **Select function**—Choose an existing function from the library file you selected.

Note: Only functions that match the prototype syntax for the trigger type selected in the Select Trigger Event screen 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 library file you selected.

Note: If more than one scenario use a function with the same name from different library files, 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 reopens, showing the function operation that you defined.

2.4.3 Post-Recovery Test Run Options Screen

When you clear the **Add another recovery operation** check box in the Recovery Operations screen 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.

➤ **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 iteration**

Stops performing steps in the current action iteration and begins the next action iteration from the beginning (or the next action if no additional iterations of the current action are required).

➤ **Proceed to next test iteration**

Stops performing steps in the current action and begins the next test iteration from the beginning (or stops running the test if no additional iterations of the current action are required).

➤ **Restart current test run**

Stops performing steps and re-runs the test or component from the beginning.

➤ **Stop the test run**

Stops running the test or component.

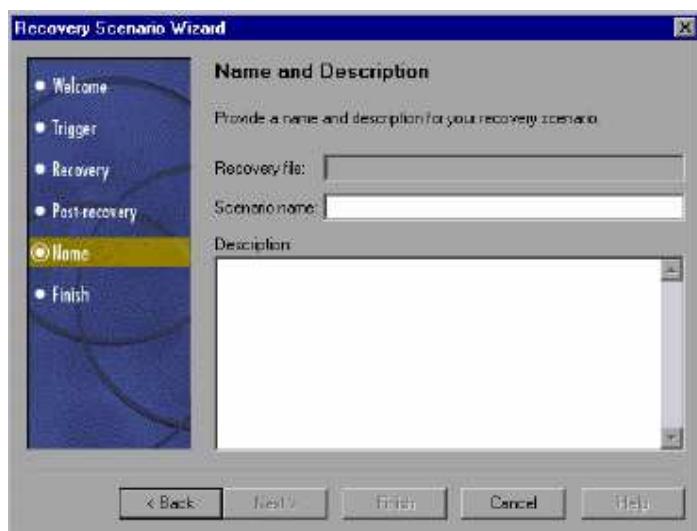
Note: If you chose **Restart Microsoft Windows** as a recovery operation, you can choose from only the last two test run options listed above.

Select a test run option and click **Next** to continue to the Name and Description screen.

2.4.4 Name and Description Screen

After you specify a test run option in the Post-Recovery Test Run Options screen, 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.

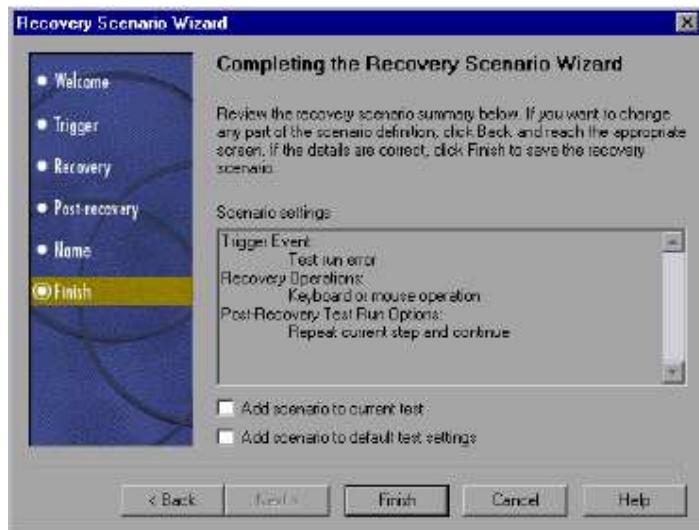


Enter a name and a textual description for your recovery scenario, and click **Next** to continue to the Completing the Recovery Scenario Wizard screen.

2.4.5 Completing the Recovery Scenario Wizard Screen

After you specify a recovery scenario name and description in the Name and Description screen 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 or component, and/or to add it to the default settings for all new tests.



Select the **Add scenario to current test** check box to associate this recovery scenario with the current test or component. When you click **Finish**, QuickTest adds the recovery scenario to the **Scenarios** list in the Recovery tab of the Test Settings dialog box or Business Component Settings dialog box, respectively.

