

OPTIMIZE

MERCURY QUICKTEST PROFESSIONAL™

VERSION 9.1

User's Guide

Volume 1

MERCURY™

BUSINESS TECHNOLOGY OPTIMIZATION

Mercury QuickTest Professional

User's Guide - Volume 1

Version 9.1

Document Release Date: October 1, 2006

Mercury QuickTest Professional User's Guide - Volume 1, Version 9.1

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: United States: 5,511,185; 5,657,438; 5,701,139; 5,870,559; 5,958,008; 5,974,572; 6,137,782; 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; 6,564,342; 6,587,969; 6,631,408; 6,631,411; 6,633,912; 6,694,288; 6,738,813; 6,738,933; 6,754,701; 6,792,460 and 6,810,494. Australia: 763468 and 762554. Other patents pending. All rights reserved.

Mercury, Mercury Interactive, the Mercury logo, the Mercury Interactive logo, LoadRunner, WinRunner, SiteScope and TestDirector are trademarks of Mercury Interactive Corporation and may be registered in certain jurisdictions. The absence of a trademark from this list does not constitute a waiver of Mercury's intellectual property rights concerning that trademark.

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

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

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

© 1992 - 2006 Mercury Interactive Corporation, All rights reserved

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

Table of Contents

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

Welcome to This Guide	xvii
How This Guide Is Organized	xviii
Who Should Read This Guide	xx
QuickTest Professional Online Documentation	xx
Additional Online Resources.....	xxii
Typographical Conventions.....	xxii

PART I: INTRODUCING QUICKTEST PROFESSIONAL (Vol. 1)

Chapter 1: Introduction	3
Testing with QuickTest.....	5
Understanding the Testing Process.....	6
Programming in the Expert View.....	10
Understanding Functions and Function Libraries.....	10
Managing the Testing Process Using Quality Center	11
Understanding Business Process Testing.....	12
Setting Required Access Permissions.....	13
Using the Sample Site.....	14
Modifying License Information	14
Updating the QuickTest Software	15

Chapter 2: QuickTest at a Glance	17
Starting QuickTest	18
QuickTest Window	20
Keyword View	23
Expert View	25
Function Library	26
Active Screen	27
Information Pane	28
Missing Resources Pane	29
Data Table.....	30
Debug Viewer Pane.....	30
Customizing the QuickTest Window Layout.....	31
Working With Multiple Documents	39
Using QuickTest Commands.....	41
Browsing the QuickTest Professional Program Folder	53
Viewing Product Information	57
Chapter 3: Understanding the Test Object Model.....	61
About Understanding the Test Object Model	61
Applying the Test Object Model Concept	65
Viewing Object Properties Using the Object Spy	70
Viewing Object Methods and Method Syntax	
Using the Object Spy	73

PART II: CREATING TESTS (Vol. 1)

Chapter 4: Designing Tests	79
About Designing Tests.....	80
Planning and Preparing to Create a Test	81
Creating a Test Using the Keyword-Driven Methodology	82
Recording a Test	88
Understanding Your Recorded Test	92
Choosing the Recording Mode	93
Enhancing Your Test	101
Managing Your Test	103

Chapter 5: Working with the Keyword View	111
About Working with the Keyword View	112
Understanding the Keyword View	113
Understanding the QuickTest Recorded Object Hierarchy	118
Adding a Standard Step to Your Test	120
Adding Other Types of Steps to Your Test	133
Modifying the Parts of a Step	136
Working with Comments	136
Managing Action Steps	137
Using Keyboard Commands in the Keyword View	140
Defining Keyword View Display Options	141
Viewing Properties of Step Elements in the Keyword View	147
Working with Breakpoints in the Keyword View	148
Chapter 6: Working with Test Objects	149
About Working with Test Objects	150
Understanding Object Repository Types	151
Understanding the Object Repository Window	156
Viewing and Modifying Test Object Properties	163
Mapping Repository Parameter Values	185
Adding Objects to the Object Repository	189
Defining New Test Objects	200
Copying, Pasting, and Moving Objects	
in the Object Repository	202
Deleting Objects from the Object Repository	205
Locating Objects	206
Working with Test Objects During a Run Session	213
Managing Shared Object Repository Associations	214
Exporting Local Objects to an Object Repository	217
Chapter 7: Working with the Active Screen	219
About Working with the Active Screen	219
Increasing or Decreasing the Active Screen Information	
Saved with a Test	221
Changing the Active Screen	223
Tips for Improving Active Screen Performance	224
Chapter 8: Understanding Checkpoints	225
About Understanding Checkpoints	225
Adding Checkpoints to a Test	227
Understanding Types of Checkpoints	228

Chapter 9: Checking Tables	233
About Checking Tables	233
Creating a Table Checkpoint	234
Understanding the Table Checkpoint Properties Dialog Box.....	238
Checking Table Content	239
Checking Table Properties.....	249
Modifying a Table Checkpoint	252
Chapter 10: Checking Text	255
About Checking Text	256
Creating a Text Checkpoint	257
Understanding the Text Checkpoint Properties Dialog Box	259
Modifying a Text Checkpoint	269
Creating a Standard Checkpoint for Checking Text.....	269
Chapter 11: Checking Object Property Values	273
About Checking Object Property Values	273
Creating Standard Checkpoints	274
Understanding the Checkpoint Properties Dialog Box	277
Understanding the Image Checkpoint Properties Dialog Box.....	281
Modifying Checkpoints.....	283
Chapter 12: Checking Bitmaps	285
About Checking Bitmaps	285
Checking a Bitmap	286
Modifying a Bitmap Checkpoint	293
Chapter 13: Checking Databases	297
About Checking Databases.....	297
Creating a Check on a Database	298
Understanding the Database Checkpoint Properties Dialog Box.....	303
Modifying a Database Checkpoint.....	312
Chapter 14: Checking XML	313
About Checking XML.....	314
Creating XML Checkpoints.....	316
Updating the XML Hierarchy for XML Test Object Operation	
Checkpoints (for WebService Test Objects Only)	334
Modifying XML Checkpoints.....	342
Reviewing XML Checkpoint Results	342
Using XML Objects and Methods to Enhance Your Test	343

Chapter 15: Configuring Values.....	345
About Configuring Values.....	345
Configuring Constant and Parameter Values	346
Understanding and Using Regular Expressions	352
Defining Regular Expressions.....	355
Chapter 16: Parameterizing Values	365
About Parameterizing Values	365
Parameterizing Values in Steps and Checkpoints.....	367
Using Test and Action Input Parameters	374
Using Data Table Parameters.....	378
Using Environment Variable Parameters	385
Using Random Number Parameters	396
Example of a Parameterized Test.....	398
Using the Data Driver to Parameterize Your Test	403
Chapter 17: Outputting Values.....	411
About Outputting Values	411
Creating Output Values.....	412
Outputting Property Values	419
Specifying the Output Type and Settings	425
Outputting Text Values.....	430
Outputting Table Values	436
Outputting Database Values.....	450
Outputting XML Values	454
Updating the XML Hierarchy for XML Test Object Operation	
Output Value Steps (For WebService Test Objects Only)	466
Chapter 18: Working with Actions	471
About Working with Actions	472
Using Global and Action Data Sheets	475
Using the Action Toolbar in the Keyword View	477
Creating New Actions.....	478
Guidelines for Working with Actions	480
Setting Action Properties.....	482
Nesting Actions	493
Splitting Actions.....	495
Renaming Actions	497
Removing Actions from a Test	499
Creating an Action Template	503

Chapter 19: Handling Missing Resources	505
About Handling Missing Resources.....	506
Handling Calls to Missing Actions.....	508
Handling Missing Shared Object Repositories	514
Handling Unmapped Shared Object Repository Parameter Values.....	515
Chapter 20: Working with Data Tables	517
About Working with Data Tables.....	518
Working with Global and Action Sheets	519
Saving the Data Table.....	521
Editing the Data Table.....	522
Using Data Table Files with Quality Center.....	530
Importing Data from a Database.....	531
Using Formulas in the Data Table.....	534
Using Data Table Scripting Methods.....	538
Chapter 21: Adding Steps Containing Programming Logic	539
About Adding Steps Containing Programming Logic	540
Inserting Steps Using the Step Generator	541
Using Conditional Statements	560
Using Loop Statements.....	566
Generating With Statements for Your Test.....	569
Generating Messages	575
Adding Comments	578
Synchronizing Your Test	579

PART III: RUNNING AND DEBUGGING TESTS (Vol.1)

Chapter 22: Debugging Tests and Function Libraries	587
About Debugging Tests and Function Libraries	588
Slowing a Debug Session	590
Using the Single Step Commands.....	590
Using the Run to Step and Start from Step Commands	593
Pausing a Run Session	595
Using Breakpoints	596
Using the Debug Viewer.....	600
Handling Run Errors.....	602
Practicing Debugging an Action or a Function.....	603

Chapter 23: Running Tests	607
About Running Tests	608
Running Your Entire Test	609
Running Part of Your Test	613
Updating a Test	616
Using Optional Steps	625
Running a Test Batch	627
Chapter 24: Analyzing Test Results	631
About Analyzing Test Results	632
Understanding the Test Results Window	634
Viewing the Results of a Run Session	638
Viewing Checkpoint Results	653
Viewing Parameterized Values and Output Value Results	675
Analyzing Smart Identification Information in the Test Results	685
Deleting Test Run Results	689
Submitting Defects Detected During a Run Session	697
Viewing WinRunner Test Steps in the Test Results	699
Customizing the Test Results Display	702

PART IV: CONFIGURING BASIC SETTINGS (Vol. 1)

Chapter 25: Setting Global Testing Options	707
About Setting Global Testing Options	707
Using the Options Dialog Box	708
Setting General Testing Options	710
Setting Folder Testing Options	712
Setting Active Screen Options	715
Setting Run Testing Options	723
Setting Windows Application Testing Options	726
Setting Web Testing Options	739
Chapter 26: Setting Options for Individual Tests	755
About Setting Options for Individual Tests	756
Using the Test Settings Dialog Box	757
Defining Properties for Your Test	759
Defining Run Settings for Your Test	763
Defining Resource Settings for Your Test	767
Defining Parameters for Your Test	771
Defining Environment Settings for Your Test	774
Defining Web Settings for Your Test	782
Defining Recovery Scenario Settings for Your Test	784

Chapter 27: Setting Record and Run Options	789
About Setting Record and Run Options.....	790
Using the Record and Run Settings Dialog Box	790
Setting Web Record and Run Options	793
Setting Windows Applications Record and Run Options.....	796
Using Environment Variables to Specify the Record and Run Details for Your Test.....	802

PART V: WORKING WITH SUPPORTED ENVIRONMENTS (Vol. 2)

Chapter 28: Working with QuickTest Add-Ins	809
About Working with QuickTest Add-Ins	810
Loading QuickTest Add-ins	811
Tips for Working with QuickTest Add-ins	814
Chapter 29: Testing Web Objects	817
About Testing Web Objects.....	818
Working with Web Browsers.....	820
Checking Web Objects	825
Checking Web Pages	829
Checking Web Content Accessibility.....	841
Accessing Password-Protected Resources in the Active Screen	845
Activating Methods Associated with a Web Object.....	850
Using Scripting Methods with Web Objects	851
Registering Browser Controls	852

PART VI: WORKING WITH ADVANCED TESTING FEATURES (Vol. 2)

Chapter 30: Working with Advanced Action Features	855
About Working with Advanced Action Features	856
Inserting Calls to Existing Actions	856
Setting Action Parameters	864
Using Action Parameters	868
Setting Action Call Properties	873
Sharing Action Information	878
Understanding Action Syntax in the Expert View.....	880
Exiting an Action	883
Chapter 31: Learning Virtual Objects	885
About Learning Virtual Objects	885
Understanding Virtual Objects	887
Understanding the Virtual Object Manager	888
Defining a Virtual Object	889
Removing or Disabling Virtual Object Definitions.....	894

Chapter 32: Defining and Using Recovery Scenarios	897
About Defining and Using Recovery Scenarios.....	898
Deciding When to Use Recovery Scenarios.....	900
Defining Recovery Scenarios	901
Understanding the Recovery Scenario Wizard	905
Managing Recovery Scenarios	931
Setting the Recovery Scenarios List for Your Tests.....	935
Programmatically Controlling the Recovery Mechanism	941
Chapter 33: Configuring Object Identification.....	943
About Configuring Object Identification	944
Understanding the Object Identification Dialog Box.....	945
Configuring Smart Identification.....	959
Mapping User-Defined Test Object Classes.....	969
Chapter 34: Working in the Expert View and Function Library Windows.....	973
About Working in the Expert View and Function Library Windows	974
Understanding and Using the Expert View	975
Navigating in the Expert View and Function Libraries	986
Understanding Basic VBScript Syntax.....	996
Using Programmatic Descriptions.....	1005
Running and Closing Applications Programmatically	1017
Using Comments, Control-Flow, and Other VBScript Statements	1019
Retrieving and Setting Test Object Property Values	1027
Accessing Run-Time Object Properties and Methods	1029
Running DOS Commands.....	1031
Enhancing Your Tests and Function Libraries Using the Windows API.....	1031
Choosing Which Steps to Report During the Run Session	1035
Chapter 35: Working with User-Defined Functions and Function Libraries	1037
About Working with User-Defined Functions and Function Libraries	1038
Managing Function Libraries	1040
Working with Associated Function Libraries	1052
Using the Function Definition Generator.....	1056
Registering User-Defined Functions as Test Object Methods	1072
Additional Tips for Working with User-Defined Functions	1077
Executing Externally-Defined Functions from Your Test	1080

Chapter 36: Working with the QuickTest Script Editor	1083
About the QuickTest Script Editor	1084
Understanding the QuickTest Script Editor Window	1085
Customizing the QuickTest Script Editor Window	1086
Understanding the Flow Pane	1088
Understanding the Resources Pane	1090
Understanding the Display Area	1093
Working with Tests	1095
Working with Function Libraries	1099
Chapter 37: Automating QuickTest Operations	1105
About Automating QuickTest Operations	1106
Deciding When to Use QuickTest Automation Scripts	1107
Choosing a Language and Development Environment for Designing and Running Automation Scripts	1108
Learning the Basic Elements of a QuickTest Automation Script	1110
Generating Automation Scripts	1111
Using the QuickTest Automation Reference	1112

PART VII: MANAGING AND MERGING OBJECT REPOSITORIES (Vol. 2)

Chapter 38: Managing Object Repositories	1115
About Managing Object Repositories	1116
Understanding the Object Repository Manager	1118
Working with Object Repositories	1124
Manipulating Objects in Shared Object Repositories	1129
Working with Repository Parameters	1133
Modifying Test Object Details	1140
Locating Objects	1143
Performing Merge Operations	1144
Performing Import and Export Operations	1145
Manipulating Object Repositories Using Automation	1148