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 tab of the Test Settings dialog box.

Note: You can remove scenarios from the default scenarios list.

Note: You define the default recovery scenarios for all new components in the component template.

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.

To save a new or modified recovery file:

1 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 Attachment 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.

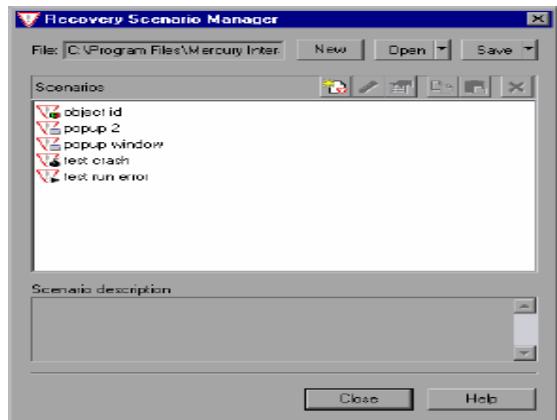
2 Choose the folder in which you want to save the file.

3 Type a name for the file in the **File name** box. The recovery file is saved in the specified location with the file extension **.qrs**.

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 above. 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.

Managing Recovery Scenarios

Once you have created recovery scenarios, you can use the Recovery Scenario Manager to manage them.



The Recovery Scenario Manager contains the following recovery scenario icons:

Icon Description

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 or component 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:

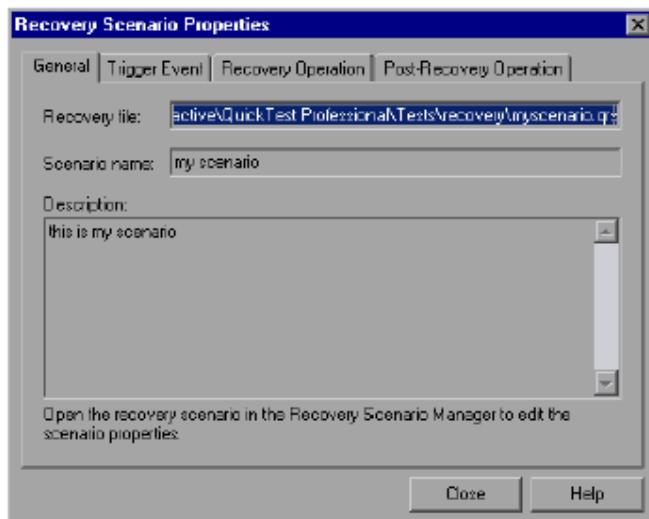
- Viewing Recovery Scenario Properties
- Modifying Recovery Scenarios
- Deleting Recovery Scenarios
- Copying Recovery Scenarios between Recovery Scenario Files

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 operation(s) 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.

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 or component, 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.

Setting the Recovery Scenarios List for Your Tests or Components

After you have created recovery scenarios, you associate them with selected tests or components so that QuickTest will perform the appropriate scenario(s) 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 or component. You can also define which recovery scenarios will be used as the default scenarios for all new tests.

Note: You define the default recovery scenarios for all new components in the component template.

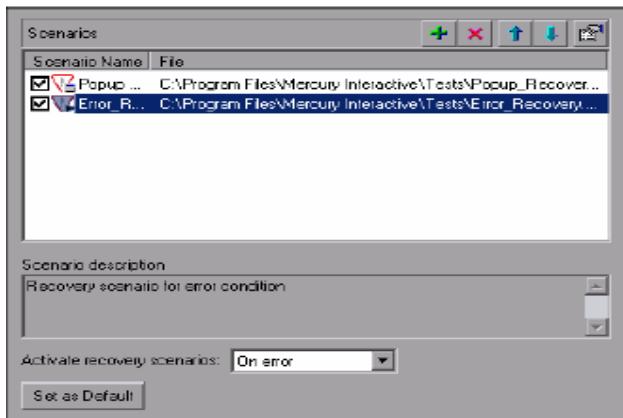
Adding Recovery Scenarios to Your Test or Component

After you have created recovery scenarios, you can associate one or more scenarios with a test or component in order to instruct QuickTest to perform the recovery scenario(s) during the run session if a trigger event occurs. The Recovery tab of the Test Settings dialog box or Business Component Settings dialog box lists all the recovery scenarios associated with the current test or component.

Tip: When a trigger event occurs, QuickTest checks for applicable recovery scenarios in the order in which they are displayed in the Recovery tab.

To add a recovery scenario to a test or component:

- 1 Choose **Test > Settings** or **Component > Settings**. The Test Settings dialog box or Business Component Settings dialog box opens. Select the Recovery tab.



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 scenario(s) you want to associate with the test or component. 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 scenario(s) that you want to associate with the test or component and click **Add Scenario**. The Add Recovery Scenario dialog box closes and the selected scenarios are added to the **Scenarios** list in the Recovery tab.

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 or component.

Viewing Recovery Scenario Properties

You can view properties for any recovery scenario associated with your test or component.

Note: You modify recovery scenario settings from the Recovery Scenario Manager dialog box.

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.

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 tab of the Test Settings dialog box or Business Component 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.
- 3 Repeat steps 1-2 for each scenario whose priority you want to change.

Removing Recovery Scenarios from Your Test or Component

You can remove the association between a specific scenario and a test or component using the Recovery tab of the Test Settings dialog box or Business Component Settings dialog box. After you remove a scenario from a test or component, 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 or component:

- 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 or component.

Enabling and Disabling Recovery Scenarios

You can enable or disable specific scenarios and determine when QuickTest activates the recovery scenario mechanism in the Recovery tab of the Test Settings dialog box or Business Component Settings dialog box. When you disable a specific scenario, it remains associated with the test or component, but is not performed by QuickTest during the run session. You can enable the scenario at a later time.

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.
 - **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.

Note: Choosing **On every step** may result in slower performance during the run session.

Tip: You can also enable or disable specific scenarios or all scenarios associated with a test or component programmatically during the run session.

Setting Default Recovery Scenario Settings for All New Tests

You can click the **Set as Default** button in the Recovery tab 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. You define the default recovery scenarios for all new components in the component template.

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.

18.0 Scripting in Real Time Environments

18.1 QuickTest Pro Coding Standards & Best Practices

18.1.1 Introduction:

These standards are for testers who are using QuickTest Pro to develop their automated test scripts. The purpose of this document is to provide testers with general guidelines and rules for building repeatable, accurate, maintainable, and portable test scripts. This standard document covers areas such as Variable Naming, Commenting Code, Formatting of Code etc.

18.2 Naming Conventions

18.2.1 Local scope variables

Variables represent values that can be changed within a procedure or function. Local scope variables are placeholders that reside within a function- or a script-body.

Prefixes for Variable Data Types

Data Type	Prefix	Example
Boolean	bln	BlnLoggedIn
Date	d	dTomorrow
Variant	v	vTblValue
Integer	int	IntAge
Object	o	oUserTable
String	str	strName
Arrays	a	Used in another prefix, as in astrEmployee

18.2.2 Global scope variables

The values of global variables can be used and changed all over the project within all scripts and libraries.

Syntax

"g" + [Prefix]+[ShortDescription]

Letter “g” indicates that the scope of the variable is global. *[Prefix]* is a lowercase letter that represents the type of the global variable. The rules for *[Prefix]* are the same as for “Local scope variables”.

Examples

gintNumOfPersons
gstrPersonLastname

18.2.3 Constants

Constants are “variables” that cannot be changed within a function- or script-body. The value will always be the same during script-execution. Constant names should be descriptive and not ambiguous. Constant names should be in upper case, with proper words separated by the underscore character.

Examples

ID_OS_VERSION
MY_TEMP_URL

18.2.4 Functions/Actions

The Function/Action name should be descriptive of its primary purpose. The Function/Action name must always correspond to what the Function/Action is actually doing. If name of the function is “OpenLogFile("log file name")” then it is expected that the function only opens log file.

Examples

CountItemInString()

18.2.5 Reusable Actions

Quick Test Pro reusable actions names should start with an “r” to reflect that the action is reusable.

18.2.6 Scripts

must capitalize the first letter of each word in the test script name . The exception being the product name, which should always be capitalized.