Chapter 39: Merging Shared Object Repositories.....	1151
About Merging Shared Object Repositories	1152
Understanding the Object Repository Merge Tool	1153
Using Object Repository Merge Tool Commands.....	1159
Defining Default Settings	1161
Merging Two Object Repositories	1165
Updating a Shared Object Repository from Local Object Repositories.....	1168
Viewing Merge Statistics.....	1174
Understanding Object Conflicts	1175
Resolving Object Conflicts	1178
Filtering the Target Repository Pane	1180
Finding Specific Objects	1181
Saving the Target Object Repository	1183
Chapter 40: Comparing Shared Object Repositories	1187
About Comparing Shared Object Repositories.....	1188
Understanding the Object Repository Comparison Tool	1189
Using Object Repository Comparison Tool Commands.....	1193
Understanding Object Differences	1194
Changing Color Settings	1195
Comparing Object Repositories	1197
Viewing Comparison Statistics.....	1199
Filtering the Repository Panes.....	1200
Synchronizing Object Repository Views	1201
Finding Specific Objects	1202

PART VIII: CONFIGURING ADVANCED SETTINGS (Vol. 2)

Chapter 41: Configuring Web Event Recording.....	1207
About Configuring Web Event Recording	1208
Selecting a Standard Event Recording Configuration.....	1209
Customizing the Web Event Recording Configuration	1211
Recording Right Mouse Button Clicks	1221
Saving and Loading Custom Event Configuration Files.....	1226
Resetting Event Recording Configuration Settings.....	1228
Chapter 42: Customizing the Expert View and Function Library Windows.....	1231
About Customizing the Expert View and Function Library Windows.....	1232
Customizing Editor Behavior	1232
Customizing Element Appearance	1236
Personalizing Editing Commands.....	1238

Chapter 43: Setting Testing Options During the Run Session.....	1241
About Setting Testing Options During the Run Session.....	1241
Setting Testing Options.....	1242
Retrieving Testing Options.....	1244
Controlling the Test Run	1244
Adding and Removing Run-Time Settings.....	1245

PART IX: WORKING WITH OTHER MERCURY PRODUCTS (Vol.2)

Chapter 44: Working with WinRunner	1249
About Working with WinRunner	1249
Calling WinRunner Tests	1250
Calling WinRunner Functions	1254
Chapter 45: Working with Quality Center.....	1259
About Working with Quality Center	1260
Connecting to and Disconnecting from Quality Center.....	1261
Integrating QuickTest with Quality Center	1271
Saving Tests to a Quality Center Project.....	1272
Opening Tests from a Quality Center Project.....	1273
Working with Template Tests	1277
Running a Test Stored in a Quality Center	
Project from QuickTest	1285
Managing Test Versions in QuickTest.....	1287
Setting Preferences for Quality Center Test Runs	1295
Chapter 46: Working with Business Process Testing.....	1303
About Working with Business Process Testing	1303
Understanding Business Process Testing Roles	1304
Understanding Business Process Testing Methodology.....	1308
Chapter 47: Working with Mercury Performance Testing	
 and Business Availability Center Products	1313
About Working with Mercury Performance Testing	
and Business Availability Center Products	1314
Using QuickTest Performance Testing	
and Business Availability Center Features	1315
Designing QuickTest Tests for Use with LoadRunner	
or Business Process Monitor	1316
Inserting and Running Tests in LoadRunner	
or Business Process Monitor	1317
Measuring Transactions	1319
Using Silent Test Runner	1323

PART X: APPENDIX (Vol. 2)

Appendix A: Frequently Asked Questions.....	1329
Recording and Running Tests	1330
Programming in the Expert View.....	1331
Working with Dynamic Content	1332
Advanced Web Issues	1333
Standard Windows Environment.....	1336
Test Maintenance	1337
Testing Localized Applications.....	1340
Improving QuickTest Performance	1341
Index.....	I-1

Table of Contents

Welcome to This Guide

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

This chapter describes:	On page:
How This Guide Is Organized	xviii
Who Should Read This Guide	xx
QuickTest Professional Online Documentation	xx
Additional Online Resources	xxii
Documentation Updates	xxiii
Typographical Conventions	xxiv

How This Guide Is Organized

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

This guide contains the following parts:

Volume 1

Part I Introducing QuickTest Professional

Provides an overview of QuickTest and the main stages of the testing process, and describes how QuickTest identifies objects in your application.

Part II Creating Tests

Describes how to create tests, insert checkpoints, parameters, and output values, use regular expressions, and work with actions.

Part III Running and Debugging Tests

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

Part IV Configuring Basic Settings

Describes how to modify basic QuickTest test, global, and record and run settings to meet your testing needs.

Volume 2

Part V Working with Supported Environments

Explains how to work with QuickTest built-in add-ins, and includes environment-specific information for testing Web sites.

Part VI Working with Advanced Testing Features

Describes how to work with actions, virtual objects, recovery scenarios, configure object identification, and create Smart Identification definitions. It also describes how to work with user-defined functions and function libraries in QuickTest. In addition, it describes several programming techniques to create more powerful scripts, and describes how to enhance your test using the Expert View. It also describes how to automate QuickTest operations.

Part VII Managing and Merging Object Repositories

Describes how to manage and merge object repositories.

Part VIII Configuring Advanced Settings

Describes how to configure Web event recording, customize the Expert View and function library windows, and set testing options during a run session.

Part IX Working with Other Mercury Products

Describes how you can run tests 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 QuickTest interacts with Mercury Quality Center (formerly TestDirector), the Mercury centralized quality solution. This section also describes considerations for designing QuickTest tests for use with Mercury performance testing and application management products.

Part X Appendix

Provides information on frequently asked questions.

Who Should Read This Guide

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

QuickTest Professional Online Documentation

QuickTest Professional includes the following online documentation:

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

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

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

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

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

QuickTest Professional Help includes:

- **What's New in QuickTest** describes the newest features, enhancements, and supported environments in the latest version of QuickTest.
- **QuickTest User's Guide** describes how to use QuickTest to test your application.
- **QuickTest for Business Process Testing User's Guide** provides step-by-step instructions for using QuickTest to create and manage assets for use with Business Process Testing.

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

Choose **Help > QuickTest Professional Help**. Online Help is also available from specific QuickTest windows and dialog boxes by clicking in the window and pressing F1. You can also view a description, syntax, and examples for a QuickTest test object, method, or property by placing the cursor on it and pressing F1.

Note: Your QuickTest Help may contain additional items relevant to any QuickTest add-ins you have installed. For more information, refer to the relevant add-in documentation.

Additional Online Resources

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

Knowledge Base uses your default Web browser to open the Mercury Customer Support Web Site directly to the Knowledge Base landing page. Choose **Help > Knowledge Base**. The URL for this Web site is <http://support.mercury.com/cgi-bin/portal/CSO/kbBrowse.jsp>.

Customer Support Web Site uses your default Web browser to open the Mercury Customer Support Web site. This site enables you to browse the Mercury Support Knowledge Base and add your own articles. You can also post to and search user discussion forums, submit support requests, download patches and updated documentation, and more. Choose **Help > Customer Support Web Site**. The URL for this Web site is <http://support.mercury.com>.

Send Feedback enables you to send online feedback about QuickTest to the product team. Choose **Help > Send Feedback**.

Mercury Home Page uses your default Web browser to access Mercury's Web site. 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. Choose **Help > Mercury Home Page**. The URL for this Web site is <http://www.mercury.com>.

Mercury Best Practices contain guidelines for planning, creating, deploying, and managing a world-class IT environment. Mercury provides three types of best practices: Process Best Practices, Product Best Practices, and People Best Practices. Licensed customers of Mercury software can read and use the Mercury Best Practices available from the Customer Support site, <http://support.mercury.com>.

Documentation Updates

Mercury is continually 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 **Please Select Product**, select **QuickTest Professional**.

Note that if the required product 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 updated recently, **Updated** appears next to the document name.
- 4** Click a document link to download the documentation.

Typographical Conventions

This guide uses the following typographical conventions:

UI Elements	This style indicates the names of interface elements on which you perform actions, file names or paths, and other items that require emphasis. For example, “Click the Save button.”
<i>Arguments</i>	This style indicates method, property, or function arguments and book titles. For example, “Refer to the <i>Mercury User’s Guide</i> .”
<Replace Value>	Angle brackets enclose a part of a file path or URL address that should be replaced with an actual value. For example, <MyProduct installation folder>\bin .
Example	This style is used for examples and text that is to be typed literally. For example, “Type Hello in the edit box.”
CTRL+C	This style indicates keyboard keys. For example, “Press ENTER.”
Function_Name	This style indicates method or function names. For example, “The wait_window statement has the following parameters:”
[]	Square brackets enclose optional arguments.
{ }	Curly brackets indicate that one of the enclosed values must be assigned to the current argument.
...	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.
	A vertical bar indicates that one of the options separated by the bar should be selected.

Part I

Introducing QuickTest Professional

1

Introduction

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

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

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

QuickTest Professional 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. You can test non-English language applications, as long as the relevant Windows language support is installed on the computer on which QuickTest Professional is installed (**Start > Settings > Control Panel > Regional Options** or similar).

This chapter describes:	On page:
Testing with QuickTest	5
Understanding the Testing Process	6
Programming in the Expert View	10
Understanding Functions and Function Libraries	10
Managing the Testing Process Using Quality Center	11
Understanding Business Process Testing	12
Setting Required Access Permissions	13
Using the Sample Site	14
Modifying License Information	14
Updating the QuickTest Software	15

Testing with QuickTest

QuickTest Professional recognizes and learns the objects in your application so that you can design automated tests that perform the same types of operations and business processes that your customers do. You can then run these tests to check that your application works as expected.

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

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

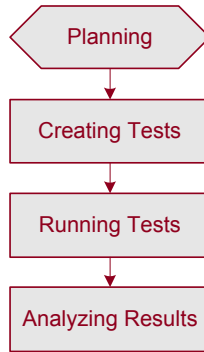
You can further enhance your test by adding and modifying steps. You can also create function libraries and call their functions from your test. For example, you can define functions and use them as keywords in your test.

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

Note: Many QuickTest operations are performed using the mouse. In accordance with 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 QuickTest operations using shortcut keys. For a list of shortcut keys, see “Performing Commands Using Shortcut Keys” on page 45.

Understanding the Testing Process

Testing with QuickTest involves the following main stages:



Planning

Before beginning to create a test, you should plan it and prepare the required infrastructure. For example, determine the functionality you want to test, and decide which information you want to check during the test run.

Before preparing the required infrastructure, you should spend time analyzing your application and determining which objects and operations are used by the set of business processes that need to be tested. You should also determine which operations require customized keywords to provide additional functionality.

For more information, see “Planning and Preparing to Create a Test” on page 81.

Creating Tests

You create a test either by building an object repository and adding steps manually or by recording a session on your application. You can create steps using the table-like, graphical Keyword View using keyword-driven functionality—or you can use the Expert View, if you prefer programming steps directly in VBScript.

Every test is composed of one or more actions. At its most basic level, each action contains steps that duplicate the activities that you or another user might perform when using your application or Web site. You can enhance the testing process by modifying your test with special testing options and/or with programming statements.

By default, each test begins with a single action. You can organize your test by dividing it into multiple actions. This is similar to creating separate modules, or logical units, for testing various parts of your application or Web site.

When you create your test, you:

- Add steps to your test:
 - Build an object repository and use these objects to add keyword-driven steps manually in the Keyword View or Expert View.

The object repository should contain all the objects that you want to test in your application or Web site. For more information on building an object repository, see Chapter 6, “Working with Test Objects.”

Create steps by selecting items and operations in the Keyword View and entering information as required. For more information, see Chapter 5, “Working with the Keyword View.” Advanced users can also add steps using the Expert View. For more information, see Chapter 34, “Working in the Expert View and Function Library Windows.”

- Record a session on your application or Web site.

As you navigate through your application or site, QuickTest graphically displays each step you perform as a row in the Keyword View. A step is something that causes or makes a change in your site or application, such as clicking a link or image, or submitting a data form. In the Expert View, these steps are displayed as lines in a test script (VBScript). The **Documentation** column of the Keyword View also displays a description of each step in easy-to-understand sentences. For more information, see Chapter 4, “Designing Tests.”

- Insert checkpoints into your test.

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 8, “Understanding Checkpoints.”

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

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

When you parameterize your test, QuickTest substitutes the fixed values in your test with parameters. When you use Data Table parameters, QuickTest uses the values from a different row in the Data Table for each iteration of the test or action. (Each run session that uses a different set of parameterized data is called an iteration.) For more information, see Chapter 16, “Parameterizing Values.”

You can also use output values to extract data from your test. An **output value** is a value retrieved during the run session and entered into your Data Table or saved as a variable or a parameter. You can subsequently use this output value as input data in your test. This enables you to use data retrieved during a run session in other parts of the test. For more information, see Chapter 17, “Outputting Values.”

- Add user-defined keywords and functions by creating function libraries and calling their functions from your test. For more information, see Chapter 35, “Working with User-Defined Functions and Function Libraries.”
- Use the many functional testing features included in QuickTest to enhance your test and/or add programming statements to achieve more complex testing goals.

Running Tests

After you create your test, you run it.

- Run your test to check your site or application.

The test starts running from the first line in your test and stops at the end of the test. While running, QuickTest connects to your Web site or application and performs each operation in your test, including any checkpoints, such as checking any text strings, objects, tables, and so forth. If you parameterized your test with Data Table parameters, QuickTest repeats the test (or specific actions in your test) for each set of data values you defined. For more information, see Chapter 23, “Running Tests.”

- Run your test to debug it.

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

Analyzing Results

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

- View the results in the Test Results window.

After you run your test, you can view the results of the run in the Test Results window. You can view a summary of your results as well as a detailed report. For more information, see Chapter 24, “Analyzing Test Results.”

- Report defects detected during a run session.

If you have access to Quality Center, the Mercury centralized quality solution, you can report the defects you discover to the project database. You can instruct QuickTest to automatically report each failed step in your test, or you can report them manually from the Test Results window. For more information, see Chapter 45, “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. The test is composed of statements written in VBScript (Microsoft Visual Basic Scripting Edition) that correspond to the steps and checks displayed in the Keyword View. For more information, see Chapter 34, “Working in the Expert View and Function Library Windows.”

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

Understanding Functions and Function Libraries

If you have sets of steps that are repeated in several actions or tests, you may want to consider creating and using user-defined functions. A user-defined function encapsulates an activity (or a group of steps that require programming) into a keyword (also called an operation). By using user-defined functions in your tests, your tests are shorter, and easier to design, read, and maintain.

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

When you create a function, you can insert it directly in an action to make it available only within that action, or you can insert it in a function library to make it available to any test that is associated with that function library. For more information, see Chapter 35, “Working with User-Defined Functions and Function Libraries.”