Examples

IB_Auditlog_Save

18.2.7 Function Libraries

Functional library names should reflect the product name, underscore, functionality that is being tested, underscore and the words “lib” at the end.

Example:

IB_Log_Lib

IB = Intranet Browser product

Log = log functionality in the application

Lib = Functional Library

18.2.8 Object Repository Files

Object GUI files names should reflect the product name being tested.

Example:

IB_GUI.tsr

IB = Intranet Browser product

18.3 Coding Rules

18.3.1 Commenting Code

Headers - Every test script, functional library, or actions should contain a header comment that displays the name, creator, date, description, assumptions, and post conditions.

Test Script Header Comment.

```
*****
' SCRIPT NAME    : CL11_FeatureAuditAnnotateSDOC
' CREATED BY     : Applabs
' DATE CREATED   : 16/Mar/2001
' DESCRIPTION    : This Script Checks That Feature Audit for Annotate is not
'                  transferred to the auditlog  file when a sealed Word document
'                  is annotated by inserting comment
' Modification Log :
' ID  Date    Name      Purpose
' ---- ----- -----
*****
```

Function Header Comment.

```
*****
' FUNCTION NAME   : CheckClientFeatureAudited
' CREATED BY     : Applabs
' DATE CREATED   : 09/March/2001
```

```
' DESCRIPTION      : This Function checks that required client feature audit is transferred to
auditlog file.

' Parameters       : strFilePath - Path of decrypted auditlog file.
'                   : strFeature - Expected feature to be audited.
'                   : strPublisherID - Publisher ID of sealed content.

' Return Values    : Returns the number of times the client feature is audited in the auditlog file.

' Preconditions    : Audit log files exist
' Postcondition:   : Feature audit transfer to audit log file
' Modification Log :  
' ID   Date   Name      Purpose  
' ----  -----  -----  
*****  
*****
```

Commenting single lines / paragraphs – The need to comment an individual line of code is rare. Two possible reasons for commenting a single line of code are:

The line is complicated enough to need an explanation.

The single line once had an error and you want to record the change. When modifying a single line of code, the original line should be commented out using the author's initials and date in brackets, separated by a single dash.

For example (in VBScript), the following shows a code change by SMB (initials):

```
[SMB-03/09/01]Browser("e.POWER").Page("Worklist").WebRadioGroup("item_select").Select
"122|0|23"
Browser("e.POWER").Page("Worklist").WebRadioGroup("item_select").Select strItemID  '[SMB-
03/09/01]
```

Use end of line comments to annotate data declarations.

Example:

```
Dim intEmpAge      'Stores Employee Age.
Dim strEmpName     'Stores Employee Name.
```

Such data annotations include: units of numeric data, range of allowable values, limitations on input data, and the meaning of enumerated or constant codes.

Focus paragraph comments on the *why* rather than the *how*. Simply put, a reader should be able to discern from quality code how an operation is performed. Why that operation is performed will be less apparent.

When commenting individual lines or paragraphs, indent the comment and align it with its corresponding code. Also, precede the comment line by a single blank line.

Commenting Control Structures

Place a comment before each block of statements, if, case, or loop. Use the comment to clarify the purpose of the control structure.

Example:

```
'Loop through the each row in the data table.
```

```
For intCount=1 to datatable.GetRowCount  
    'Check whether content in both columns are equal.  
    If datatable("Name1", dtGlobalSheet)= datatable("Name2", dtGlobalSheet) then  
        reporter.ReportEvent micPass,"content of both column are equal in row" & intCount  
    End If  
Next
```

For long code blocks, comment the end of each control structure. This makes script code which spans many lines easier to follow.

Example:

```
For intCount=1 to datatable.GetRowCount  
    'Check whether content in both columns are equal.  
    If datatable("Name1", dtGlobalSheet)= datatable("Name2", dtGlobalSheet) then  
        reporter.ReportEvent micPass,"content of both column are equal in row" & intCount  
    End If  
Next    'Loop through the each row in the data table.
```

18.3.2 Formatting Code

The following are the directives that must be complied with while formatting the code

- Do not place multiple statements on a single line.
Placing multiple statements in a single line increase complexity. Do not do it.
- Do not exceed 90 Characters on a line
- Make sure that not more than 90 characters are placed on a single line.
- Use Indentation to show organizational structure
- Do proper indentation through out your coding. Flow control statements should be indented one tab length for easier readability.

Example:

```
For intCount = 0 to 10  
    MyVar = MyVar + 1  
    If MyVar = 20  
        MyVar =0  
    End If
```

Next

18.3.3 Using Shared Object Repository

Use shared Object Repository instead of Object Repository per Action. This will an easy way for handling changes in the object repository.

18.3.4 Using Relative paths

Use relative paths while calling Shared object repository files, VbScript function library files and Reusable actions.

18.3.5 Using Global Variables

Declare and Assign, all the variables that can be used in many scripts (For example: Username, Password, PrinterName etc) in a Reusable Action or VBScript file and use these variables in Other Actions. Using this method we can modify the value of variable in single global file instead of making modification in many script files.

Mercury QuickTest Professional 8.0 Shortcut Key Reference Card

File Menu

Command	Press
New Test	CTRL + N
Open Test	CTRL + SHIFT + N
Business Component > New	CTRL + SHIFT + O
Business Component > Open	CTRL + SHIFT + E
Business Component > Edit Template	CTRL + O
Save	CTRL + S
Export to Zip File	CTRL + ALT + S
Import from Zip File	CTRL + ALT + O
Print	CTRL + P

Edit Menu

Cut	CTRL + X (EV only)
Copy	CTRL + C
Paste	CTRL + V
Delete	DEL
Undo	Ctrl + Z (EV only)
Redo	CTRL + Y (EV only)
Rename Action	F2
Find	CTRL + F (EV only)
Replace	CTRL + H (EV only)
Go To	CTRL + G (EV only)
Bookmarks	CTRL + B (EV only)
Complete Word	CTRL + SPACE (EV only)
Argument Info	CTRL + SHIFT + SPACE (EV only)
Apply "With" To Script	CTRL + W (EV only)
Remove "With" Statements	CTRL + SHIFT + W (EV only)

Insert Menu

Checkpoint > Standard Checkpoint	F12
Output Value > Standard Output Value	CTRL + F12
Step > Step Generator	F7
New Step	F8 (KV only)
New Step After Block	SHIFT + F8 (KV only)

Key: KV = Keyword View
EV = Expert View

Command	Press
Record	F3
Run	F5
Stop	F4
Analog Recording	CTRL + SHIFT + F4
Low Level Recording	CTRL + SHIFT + F3
Object Properties	CTRL + ENTER
Value Configuration Options	CTRL + F11 on a value (KV only)
Pause	PAUSE
Step Into	F11
Step Over	F10
Step Out	SHIFT + F11
Insert/Remove Breakpoint	F9
Clear All Breakpoints	CTRL + SHIFT + F9
Edit > Cut	CTRL + X
Edit > Copy	CTRL + C
Edit > Paste	CTRL + V
Edit > Clear > Contents	CTRL + DEL
Edit > Insert	CTRL + I
Edit > Delete	CTRL + K
Edit > Fill Right	CTRL + R
Edit > Fill Down	CTRL + D
Edit > Find	CTRL + F
Edit > Replace	CTRL + H
Data > Recalc	F9
Insert Multi-line Value	CTRL + F2 while editing cell
Activate next/previous sheet	CTRL + PAGEUP/ CTRL + PAGEDOWN
View Keyword View/Expert View	CTRL + TAB
Open context menu for step or Data Table cell	SHIFT + F10 or Application key (█)
Expand all branches	* [on numeric keypad] (KV only)
Expand branch	+ [on numeric keypad] (KV only)
Collapse branch	- [on numeric keypad] (KV only)



Mercury Interactive Corporation

379 North Whisman Road
Mountain View, CA 94043

Main Telephone: (650) 603-5200

Sales & Information: (800) TEST-911

Customer Support: (877) TEST-HLP

Fax: (650) 603-5300

Home Page: www.mercury.com

Customer Support: support.mercury.com