Managing the Testing Process Using Quality Center

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

In QuickTest, you can create tests and then save them directly to your Quality Center project. For more information, see Chapter 45, “Working with Quality Center.” You can also run QuickTest tests from Quality Center and then use Quality Center to review and manage the results. For more information, refer to the *Mercury Quality Center User's Guide*.

Finally, you can use Quality Center with Business Process Testing support to create business process tests, which are comprised of the business components you create either in QuickTest or Quality Center (with Business Process Testing support). For more information, see Chapter 46, “Working with Business Process Testing.”

Understanding Business Process Testing

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

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

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

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

For more information on using QuickTest with Business Process Testing, refer to the *Mercury QuickTest Professional for Business Process Testing User's Guide*.

Setting Required Access Permissions

You must make sure the following access permissions are set in order to run QuickTest Professional.

Permissions Required to Run QuickTest Professional

You must have the following file system permissions:

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

You must have the following registry key permissions:

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

Permissions Required When Working with Quality Center

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

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

Using the Sample Site

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

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

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

Modifying License Information

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

- a 14-day **demo** license
- a permanent **seat** license that is specific to the computer on which it is installed
- a network-based **concurrent** license that can be used by multiple QuickTest users

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

If needed, you can request a new seat license on the Mercury Customer Support Web site. The URL for the License Request Web site is <http://support.mercury.com/license>.

For information on modifying your license information, refer to the *QuickTest Professional Installation Guide*.

Updating the QuickTest Software

By default, QuickTest automatically checks for online software updates each time you start the application. You can also manually check for updates at any time by choosing **Help > Check for Updates** from within QuickTest, or by choosing **Start > Programs > QuickTest Professional > Check for Updates**.

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

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

2

QuickTest at a Glance

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

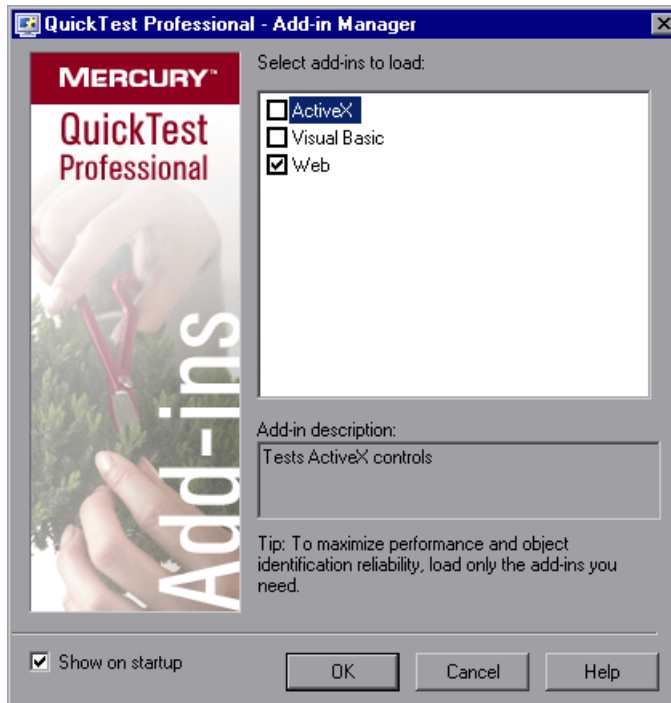
This chapter describes:	On page:
Starting QuickTest	18
QuickTest Window	20
Keyword View	23
Expert View	25
Function Library	26
Active Screen	27
Information Pane	28
Missing Resources Pane	29
Data Table	30
Debug Viewer Pane	30
Customizing the QuickTest Window Layout	31
Working With Multiple Documents	39
Using QuickTest Commands	41
Browsing the QuickTest Professional Program Folder	53
Viewing Product Information	57

Starting QuickTest



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

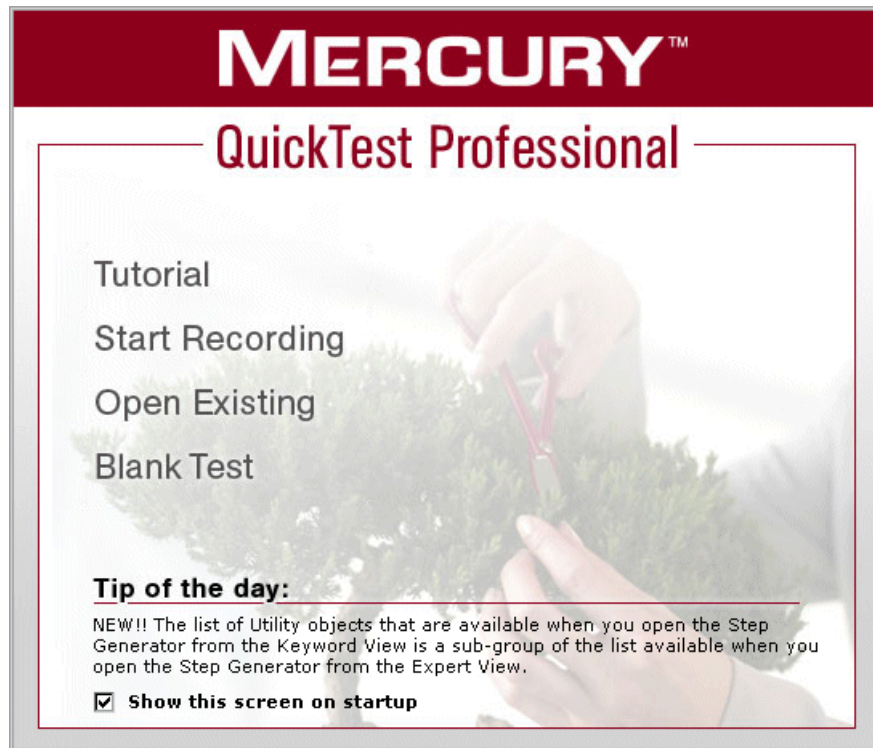
The first time you start QuickTest, the Add-in Manager dialog box opens.



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

For more information on loading add-ins, see “Loading QuickTest Add-ins” on page 811.

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.

QuickTest Window

The QuickTest window displays your testing document(s) in the document area.

You can work on one test and one or more function libraries simultaneously. (For your convenience, you can display one active document in the document area, or you can cascade or tile your open documents.) For more information, see “Working With Multiple Documents” on page 39.

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

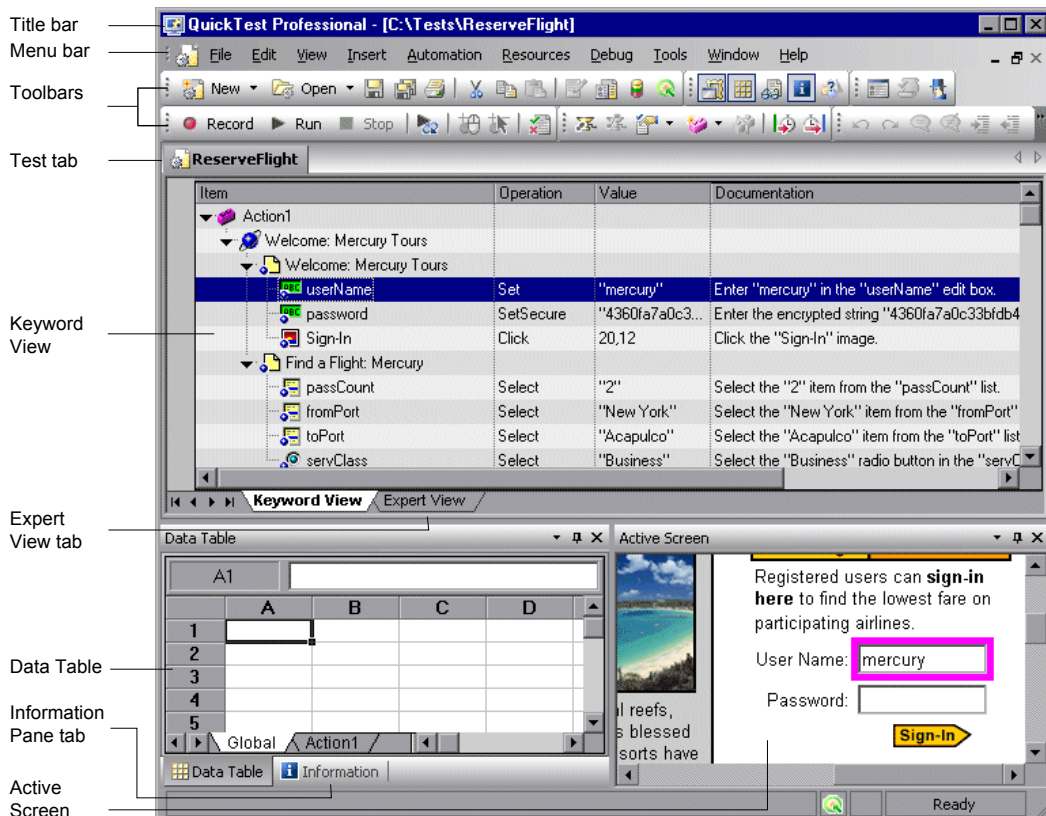
- **Test.** Enables you to create, view, and modify your test in Keyword View or Expert View (described below).
- **Function Library.** Enables you to create, view, and modify functions (operations) for use with your test. For more information, see Chapter 35, “Working with User-Defined Functions and Function Libraries.”

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

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

- **View toolbar.** Contains buttons to assist you in viewing your document.
- **Action toolbar.** Contains buttons and a list of actions, enabling you to view the details of an individual action or the entire test flow. (Not displayed by default)
- **Document tabs and scroll arrows.** Enables you to navigate open documents in the document area by selecting the tab of the document you want to activate (bring into focus). When there is not enough space in the document area to display all of the tabs simultaneously, you can use the left and right arrows to scroll between your open documents.
- **Keyword View.** Contains each step, and displays the object hierarchy, in a modular, icon-based table. For more information, see Chapter 5, “Working with the Keyword View.”
- **Expert View.** Contains each step as a VBScript line. In object-based steps, the VBScript line defines the object hierarchy. For more information, see Chapter 34, “Working in the Expert View and Function Library Windows.”
- **Active Screen.** Provides a snapshot of your application as it appeared when you performed a certain step during the recording session.
- **Information pane.** Displays a list of syntax errors found in your test and function library scripts.
- **Missing Resources pane.** Provides a list of the resources that are specified in your test but cannot be found, such as missing calls to actions, unmapped shared object repositories, and parameters that are connected to shared object repositories. (Not displayed by default)
- **Data Table.** Assists you in parameterizing your test. The Data Table contains the **Global** tab and a tab for each action.
- **Debug Viewer pane.** Assists you in debugging your document. The Debug Viewer pane contains the **Watch**, **Variables**, and **Command** tabs. (Not displayed by default)
- **Status bar.** Displays the status of the QuickTest application and other relevant information.

You can customize the layout of the QuickTest window by moving, resizing, displaying, or hiding most of the elements. QuickTest remembers your preferred layout settings and opens subsequent sessions with your customized layout. For more information, see “Customizing the QuickTest Window Layout” on page 31.



Changing the Appearance of the QuickTest Window

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

To change the appearance of the main QuickTest window:

In the QuickTest window, choose **View > Window Theme**, and then select the way the window should appear from the list of available themes. For example, you can apply a Microsoft Office 2000 or Microsoft Windows XP theme.

Note: You can apply the Microsoft Windows XP theme to the QuickTest window only if your computer is set to use a Windows XP theme.

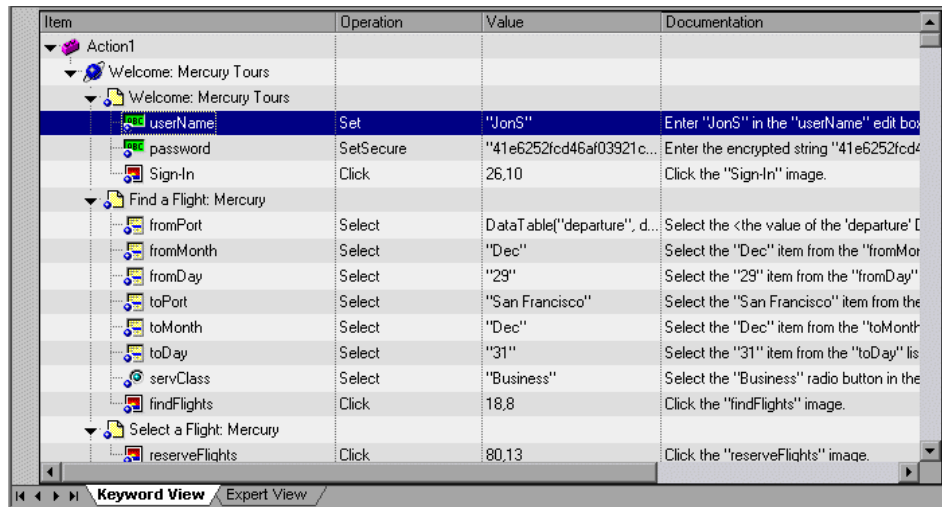
Tip: You can also change the theme used for the Test Results window. For more information, see “Changing the Appearance of the Test Results Window” on page 638.

Keyword View

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

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

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



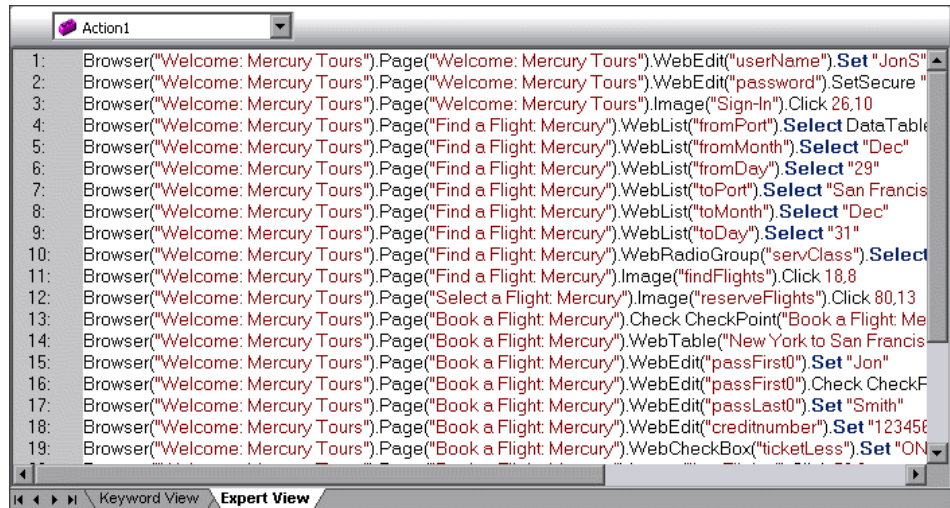
Item	Operation	Value	Documentation
▼ Action1			
▼ Welcome: Mercury Tours			
▼ Welcome: Mercury Tours			
userName	Set	"JonS"	Enter "JonS" in the "userName" edit box
password	SetSecure	"41e6252fcd46af03921c..."	Enter the encrypted string "41e6252fcd46af03921c..."
Sign-In	Click	26,10	Click the "Sign-In" image.
▼ Find a Flight: Mercury			
fromPort	Select	DataTable["departure", d...]	Select the <the value of the 'departure' [
fromMonth	Select	"Dec"	Select the "Dec" item from the "fromMonth"
fromDay	Select	"29"	Select the "29" item from the "fromDay"
toPort	Select	"San Francisco"	Select the "San Francisco" item from the
toMonth	Select	"Dec"	Select the "Dec" item from the "toMonth"
toDay	Select	"31"	Select the "31" item from the "toDay" lis
servClass	Select	"Business"	Select the "Business" radio button in the
findFlights	Click	18,8	Click the "findFlights" image.
▼ Select a Flight: Mercury			
reserveFlights	Click	80,13	Click the "reserveFlights" 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. For more information on using the Keyword View, see Chapter 5, “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.

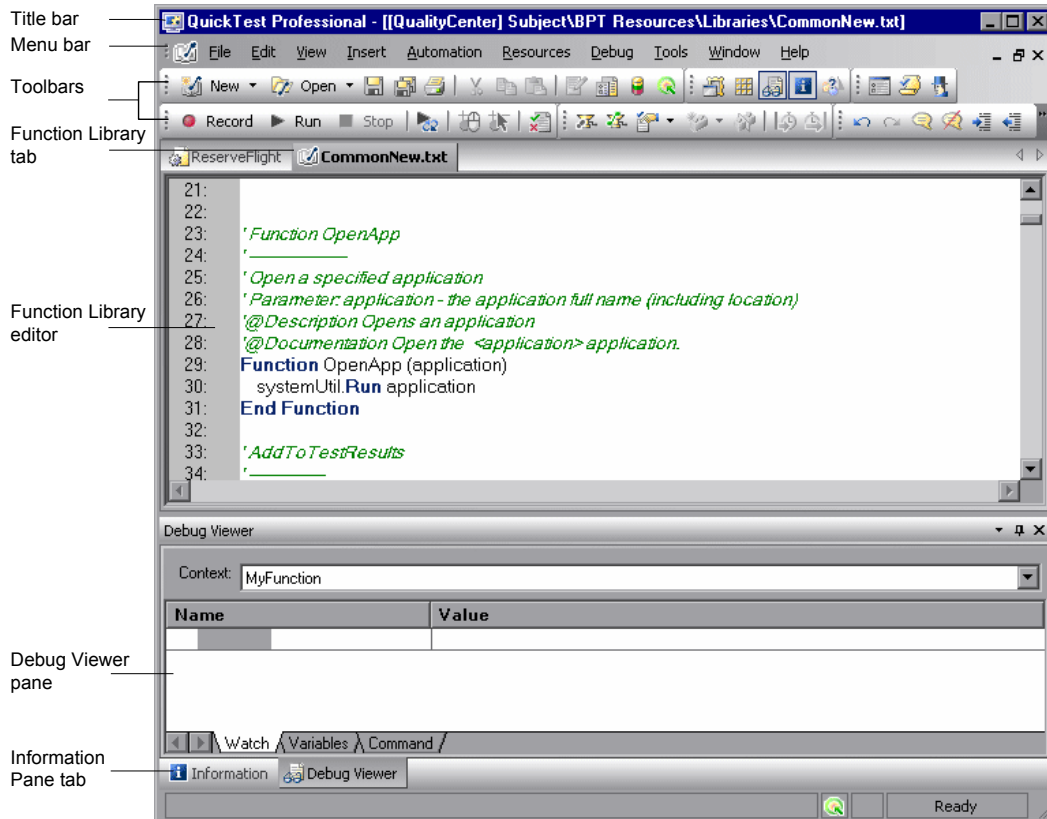
Expert View

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 34, “Working in the Expert View and Function Library Windows.”



Function Library

QuickTest provides a built-in editor that enables you to create and debug function libraries using the same editing features that are available in the Expert View. Each function library is a separate QuickTest document containing VBScript functions, subroutines, classes, modules, and so forth. Each function library opens in its own window, in addition to the test that is already open. You can work on one or several function libraries at the same time. After you finish editing a function library, you can close it, leaving your QuickTest session open. You can also close all open function libraries simultaneously. For more information, see Chapter 35, “Working with User-Defined Functions and Function Libraries.”



Active Screen



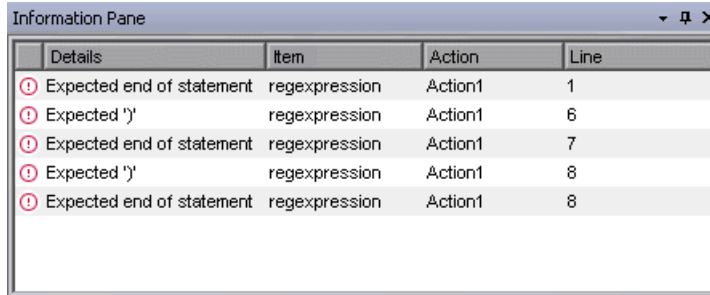
The Active Screen provides a snapshot of your application as it appeared when you performed a certain step during a recording session. Additionally, depending on the Active Screen capture options that you used while recording, the page displayed in the Active Screen can contain detailed property information about each object displayed on the page. To view the Active Screen, click the **Active Screen** button or choose **View > Active Screen**. For more information, see “Working with the Active Screen” on page 219.



Information Pane



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

A screenshot of the 'Information Pane' window in QuickTest Professional. The window has a title bar with a dropdown arrow, a maximize button, and a close button. Below the title bar is a table with four columns: 'Details', 'Item', 'Action', and 'Line'. The table contains five rows of error messages, each preceded by a red circle with a white 'i' icon. The errors are: 'Expected end of statement' at line 1, 'Expected \')' at line 6, 'Expected end of statement' at line 7, 'Expected \')' at line 8, and 'Expected end of statement' at line 8. All items are 'regexpression' and the action is 'Action1'.

Details	Item	Action	Line
Expected end of statement	regexpression	Action1	1
Expected \')	regexpression	Action1	6
Expected end of statement	regexpression	Action1	7
Expected \')	regexpression	Action1	8
Expected end of statement	regexpression	Action1	8

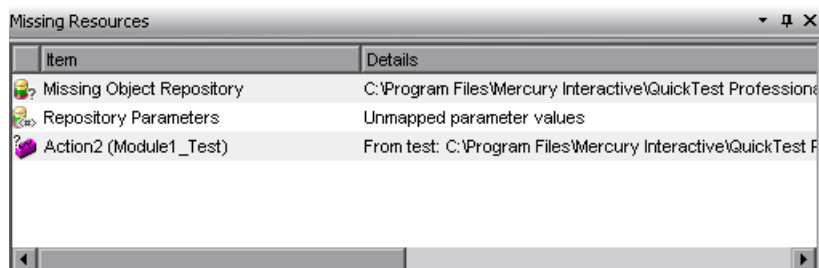
When you switch from the Expert View to the Keyword View, QuickTest automatically checks for syntax errors in your script, and shows them in the Information pane. If the Information pane is not currently displayed, QuickTest automatically opens it when a syntax error is detected.

You can double-click a syntax error to locate the error in the script or function library, and then correct it. For more information, see “Handling VBScript Syntax Errors” on page 1003.

Missing Resources Pane



The Missing Resources pane provides a list of the resources that are specified in your test but cannot be found, such as missing calls to actions, unmapped shared object repositories, and parameters that are connected to shared object repositories. To show or hide the Missing Resources pane, choose **View > Missing Resources** or click the **Missing Resource** button.



Each time you open your test or function library, QuickTest automatically checks that all specified resources are accessible. If it finds any resources that are not accessible, QuickTest lists them in the Missing Resources pane. If the Missing Resources pane is not currently displayed, QuickTest automatically opens it when a missing resource is detected.

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

For more information, see “Handling Missing Resources” on page 505.

Data Table



The Data Table contains one Global tab plus an additional tab for each action, or test step grouping, in your test. The Data Table assists you in parameterizing your test. 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. For more information, see Chapter 20, “Working with Data Tables.”

Debug Viewer Pane



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

Watch

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

Variables

During a run session, the Variables tab displays 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 run a line of script to set or modify the current value of a variable or VBScript object in your test or function library. 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 Function Libraries.”

Customizing the QuickTest Window Layout

You can customize the layout of the QuickTest window, and you can restore the default layout. When customizing the window, you can move and resize panes, select to show or auto-hide panes, create tabbed panes, select which toolbars to display, and so forth.

Note: When customizing or restoring the QuickTest window layout, the layout is customized or restored for all document types.

Moving Panes

You can move the QuickTest window panes to suit your own personal preferences. You can rearrange the panes, and you can also change a pane to a tabbed pane, and vice versa.

While dragging a pane, markers are displayed on the QuickTest window. If you hold the cursor over one of these markers, the area represented by the marker is shaded, enabling you to preview the window layout if the pane is moved to the selected position.

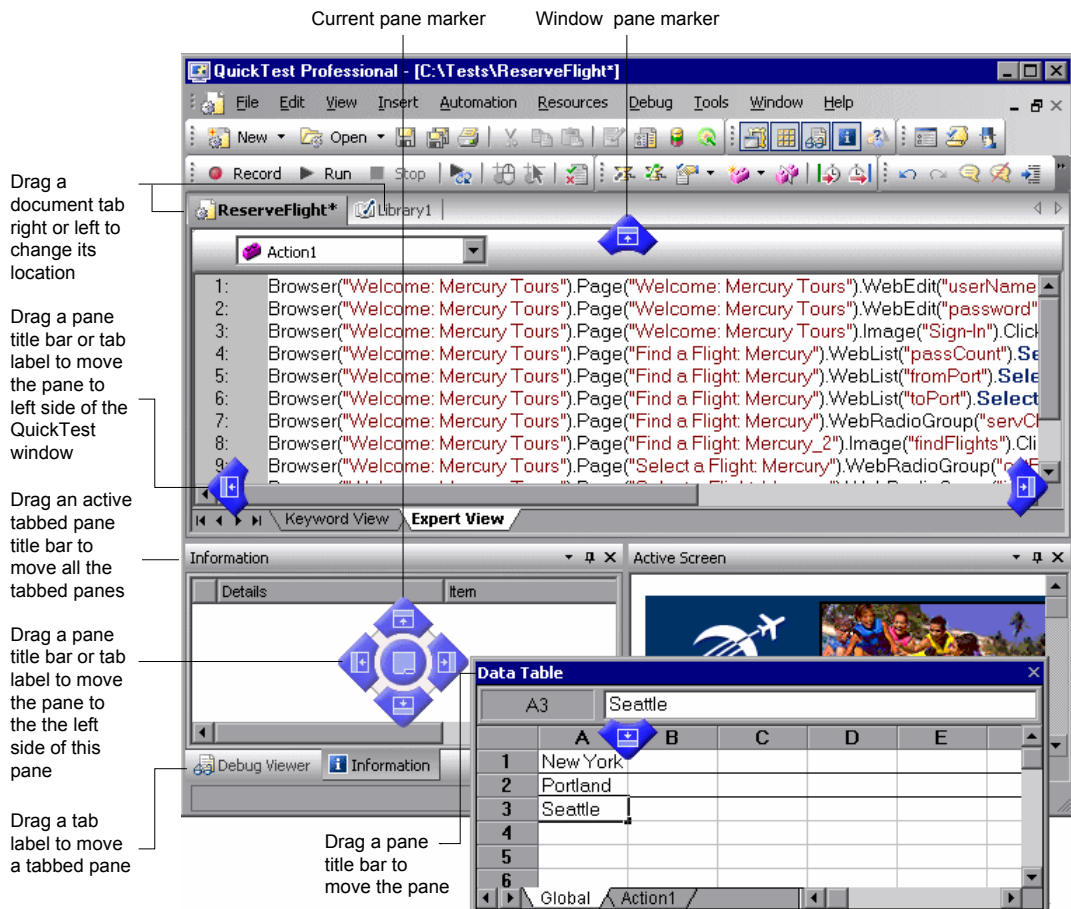
Tip: To move a dockable pane without snapping it into place, press CTRL while dragging it to the required location.

To move panes:

- 1 In the QuickTest window, drag the title bar or tab of the pane you want to move.

Tip: If the required pane is not displayed in the QuickTest window, you can select it from the **View** menu.

For example, you can move the Data Table tabbed pane located at the bottom left to be a new pane at the top right of the window. As you drag the pane, markers are displayed in the active pane and on each edge of the QuickTest window.








Tips:

To move a single tabbed pane, drag the tab label. Once you start dragging the tabbed pane, the tab is removed, and its title bar is displayed.

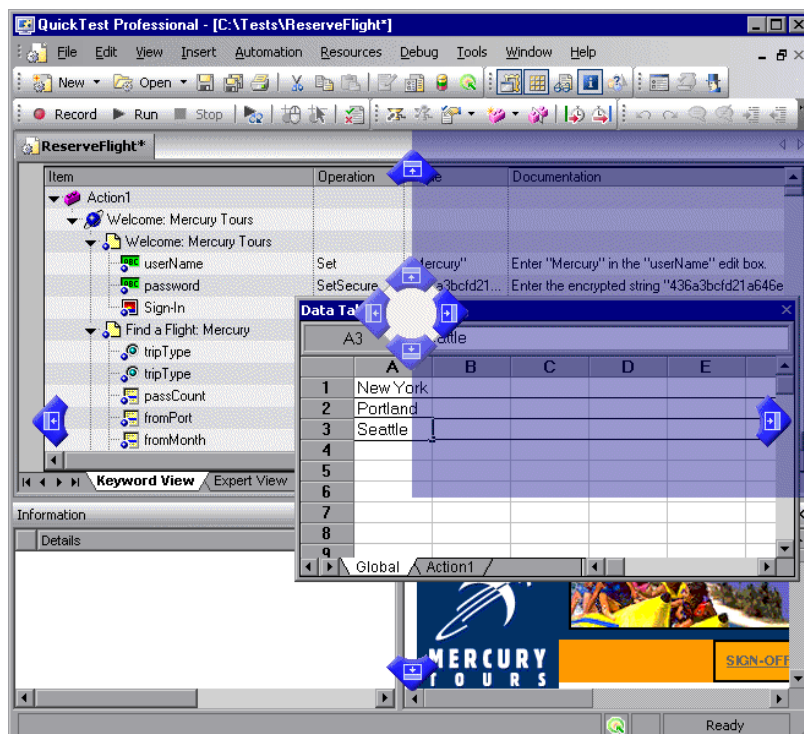
To move all the tabbed panes, drag the title bar of the active tabbed pane.

The following markers are displayed:

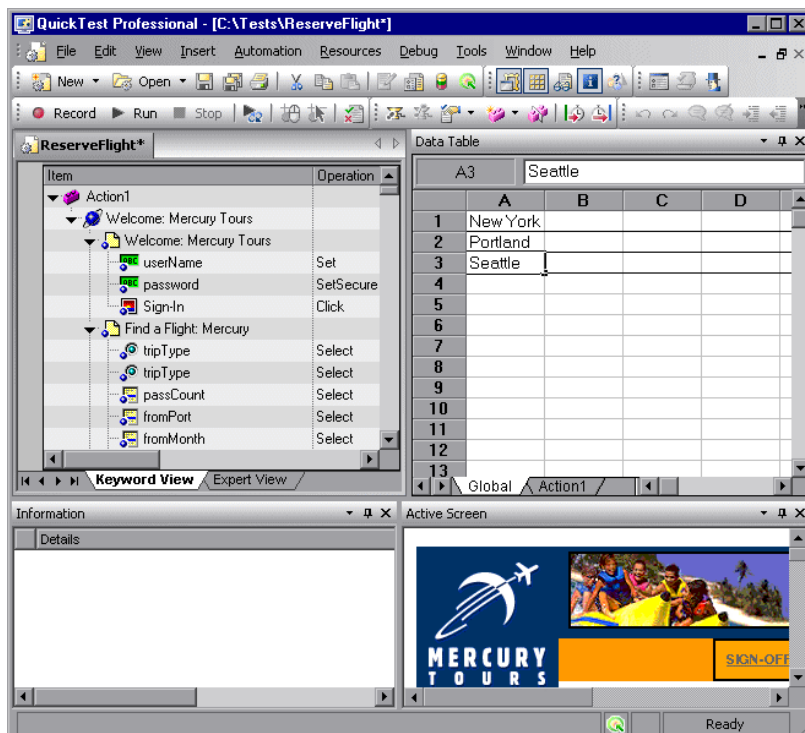
Type	Marker	Description
Current pane markers		<p>Enables you to:</p> <ul style="list-style-type: none"> position the pane as a new pane in the top, bottom, left or right half, or middle of the active pane, according to the arrow marker selected when you release the mouse button. position the pane as a new tabbed pane in the active window, by releasing the mouse button while selecting the center marker. <p>Note: The center marker is displayed only if you are dragging a pane onto an existing pane (other than the document pane).</p>
Window pane markers		Enables you to position the pane across the top of the QuickTest window.
		Enables you to position the pane across the right side of the QuickTest window.
		Enables you to position the pane across the bottom of the QuickTest window.
		Enables you to position the pane across the left side of the QuickTest window.



- 2 Drag the Data Table and hold the cursor over the active pane right-arrow marker, as shown below. A shaded area is displayed, indicating the new location of the pane, as shown below.



- 3 Release the mouse button. The Data Table snaps into place and is displayed as a new pane in the shaded area.



Tip: You can also leave the pane as a floating pane anywhere on the QuickTest window, or on your screen. For more information on floating panes, see "Showing and Hiding Panes" on page 36.

- 4 Repeat this procedure for each pane you want to move.

Showing and Hiding Panes

After you move the panes to their default positions, you can decide whether these panes should be displayed at all times, or whether you want to auto-hide them, and only display them as required.

Panes can have one of the following states—docked or floating:

- **Docked panes.** Docked panes are fixed in a set position relative to the rest of the application. For example, when you move a pane to a position indicated by a marker, the pane is docked in that position.

You can decide whether to continuously display the docked panes in the QuickTest window, or to auto-hide them. Auto-hiding means that the pane is displayed as a side-tab on the edge of the QuickTest window, and is displayed only when you hold the cursor over the tab. After you select a different pane or side-tab, the auto-hidden pane closes and is displayed as a side-tab.

Note: If you auto-hide the Information pane, it is automatically displayed when syntax errors are detected in a test script.

By default, auto-hidden panes open across the QuickTest window, according to their position in the QuickTest window. For example, if the docked pane was positioned on the right side of the QuickTest window, it is displayed as a side tab on the right edge of the QuickTest window, and is displayed across the right side of the QuickTest window when selected.



Tip: To auto-hide all the tabbed panes, select the title bar of the active tabbed pane, right-click and choose **Auto Hide**. The tabbed panes are displayed as a group of side-tabs on the edge of the QuickTest window, and each pane is displayed only when you hold the cursor over that side-tab.



- **Floating panes.** Floating panes are displayed on top of all other windows. They can be dragged to any position on your screen, even outside the QuickTest window. Floating panes have their own title bars.

Note: You cannot auto-hide floating panes or individual tabbed panes.

To show or hide panes:

In the QuickTest window, select the pane you want to auto-hide, and display as a side-tab on one of the edges of the QuickTest window. The following buttons may be displayed on the title bar:

Button	Description
	<p>The Menu button enables you to select any of the following:</p> <ul style="list-style-type: none"> • Floating. Opens the pane on top of all the other windows and panes, with its own title bar • Docking. Docks the pane to the QuickTest window. • Auto-hide. Displays the pane as a side-tab either at the top or bottom of the QuickTest window, or on one of the edges, according to its position in the QuickTest window. • Hide. Closes the pane.
	<p>The Auto Hide button hides the pane.</p> <p>The pane is displayed as a side-tab either at the top or bottom of the QuickTest window, or on one of the edges, according to its position in the QuickTest window.</p> <p>To display the pane, hold the cursor over the side-tab. The button toggles to the Dock button, shown below.</p>

Button	Description
	<p>The Dock button docks the pane to the QuickTest window.</p> <p>The pane returns the position it was in before it was hidden, and the button toggles to the Auto Hide button, shown above.</p>
	<p>The Close button closes the pane.</p> <p>The pane is removed from the QuickTest window. To reopen the pane, select it from the View menu.</p> <p>Tip: You can also close a pane by right-clicking and choosing Hide from the context menu.</p>

Tips:

To auto-hide all the tabbed panes, select the title bar of the active tabbed pane, right-click and choose **Auto Hide**. The tabbed panes are displayed as a group of side-tabs on the edge of the QuickTest window, and each pane is displayed only when you hold the cursor over that side-tab.

You can float a pane by right-clicking the title bar, and choosing **Floating** from the context menu. The pane opens on top of all the other windows and panes, with its own title bar. To dock the pane, double-click the title bar, or right-click the title bar and choose **Docking**. The pane returns to its previous position in the QuickTest window.

Showing and Hiding Toolbars

You can show or hide toolbars using the **View > Toolbars** menu options.



You can float a toolbar by moving your cursor over the toolbar handle on the left of the toolbar and then dragging the toolbar to the required position. The toolbar is displayed with a title bar.



You can double-click the title bar of the menu to dock the menu and return it to its previous position in the QuickTest window, or you can close it by clicking the **Close** button.

Restoring the Default Layout of the QuickTest Window

You can restore the default QuickTest window layout for all document types at any time.

To restore the default layout:

- 1** Choose **Tools > Options**. The Options dialog box is displayed.
- 2** In the General tab, click the **Restore Layout** button. The panes and toolbars of all document types are restored to their default size and location.

Note: For more information on the Options dialog box, see Chapter 25, “Setting Global Testing Options.”

Working With Multiple Documents

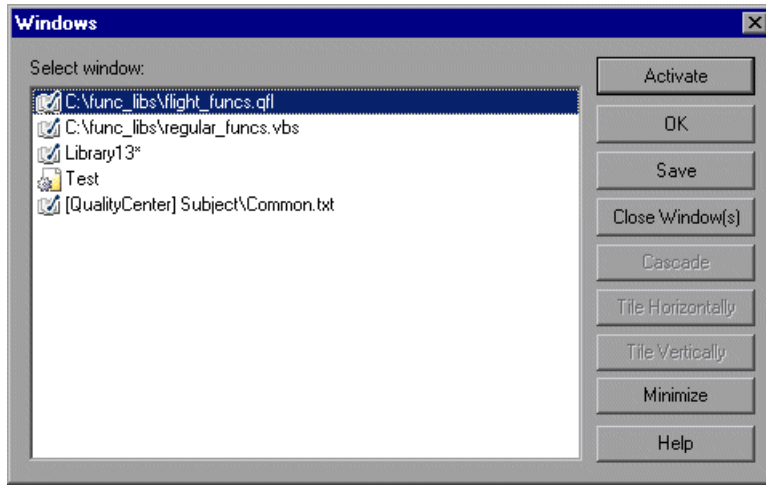
QuickTest enables you to open and work on one test at a time. In addition, you can open and work on multiple function libraries simultaneously. You can open any function library, regardless of whether it is associated with the currently open test.

The **Window** menu options enable you to locate and activate (bring into focus) an open document window, select how the open document windows are arranged in the QuickTest window, or close all the open function library windows.

You can also use the Windows dialog box to manage your open QuickTest document windows.

To work with multiple documents using the Windows dialog box:

- 1 Choose **Window > Windows**. The Windows dialog box opens.



The Windows dialog box displays a list of the open document windows, including the open test, component, or application area, as well as all the currently open function library windows.

- 2 The Windows dialog box contains the following buttons, enabling you to manage your open documents:

Button	Description
Activate	Brings the selected document into focus in the QuickTest window.
OK	Closes the Windows dialog box.
Save	Saves the selected document(s).
Close Window(s)	Closes the selected function libraries.
Cascade	Arranges the selected documents in a cascading order that overlaps.

Button	Description
Tile Horizontally	Arranges the selected documents side-by-side horizontally, without overlapping.
Tile Vertically	Arranges the selected documents side-by-side vertically, without overlapping.
Minimize	Minimizes the selected documents.
Help	Displays the QuickTest Professional Help topic for this dialog box.

- 3 Click **OK** to close the Windows dialog box.

Using QuickTest Commands

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

Most commands are available from the menu bar or by pressing shortcut keys. You can perform frequently used QuickTest commands by clicking buttons on the toolbars.

Choosing Commands from a Menu

You can select most QuickTest commands from the menus in the menu bar. Shortcut keys, where available, are displayed with the corresponding menu commands. For more information, see “Performing Commands Using Shortcut Keys” on page 45.

Clicking Commands on a Toolbar

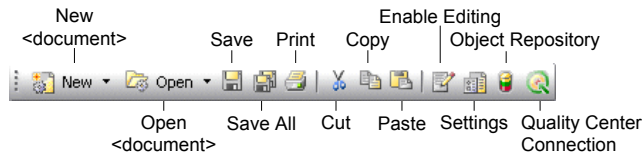
You can perform some QuickTest commands by clicking buttons on the toolbars. QuickTest has eight built-in toolbars—the **Standard** toolbar, the **Edit** toolbar, the **Automation** toolbar, the **View** toolbar, the **Insert** toolbar, the **Tools** toolbar, the **Debug** toolbar, and the **Action** toolbar.

Note: You can display, hide, or move the toolbars, but you cannot customize them.

Standard Toolbar

The **Standard** toolbar contains buttons for managing a test or function library. For more information on managing your test, see Chapter 4, “Designing Tests.” For more information on managing business process tests, see Chapter 46, “Working with Business Process Testing.” For more information on working with function libraries, see Chapter 35, “Working with User-Defined Functions and Function Libraries.”

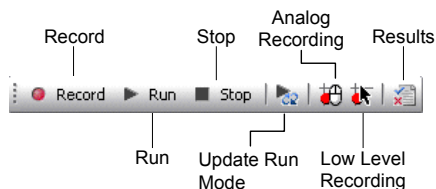
The following buttons are displayed on the **Standard** toolbar:



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

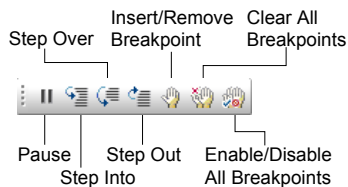
Automation Toolbar

The **Automation** toolbar contains buttons for recording and running your test. The following buttons are displayed on the **Automation** toolbar:



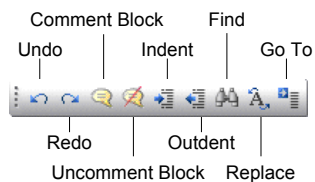
Debug Toolbar

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



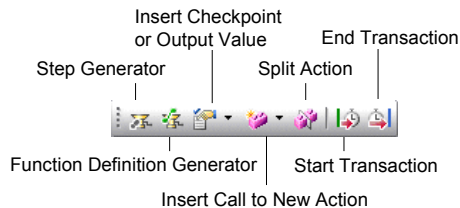
Edit Toolbar

The **Edit** toolbar contains buttons for the commands used when editing your test or function library. The following buttons are displayed on the **Edit** toolbar:



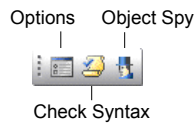
Insert Toolbar

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



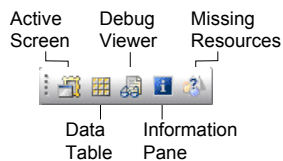
Tools Toolbar

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



View Toolbar

The **View** toolbar contains buttons for viewing different elements of the QuickTest window. The following buttons are displayed on the **View** 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 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 on actions, see Chapter 18, “Working with Actions” and Chapter 30, “Working with Advanced Action Features.”

Performing 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.
New > Business Component	CTRL+SHIFT+N	Creates a new business component.
New > Application Area	CTRL+ALT+N	Creates a new application area.
New > Function Library	SHIFT+ALT+N	Creates a new function library.

Command	Shortcut Key	Function
Open > Test	CTRL+O	Opens an existing test.
Open > Business/Scripted Component	CTRL+SHIFT+O	Opens an existing business or scripted component.
Open > Application Area	CTRL+ALT+O	Opens an existing application area.
Open > Function Library	SHIFT+ALT+O	Opens an existing function library.
Save	CTRL+S	Saves the active document.
Export Test to Zip File	CTRL+ALT+S	Creates a zip file of the active document.
Import Test from Zip File	CTRL+ALT+I	Imports a document from a zip file.
Convert to Scripted Component	CTRL+ALT+C	Converts a business component to a scripted component.
Print	CTRL+P	Prints the active document.

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

Command	Shortcut Key	Function
Undo	CTRL+Z	Reverses the last command or deletes the last entry you typed.
Redo	CTRL+Y	Reverses the action of the Undo command.
Cut	CTRL+X	Removes the selection from your document.
Copy	CTRL+C	Copies the selection from your document.
Paste	CTRL+V	Pastes the selection to your document.

Command	Shortcut Key	Function
Delete	DELETE	Deletes the selection from your document.
Action > Rename Action	SHIFT+F2	Changes the name of an action.
Step Properties > Comment Properties	CTRL+ENTER; ALT+ENTER	Displays the Comment Properties dialog box of a selected object.
Step Properties > Object Properties	CTRL+ENTER; ALT+ENTER	Displays the Object Properties dialog box of a selected object.
Step Properties > Report Properties	CTRL+ENTER; ALT+ENTER	Displays the Report Properties dialog box of a selected object.
Find	CTRL+F	Searches for a specified string.
Replace	CTRL+H	Searches and replaces a specified string.
Go To	CTRL+G	Moves the cursor to a particular line in the test.
Bookmarks	CTRL+B	Creates bookmarks in your script for easy navigation.
Advanced > Comment Block	CTRL+M	Comments out the current row, or selected rows.
Advanced > Uncomment Block	CTRL+SHIFT+M	Removes the comment formatting from the current or selected rows.
Advanced > Go to Function Definition	ALT+G	Navigates to the definition of the selected function.
Advanced > Complete Word	CTRL+SPACE	Completes the word when you type the beginning of a VBScript method or object.
Advanced > Argument Info	CTRL+SHIFT+ SPACE	Displays the syntax of a method.

Command	Shortcut Key	Function
Advanced > Apply “With” to Script	CTRL+W	Generates With statements for the action displayed in the Expert View.
Advanced > Remove “With” Statements	CTRL+SHIFT+W	Converts any With statements in the action displayed in the Expert View to regular (single-line) VBScript statements.

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 Generator	F7	Opens the Step Generator.
New Step	F8; INSERT	Inserts a new step in the Keyword View.
New Step After Block	SHIFT+F8	Inserts a new step after a conditional or loop block in the Keyword View.

You can perform the following **Automation** 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.

Command	Shortcut Key	Function
Run from Step	CTRL+F5	Starts a run session from the selected step.
Analog Recording	SHIFT+ALT+F3	Starts recording in analog recording mode.
Low Level Recording	CTRL+SHIFT+F3	Starts recording in low level recording mode.

You can perform the following **Resources** menu command by pressing the corresponding shortcut key:

Command	Shortcut Key	Function
Object Repository	CTRL+R	Opens the Object Repository dialog box.

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

Command	Shortcut Key	Function
Step Into	F11	Runs only the current line of the script. If the current line calls a method, the method is displayed in the view but is not performed.
Step Over	F10	Runs only the current line of the script. When the current line calls a method, the method is performed in its entirety, but is not displayed in the view.
Step Out	SHIFT+F11	Runs to the end of the method then pauses the run session. (Available only after running a method using Step Into .)
Run to Step	CTRL+F10	Runs until the current step.

Command	Shortcut Key	Function
Add to Watch	CTRL+T	Adds the selected item to the Watch tab.
Insert/Remove Breakpoint	F9	Sets or clears a breakpoint in the test.
Enable/Disable Breakpoint	CTRL+F9	Enables or disables a breakpoint in the test.
Clear All Breakpoints	CTRL+SHIFT+F9	Deletes all breakpoints in the test.

You can perform the following **Tools** menu command by pressing the corresponding shortcut key:


Command	Shortcut Key	Function
Check Syntax	CTRL+7	Checks the syntax of the active document.

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.
Edit > Paste	CTRL+V	Pastes the contents of the Clipboard to the current table selection.
Edit > Clear > Contents	CTRL+DEL	Clears the contents from the current selection.

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

You can perform the following special options using shortcut keys:

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

Browsing the QuickTest Professional Program Folder

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

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

- **Add-ins.** Provides links to Readme documentation for any external QuickTest add-ins installed on your computer.
- **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 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 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 *QuickTest Professional for Business Process Testing 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 any installed add-ins), and the *Microsoft VBScript Reference*.

- **Tutorial.** Opens the QuickTest Professional tutorial, which teaches you basic QuickTest skills and shows you how to start testing your applications.
- **QuickTest Professional Code Samples Plus.** Opens the QuickTest Professional Code Samples Plus Help, which in previous versions provided answers to frequently asked questions, tips and tricks, sample function libraries, and code and SDK samples. The utilities, features, and information that were available using QuickTest Professional Code Samples Plus have been incorporated into the main QuickTest Professional application.
- **Sample Applications.** Contains the following links to sample applications that you can use to practice testing with QuickTest:
 - **Mercury Tours Web Site.** Opens a sample flight reservation Web application. This Web application is used as a basis for the QuickTest tutorial. For more information, refer to the *Mercury QuickTest Professional Tutorial*.
 - **Flight.** Opens a sample flight reservation Windows application. To access the application, enter any username and the password **mercury**.
- **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 (tests only). For more information, see “Inserting Encoded Passwords into Method Arguments and Data Table Cells” on page 132.
 - **Remote Agent.** Activates the QuickTest Remote Agent, which determines how QuickTest behaves when a test is run by a remote application such as Quality Center. For more information, see “Setting QuickTest Remote Agent Preferences” on page 1297.
 - **Silent Test Runner.** (Relevant only for tests) Opens the Silent Test Runner dialog box, which enables you to run a QuickTest test the way it is run from LoadRunner and Business Availability Center. For more information, see “Using Silent Test Runner” on page 1323.

- **Test Batch Runner.** (Relevant only for tests) Opens the Test Batch Runner dialog box, which enables you to set up QuickTest to run several tests in succession. For more information, see “Running a Test Batch” on page 627.
- **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 689.
- **Action Conversion Tool.** Enables you to convert test actions that were created using QuickTest Professional to scripted components for use in business process testing. For more information, click the **Help** button in the Action Conversion Tool window.
- **License Validation Utility.** Opens the License Validation utility, which enables you to retrieve and validate license information. For more information, see “Validating QuickTest Licenses” on page 917.
- **Register New Browser Control.** Opens the Register Browser Control Utility, which enables you to register your browser control application so that QuickTest Professional recognizes your Web object when recording or running tests. For more information, see “Registering Browser Controls” on page 852.
- **Business Component Upgrade Tool.** Opens the Business Component Upgrade Tool, if you are connected to a Quality Center project. This tool enables you to upgrade in a batch all the QuickTest business components in a version 9.0 or later Quality Center project from an earlier version to the current format. For more information, click the **Help** button in the Business Component Upgrade Tool window.
- **QuickTest Script Editor.** Opens the QuickTest Script Editor, which enables you to open and modify the scripts of multiple tests and function libraries, simultaneously. For more information, see “Working with the QuickTest Script Editor” on page 1083.
- **Check for Updates.** Checks online for any available updates to QuickTest Professional. You can choose which updates you want to download and (optionally) install. For more information, see to “Updating the QuickTest Software” on page 15.

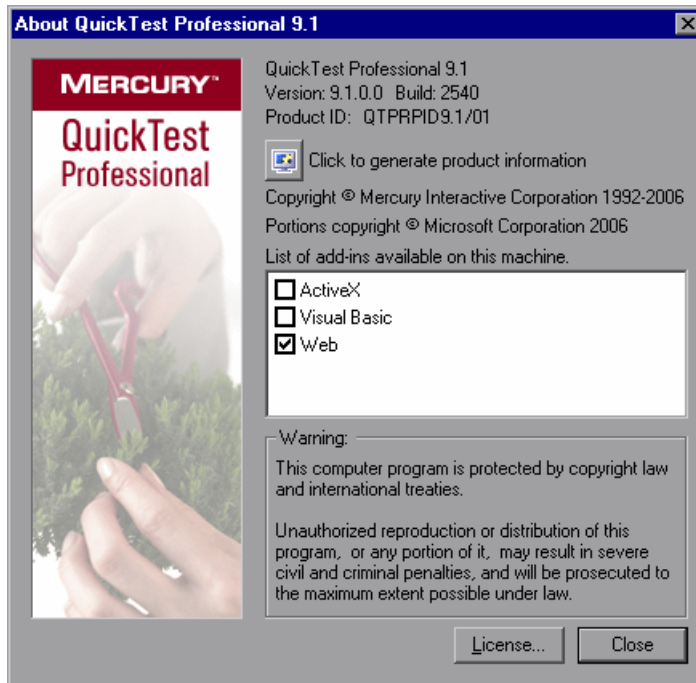
- **QuickTest Professional.** Opens the QuickTest Professional application.
- **Readme.** Opens the QuickTest Professional Readme, which provides the latest news and information on QuickTest Professional.
- **Test Results Viewer.** Opens the Test Results window, which enables you to select a test and view information about the steps performed during the run session. For more information, see “Understanding the Test Results Window” on page 634.
- **Uninstall QuickTest Professional.** Uninstalls QuickTest Professional and all of its components, including built-in and external add-ins. Refer to the *Mercury QuickTest Professional Installation Guide* for more information.

Viewing Product Information

You can view information regarding the QuickTest add-ins and patches installed on your computer, as well as about your operating system. This information is useful for troubleshooting and when dealing with Mercury Customer Support.

To view the product information:

- 1 In QuickTest, choose **Help > About QuickTest Professional**. The About QuickTest Professional 9.1 window opens.



The About QuickTest Professional 9.1 window displays the following information:

- The version of QuickTest that is installed on your computer, its build number, and Product ID number.
- The list of QuickTest add-ins that are installed on your computer. A check mark next to the add-in name indicates that the add-in is currently loaded. For more information on QuickTest add-ins, see Chapter 28, “Working with QuickTest Add-Ins.”

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



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

Product Information	
Product name:	QuickTest Professional
Product version:	9.1
Product ID:	QTPRPID9.1/01
Product build:	2558
Operating system:	Microsoft Windows 2000 Service Pack 4 (Build 2195)
Internet Explorer version:	6.0.2800.1106
Quality Center connectivity:	9.1.0.9999
Add-in Information:	
Name	Version
ActiveX	9.1.0.0
Visual Basic	9.1.0.0
Web	9.1.0.0
Patch Information:	
Name	Readme
Copyright © Mercury Interactive Corporation 1992-2006	
MERCURY™	

The Product Information window displays the following information:

- The QuickTest Professional version, product ID, and build numbers installed on your computer.
- **Operating system.** The operating system version installed on your computer.
- **Internet Explorer version.** The version of Microsoft Internet Explorer installed on your computer.

- **Quality Center connectivity.** The version of the Quality Center connectivity add-in installed on your computer.
- **Add-in Information.** The QuickTest add-ins installed on your computer, and their version and build numbers.
- **Patch Information.** The names of any QuickTest patches installed on your computer, and links to their readme files.

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 or use test objects and methods in your functions.

This chapter describes:	On page:
About Understanding the Test Object Model	61
Applying the Test Object Model Concept	65
Viewing Object Properties Using the Object Spy	70
Viewing Object Methods and Method Syntax Using the Object Spy	73

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.

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

Understanding How QuickTest Learns Objects While Recording

QuickTest learns objects just as you would. For example, suppose as part of an experiment, Johnny 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. Johnny 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, Johnny 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 Johnny and points out one of three children sitting on a picnic blanket. Johnny 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, Johnny 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 Johnny 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 Johnny 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 Johnny 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 Johnny 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, Johnny 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 Johnny 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 Johnny's experiment. In this phase, the tester shows Johnny 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. Johnny 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, Johnny 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. Johnny 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 maintains your test'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 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 to represent the actual object in your application. QuickTest stores information on the object that will help it identify and check the object during the run session.

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

When 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 **Search** button with the following HTML source code:

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

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

Name	Value
Description properties	
type	submit
name	Search
html tag	INPUT

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		
▼ Search Results: Search		
▼ Search Results: Search		
Search	Click	Click the "Search" button.

QuickTest displays your step in the Expert View as follows:

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

When you run a test, QuickTest identifies each object in your application by its test object class and its **description** (the set of 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 **Search** to look up its description. Based on the description it finds, QuickTest would then look for a WebButton object in the application with the HTML tag **INPUT**, of type **submit**, with the value **Search**. When it finds the object, it performs the **Click** method on it.

Understanding Test Object Descriptions

For each 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.

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.

Object Properties

Name: Sign-In

Class: Image

Test object details

Name	Value
Description properties	
image type	Image Button
html tag	INPUT
alt	Sign-In

test object name

test object class

default properties

Item	Operation	Value	Documentation
Action1			
Welcome: Mercury Tours			
Welcome: Mercury Tours			
userName	Set	"mercury"	Enter "mercury" in the "userName" edit box
password	SetSecure	"4082986e39ea46...	Enter the encrypted string "4082986e39ea46...
Sign-In	Click	2.2	Click the "Sign-In" image.

image icon

test object name

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

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

You can configure the mandatory, assistive, and ordinal identifier properties that QuickTest uses to 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 (for example, Microsoft for Microsoft Internet Explorer objects, Netscape for Netscape Browser 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 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 on activating run-time methods using the **Object** property, see “Retrieving and Setting Test Object Property Values” on page 1027.

Each test object method you perform while recording is recorded as a separate step in your test. When you run your test, 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. 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 use **SetTOProperty** statements during a run session. You can also use regular expressions to identify property values based on conditions or patterns you define, or you can parameterize property values with Data Table parameters so that a different value is used during each iteration of the test. For more information on modifying object properties, see Chapter 6, “Working with Test Objects.” For more information on parameterization, see Chapter 16, “Parameterizing Values.” For more information on regular expressions, see “Understanding and Using Regular Expressions” on page 352.

You can view or modify the test object property values that are stored with your test 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 on the Object Properties and Object Repository dialog boxes, see “Modifying Test Object Properties” on page 168. For information on viewing test object property values using the Object Spy, see “Viewing Object Properties Using the Object Spy” on page 70.

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 73.

You can retrieve or modify property values of the test object during the run session by adding **GetTOProperty** and **SetTOProperty** statements in the Keyword View or Expert View. You can retrieve property values 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 1027.



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 1029.

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

Viewing Object Properties Using the Object Spy

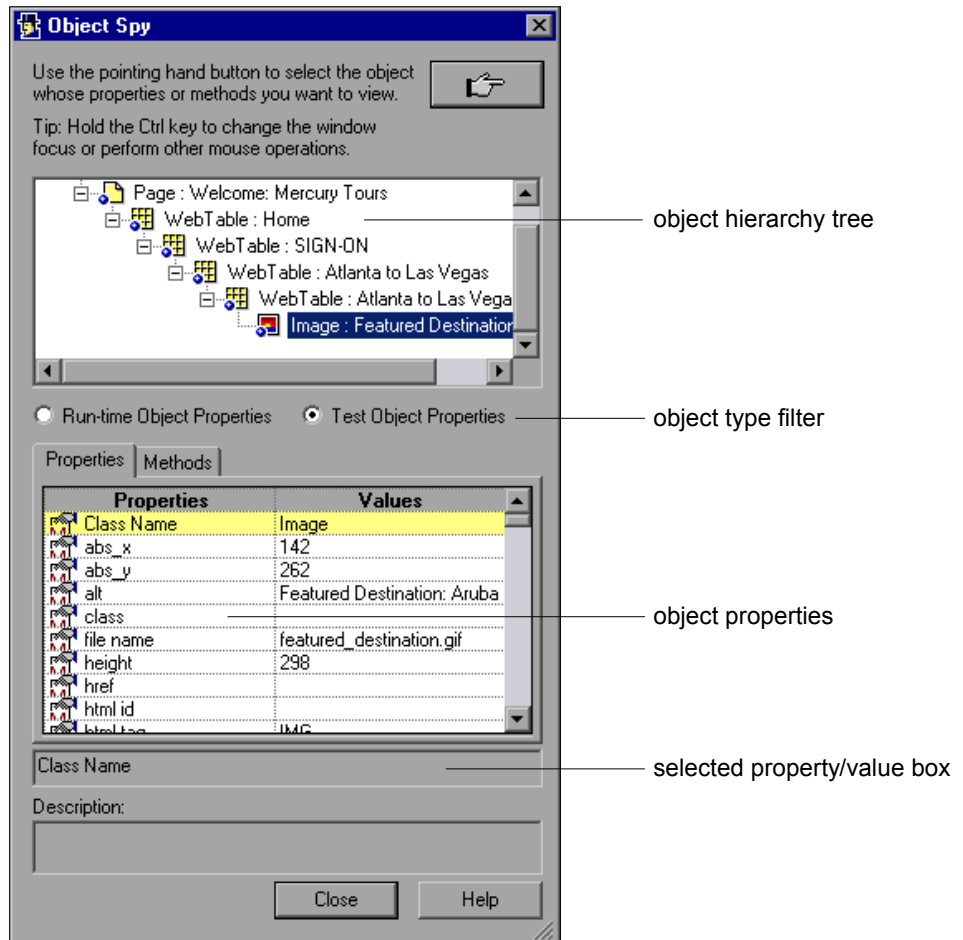
Using the Object Spy, you can view the properties of any object in an open application. You use the Object Spy pointer to point to an object. The Object Spy displays the selected object's hierarchy tree and its properties and values in the Properties tab of the Object Spy dialog box.

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 Window” on page 156.
-  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 25, “Setting Global Testing Options.” You can also hold the left CTRL key to change the window focus. Additionally, if the window containing the object you want to select is minimized, you can display it by holding the left CTRL key, right-clicking the application in the Windows task bar, and choosing **Restore** from the context menu.

- 4 If the object on which you want to spy can be displayed only by performing an event (such as right-clicking or moving the pointer over an object to display a context menu), hold the left CTRL key. The pointing hand temporarily turns into a standard pointer and you can perform the event. When the object on which you want to spy is displayed, release the left CTRL key. The pointer becomes a pointing hand again.
- 5 Click the object whose properties you want to view. The Object Spy returns to focus and displays the object hierarchy tree and the properties of the object that is selected within the tree.



Tip: You can resize the Object Spy dialog box. This is useful if you have a deep hierarchy, or long property names and values, enabling you view all the information without scrolling.

- 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.
-

Tips:

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 1029.



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 run session. For more information, see “Retrieving and Setting Test Object Property Values” on page 1027.

- 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.

Viewing Object Methods and Method Syntax Using the Object Spy

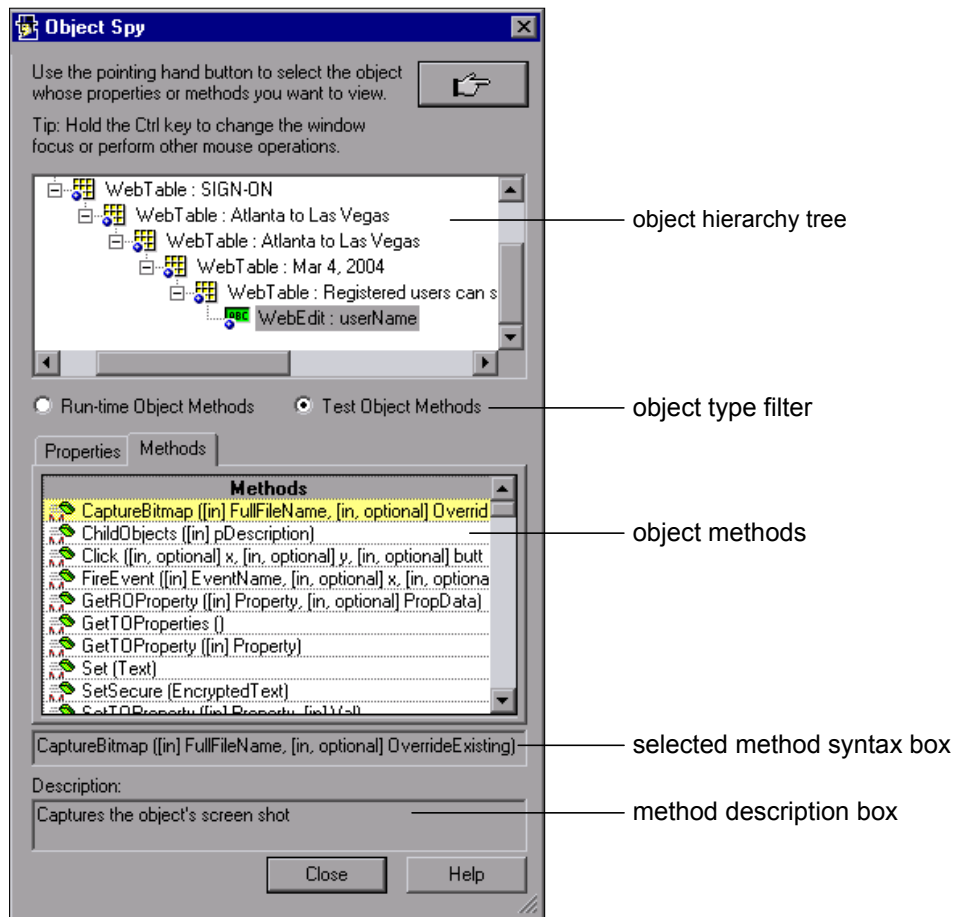
In addition to viewing object properties, the Object Spy also enables you to view both the run-time object methods and the test object methods associated with an object and to view the syntax for a selected method. You use the Object Spy pointer to point to an object. The Object Spy displays the object 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 Window” on page 156.
- 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 25, “Setting Global Testing Options.” You can also hold the left CTRL key to change the window focus. Additionally, if the window containing the object you want to select is minimized, you can display it by holding the left CTRL key, right-clicking the application in the Windows task bar, and choosing **Restore** from the context menu.

- 5 If the object on which you want to spy can be displayed only by performing an event (such as right-clicking or moving the pointer over an object to display a context menu), hold the left CTRL key. The pointing hand temporarily turns into a standard pointer and you can perform the event. When the object on which you want to spy is displayed, release the left CTRL key. The pointer becomes a pointing hand again.
- 6 Click the object whose associated methods you want to view. 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 in the tree.



Tip: You can resize the Object Spy dialog box. This is useful if you have a deep hierarchy, or long syntax, enabling you view all the information without scrolling.

- 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 1030.

- 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.

Part II

Creating Tests

4

Designing Tests

You can create a test by recording the operations you perform on your application or using the keyword-driven methodology, which enables you to select keywords to indicate the operations you want to perform on your application. After you create your test, you can enhance it using checkpoints and other special testing options.

This chapter describes:	On page:
About Designing Tests	80
Planning and Preparing to Create a Test	81
Creating a Test Using the Keyword-Driven Methodology	82
Recording a Test	88
Understanding Your Recorded Test	92
Choosing the Recording Mode	93
Enhancing Your Test	101
Managing Your Test	103

About Designing Tests

You can design tests using the keyword-driven methodology, step recording, or a combination of both.

In comparison to recording, the keyword-driven methodology generally requires a longer time-investment and at least one automation expert with a high-level of QuickTest expertise to prepare the infrastructure required for your tests, but once the infrastructure is in place, tests can be created at a more application-specific level and with a more structured design. Additionally, all the tests for a given application can be maintained more efficiently and with more flexibility than a recorded test.

In some cases, you may want to let QuickTest generate test steps by recording the typical processes that you perform on your application. As you navigate through your application, QuickTest graphically displays each step you perform as a row in the Keyword View. A step is anything a user does that changes the content of a page or object in your application, for example, clicking a link or typing data into an edit box. Recording may be easier for new QuickTest users or when beginning to design tests for a new application or a new feature.

While creating your test, you can insert checkpoints into your test. A **checkpoint** compares the value of an element captured when the object was saved in the object repository, with the value of the same element captured during the run session. This helps you determine whether or not your application 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 16, “Parameterizing Values.”

After creating your initial test, you can further enhance it by adding and modifying steps in the Keyword View or Expert View.

Planning and Preparing to Create a Test

Before you start creating your test, you should plan it and prepare the required infrastructure. 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 ones 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 8, “Understanding Checkpoints.”
- Consider changing the way that QuickTest identifies specific 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 more information, see Chapter 33, “Configuring Object Identification.”
- Decide how you want to organize your object repositories. For individual tests, you can work with the individual action’s object repositories, or you can use a common (shared) object repository for multiple tests. If you are new to testing, you may want to keep the default object repository per-action setting for tests. Once you feel more comfortable with the basics of test design, you may want to take advantage of the shared object repository option. For more information, see Chapter 6, “Working with Test Objects.”
- Find out whether an object repository containing the objects you are testing already exists. If not, create a new object repository or add objects to an existing one. For more information, see Chapter 38, “Managing Object Repositories.”
- Determine whether you need to create any new user-defined functions or whether you should associate any existing function libraries with your test. For more information, see Chapter 35, “Working with User-Defined Functions and Function Libraries.”
- If you are recording steps, 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 41, “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 16, “Parameterizing Values.”
- Consider using actions to streamline the testing process. For more information, see Chapter 18, “Working with Actions.”
- If you have useful WinRunner assets, you may want to link to WinRunner tests and call WinRunner TSL functions from your QuickTest test. For more information, see Chapter 44, “Working with WinRunner.”

Creating a Test Using the Keyword-Driven Methodology

Keyword-driven testing is a technique that separates much of the programming work from the actual test steps so that the test steps can be developed earlier and can often be maintained with only minor updates, even when the application or testing needs change significantly.

The keyword-driven testing methodology divides test creation into two stages:

- Preparing a set of testing resources (the test automation infrastructure). Preparing the test automation infrastructure includes a planning stage and an implementation stage.
- Creating tests in the QuickTest Keyword View by selecting the keywords (objects and/or operations) that represent the application functionality you want to test. This is described in Chapter 5, “Working with the Keyword View.”

Planning Stage

- Analyzing the application and determining which objects and operations are used by the set of business processes that need to be tested.
- Determining which operations require customized keywords to provide additional functionality, to achieve business-level clarity, and/or to maximize efficiency and maintainability.

Implementation Stage

- Building the object repository and ensuring that all objects have clear names that follow any predetermined naming conventions. You can create object repositories using QuickTest functionality to recognize and learn the objects in your application, or you can manually define objects. The object repository should contain all the objects that are relevant for the tests using this infrastructure.
- Developing and documenting business-level keywords in function libraries. Creating function libraries involves developing customized functions for the application you want to test. You may want to develop functions to test special application functionality that is not already supplied by the methods in the QuickTest object model. It is also useful to wrap existing methods and functions together with additional programming to create application-specific functions for testing operations or sequences that are commonly performed in your application. The functions you create will be available either as extra keywords or as replacements for built-in QuickTest keywords during the test creation stage.

Although this methodology requires more planning and a longer initial time-investment than going directly to the test creation stage and recording your steps, this methodology makes the test creation and test maintenance stages more efficient and keeps the structure of individual tests more readable and easier to modify.

By encapsulating much of the complex programming into function libraries, and by making these functions flexible enough to use in many testing scenarios (through the use of function parameters that control the way the functions behave), one or a few automation experts can prepare the keywords that many application testers who are less technical can include in multiple tests. This also makes it possible to update testing functionality without having to update all the tests that use the keywords.

In a similar fashion, by maintaining all objects that are relevant to a particular application or area of an application within one shared object repository, and by associating that object repository with all relevant tests, changes to the application can be reflected in the object repository without the need to update tests.

After the object and function library keywords are ready, application testers can use these keywords to build keyword-driven tests.

Recorded tests tend to be specific to a single procedure or set of procedures. When the application changes, especially when it changes significantly from one version to another, much of the test often needs to be rerecorded.

By creating your tests with a keyword-driven methodology in mind, your tests become more modular, focusing on the operations to test using both QuickTest built-in keywords and your own user-defined keywords. Additionally, because it is possible to add objects to the object repository before they exist in an application, it is possible to begin preparing your automated keyword-driven tests even before a build containing the new objects is available.

Developing a Keyword-Driven Automation Infrastructure

One or a few automation experts usually develop the test automation infrastructure that all tests related to a certain application or functionality will use. The automation infrastructure usually includes one or more shared object repositories and one or more function libraries.

The information in the sections below gives you some guidance on the main tasks involved in creating these resources and describes where you can find detailed documentation for these tasks.

After the test automation infrastructure is ready, the application testers can begin designing their keyword-driven tests by selecting objects and operation keywords in the Keyword View. For more information, see Chapter 5, “Working with the Keyword View.”

Creating a Shared Object Repository

The tasks involved in creating a shared object repository for the test automation infrastructure can include:

- Adding (learning) objects from your application. For more information, see “Adding Objects to the Object Repository” on page 189.

- Creating new objects to represent objects that do not yet exist in your application and then later updating the properties and values of these objects as necessary once they exist in the application. For more information, see “Defining New Test Objects” on page 200.
- Ensuring that objects in the object repository have names that are easy for application testers to recognize and that follow any established object naming guidelines.
- Copying or moving objects from one repository to another. For more information, see Chapter 38, “Managing Object Repositories.”
- Merging objects added to local repositories by application testers into the shared object repositories of the automation infrastructure, and merging two or more existing repositories. For more information, see Chapter 39, “Merging Shared Object Repositories.”

Creating a Function Library

The tasks involved in creating a function library for the test automation infrastructure can include:

- Determining which keywords are needed.
- Creating and editing function library files in the QuickTest Function Library window. For more information, see “Creating a Function Library” on page 1041.
- Creating the actual functions within the function libraries. You can do this manually, or you can use the Function Definition Generator to generate function definitions and header information. For more information, see “Using the Function Definition Generator” on page 1056.
- Defining functions as new or replacement methods for test objects. For more information, see “Registering User-Defined Functions as Test Object Methods” on page 1072.
- Debugging the function libraries. For more information, see “Debugging a Function Library” on page 1049.

Note: Although not directly associated with the keyword-driven methodology, the automation experts who maintain the object repositories and function libraries also often maintain a set of recovery scenarios that all application testers can associate with their tests. For more information, see Chapter 32, “Defining and Using Recovery Scenarios.”

Deciding When to Use the Keyword-Driven Methodology

You can create the steps in your tests using the keyword-driven methodology, recording, or a combination of both.

Recording can be useful in the following situations:

- Recording is useful for novice QuickTest users. It helps you to learn how QuickTest interprets the operations you perform on your application and how it converts them to QuickTest objects and built-in operations.
- For the same reasons as described above, recording is also useful for more advanced QuickTest users when working with a new application or major new features of an existing application. It is also helpful while developing functions that incorporate built-in QuickTest keywords.
- Recording can be useful when you need to quickly create a test that tests the basic functionality of an application or feature and does not require long-term maintenance.

Keyword-driven testing advantages include the following:

- Keyword-driven testing enables you to design your tests at a business level rather than at the object level. For example, QuickTest may recognize a single option selection in your application as several steps: a click on a button object, a mouse operation on a list object, and then a keyboard operation on a list sub-item. You can create an appropriately-named function to represent all of these lower-level operations in a single, business-level keyword.

- Similarly, by incorporating technical operations, such as a synchronization statement that waits for client-server communications to finish, into higher level keywords, tests are easier to read and easier for less technical application testers to maintain when the application changes.
- Keyword-driven testing naturally leads to a more efficient separation between resource maintenance and test maintenance. This enables the automation experts to focus on maintaining objects and functions while application testers focus on maintaining the test structure and design.
- When you record tests, you may not notice that new objects are being added to the local object repository. This may result in many testers maintaining local object repositories with copies of the same objects. When using a keyword-driven methodology, you select the objects for your steps from the existing object repository. When you need a new object, you can add it to your local object repository temporarily, but you are also aware that you need to add it to the shared object repository for future use.
- When you record a test, QuickTest enters the correct objects, methods, and argument values for you. Therefore, it is possible to create a test with little preparation or planning. Although this makes it easier to create tests quickly, such tests are harder to maintain when the application changes and often require re-recording large parts of the test.

When you use a keyword-driven methodology, you select from existing objects and operation keywords. Therefore, you must be familiar with both the object repositories and the function libraries that are available. You must also have a good idea of what you want your test to look like before you begin inserting steps. This usually results in well-planned and better-structured tests, which also results in easier long-term maintenance.

- Automation experts can add objects and functions based on detailed product specifications even before a feature has been added to a product. Using keyword-driven testing, you can begin to develop tests for a new product or feature earlier in the development cycle.

Recording a Test

You can create the main body of a test by recording the typical processes that users perform. QuickTest records the operations you perform, displays them as steps in the Keyword View, and generates them in a script (in the Expert View).

Note that by default, each test includes a single action, but can include multiple actions. This chapter describes how to record a test with a single action. For information on why and how to work with multiple actions, see Chapter 18, “Working with Actions.”

By default, QuickTest records in the normal recording mode. If you are unable to record on an object in a given environment in the standard recording mode, or if you want to record mouse clicks and keyboard input with the exact x- and y-coordinates, you may want to record on those objects using Analog or Low Level Recording. For more information on Low Level and Analog recording, see “Choosing the Recording Mode” on page 93.

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 31, “Learning Virtual Objects.”

Consider the following guidelines when recording a test:

- Before you start to record, close all applications not required for the recording session.
- If you are recording on a Web site, determine the security zone of the site. When you record on a Web browser, the browser may prompt you with security alert dialog boxes. You may choose to disable/enable these dialog boxes.
- Decide how you want to open the application 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 27, “Setting Record and Run Options.”

- Choose how you want QuickTest to record and run your test by setting global testing options in the Options dialog box and settings specific to your test in the Test Settings dialog box. For more information, see Chapter 25, “Setting Global Testing Options” and Chapter 26, “Setting Options for Individual Tests.”
- If you are recording on a Web object, you must make a change to the object’s value to make QuickTest record the step. For example, to record a selection in a WebList object, you must click on the list, scroll to an entry that was not originally showing, and select it. If you want to select the item in the list that is already displayed, you must first select another item in the list (click it), then return to the originally displayed item and select it (click it).

Note: If you are creating a test on Web objects, you can record your test on Microsoft Internet Explorer and run it on another supported browser (according to the guidelines specified in the *QuickTest Professional Readme*). QuickTest supports running tests on the following browsers—Microsoft Internet Explorer, Netscape Browser, Mozilla Firefox, and applications with embedded Web browser controls. For more information, see Chapter 29, “Testing Web Objects.”

To record a test:

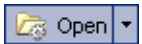


1 Open QuickTest. For more information, see “Starting QuickTest” on page 18.

2 Open a test:



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

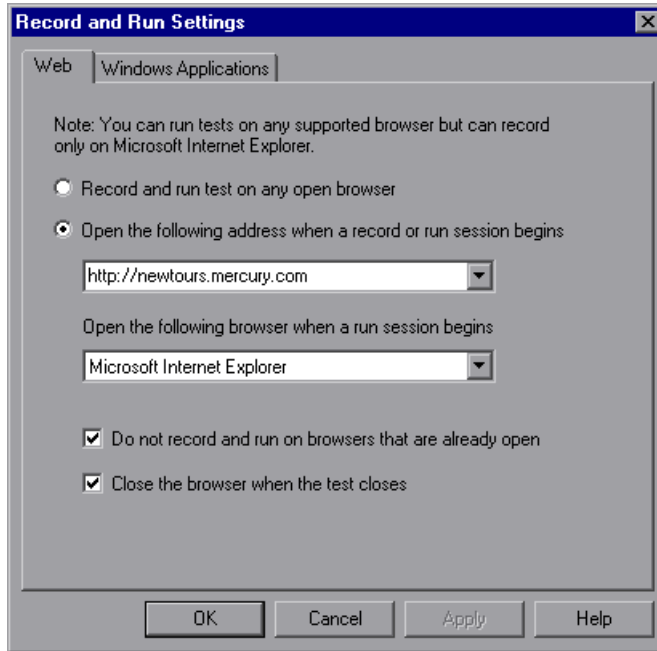


➤ To open an existing test, click the **Open** button or choose **File > Open > Test**. In the Open QuickTest Test dialog box, select a test and click **Open**.

For more information, see “Managing Your Test” on page 103.



- 3 Click the **Record** button or choose **Automation > Record**. If you are recording a new test and have not yet set your record and run settings in the Record and Run Settings dialog box (from **Automation > Record and Run Settings**), the Record and Run Settings dialog box opens.





After you 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 **Automation > Record and Run Settings** to open the Record and Run Settings dialog box. You can use this option to set or modify your record and run preferences in the following scenarios:

- You have already recorded one or more steps in the test and you want to modify the settings before you continue recording.
- You want to run the test on a different application 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 external add-ins, where applicable. The image here includes the environments listed below:

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

- 4** Set an option, as described in Chapter 27, “Setting Record and Run Options.”
- 5** To apply your changes and keep the Record and Run Settings dialog box open, click **Apply**.
- 6** Click **OK** to close the Record and Run Settings dialog box and begin recording your test.
- 7** Navigate through your application 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 application is functioning correctly, you can insert text checkpoints, object checkpoints, and bitmap checkpoints. For more information, see Chapter 8, “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 16, “Parameterizing Values.”
-  **10** When you complete your recording session, click the **Stop** button or choose **Automation > 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 103.





Understanding Your Recorded Test



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			
userName	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.
 userName 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.

Step	Description
 password SetSecure "4082986e39ea469e70dbf8c5a29429fe138c6efc"	password is the name of the edit box. SetSecure is an encryption method performed on the edit box. 4082986e39ea469e70dbf8c5a29429fe138c6efc is the encrypted value of the password.
 Sign-In 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
    "4082986e39ea469e70dbf8c5a29429fe138c6efc"
Browser("Welcome: Mercury Tours").Page("Welcome: Mercury Tours").
    Image("Sign-In").Click 2,2
```

Choosing the Recording Mode

Normal recording mode records the objects in your application and the operations performed on them. This mode is the default and takes full advantage of the QuickTest test object model, recognizing the objects in your application regardless of their location on the screen.

When working with specific types of objects or operations, however, you may want to choose from the following, alternative recording modes:

- **Analog Recording.** enables you to record the exact mouse and keyboard operations you perform in relation to either the screen or the application window. In this recording mode, QuickTest records and tracks every movement of the mouse as you drag the mouse around a screen or window.

This mode is useful for recording operations that cannot be recorded at the level of an object, for example, recording a signature produced by dragging the mouse.

Note: You cannot edit Analog Recording steps from within QuickTest.

- **Low Level Recording.** enables you to record on any object in your application, whether or not QuickTest recognizes the specific object or the specific operation. This mode records at the object level and records all run-time objects as Window or WinObject test objects. Use Low Level Recording for recording in an environment or on an object not recognized by QuickTest. You can also use Low Level Recording if the exact coordinates of the object are important for your test.

Note: Steps recorded using Low Level Recording mode may not run correctly on all objects.

Guidelines for Analog and Low Level Recording

Consider the following guidelines when choosing Analog Recording or Low Level Recording:

- Use Analog Recording or Low Level Recording only when normal recording mode does not accurately record your operation.
- Analog Recording and Low Level Recording require more disk space than normal recording mode.
- You can switch to either Analog Recording or Low Level Recording in the middle of a recording session for specific steps. After you record the necessary steps 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 run session, QuickTest will accurately identify the window location on which the analog steps were performed even if the window is in a different location when you run the analog steps. QuickTest does not record any click or mouse movement performed outside the specified window. When using this mode, QuickTest does not capture any Active Screen images.
 - **Record relative to the screen** if the window on which you are recording your analog steps moves during recording or if the operations you perform are on objects located within more than one window. This can include dragging and dropping an object from one window to another. When using this mode, QuickTest captures only the Active Screen image of the final state of the window on which you are recording.
- The steps recorded using Analog Recording are saved in a separate data file. This file is stored with the action in which the analog steps are recorded.
- When you record in Analog Recording mode, QuickTest adds to your test a **RunAnalog** statement that calls the recorded analog file. The corresponding Active Screen displays the results of the last analog step that was performed during the Analog Recording session.

Low Level Recording

- Use Low Level Recording for recording on environments or objects not supported by QuickTest.
- Use Low Level Recording for when you need to record the exact location of the operation on your application screen. While recording in normal mode, QuickTest performs the step on an object even if it has moved to a new location on the screen. If the location of the object is important to your test, switch to Low Level Recording to enable QuickTest to record the object in terms of its x- and y- coordinates on the screen. This way, the step will pass only if the object is in the correct position.
- While Low Level Recording, QuickTest records all parent level objects as Window test objects and all other objects as WinObject test objects. They are displayed in the Active Screen as standard Windows objects.
- Low Level Recording supports the following methods for each test object:
 - WinObject test objects: **Click, DblClick, Drag, Drop, Type**
 - Window test objects: **Click, DblClick, Drag, Drop, Type, Activate, Minimize, Restore, Maximize**
- Each step recorded in Low Level Recording mode is shown in the Keyword View and Expert View. (Analog Recording records only the one step that calls the external analog data file.)

Using Analog Recording

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

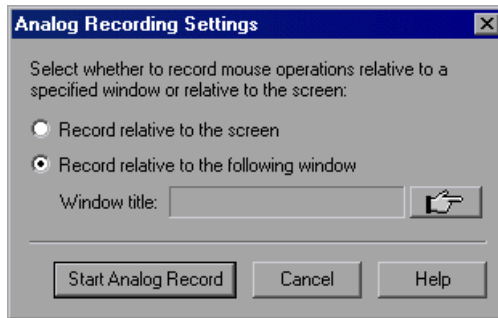
To record in Analog Recording mode:



- 1 If you are not already recording, click the **Record** button to begin a recording session.



- 2 Click the **Analog Recording** button or choose **Automation > Analog Recording**. The Analog Recording Settings dialog box opens.



- 3 Select from the following options:

- **Record relative to the screen.** QuickTest records any mouse movement or keyboard input relative to the coordinates of your screen, regardless of which application(s) are open or which application(s) you specified in the Record and Run Settings dialog box.

Select **Record relative to the screen** if you perform your analog operations on objects located within more than one window or if the window itself may move while you are recording your analog operations.

Note: When you record in Analog mode relative to the screen, the run session will fail if your screen resolution or the screen location on which you recorded your analog steps has changed from the time you recorded.

The analog tracking continues to record the movement of the mouse until the mouse reaches the QuickTest screen to turn off Analog Recording or to stop recording. Clicking on the QuickTest icon in the Windows taskbar is also recorded. This should not affect your test. The mouse movements and clicks on the QuickTest screen itself are not recorded.

- **Record relative to the following window.** QuickTest records any mouse movement or keyboard input relative to the coordinates of the specified window.

Select **Record relative to the following window** if all your operations are performed on objects within the same window and that window does not move during Analog Recording. This guarantees that the test will run the analog steps in the correct position within the window even if the window's screen location changes after recording.

Note: If you have selected to record in Analog Recording mode relative to window, any operation performed outside the specified window is not recorded while in Analog Recording mode.

- 4** If you choose to **Record relative to the following window**, click the pointing hand and click anywhere in the window on which you want to record in Analog Recording mode. The title of the window you clicked is displayed in the window title box.

Tip: You can also hold the left CTRL key to change the window focus or perform operations such as right-clicking or moving the pointer over an object to display a context menu from which you can select the required window. Additionally, if the window containing the object you want to select is minimized, you can display it by holding the left CTRL key, right-clicking the application in the Windows task bar, and choosing **Restore** from the context menu.

5 Click Start Analog Record.


6 Perform the operations you want to record in Analog Recording mode.

All of your keyboard input, mouse movements, and clicks are recorded and saved in an external file. When QuickTest runs the test, the external data file is called. It tracks every movement and click of the mouse to replicate exactly the operations you recorded.




7 When you are finished and want to return to normal recording mode, click the Analog Recording button or choose Automation > Analog Recording to turn off the option.

If you chose to **Record relative to the screen**, QuickTest inserts the **RunAnalog** step for a **Desktop** item. For example:

Item	Operation	Value
 Desktop	RunAnalog	"Track1"

Desktop.RunAnalog "Track1"

If you chose to **Record relative to the following window**, QuickTest inserts the **RunAnalog** step for a **Window** item. For example:

Item	Operation	Value
 Microsoft Internet Explorer	RunAnalog	"Track1"

Window("Microsoft Internet Explorer").RunAnalog "Track1"

The track file called by the **RunAnalog** method contains all your analog data and is stored with the current action.

You can use this track file in more than one action in your test, and also in other tests, by saving the action containing the **RunAnalog** step as a reusable action. A reusable action can be called by other tests or actions. For more information on using actions, see Chapter 18, “Working with Actions” and Chapter 30, “Working with Advanced Action Features.”

Note: When entering the **RunAnalog** method, you must use a valid and existing track file as the method argument.

Tip: To stop an analog step in the middle of a run session, 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 **Automation > Low Level Recording**.

The record mode changes to Low Level Recording and all of your keyboard input and mouse clicks are recorded based on mouse coordinates. When QuickTest runs the test, the cursor retraces the recorded clicks.



- 3 When you are finished and want to return to normal recording mode, click the **Low Level Recording** button or choose **Automation > Low Level Recording** to turn off the option.

The following examples illustrate the difference between the same operations recorded using normal mode and Low Level Recording mode.

Suppose you type the word `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:

▼ 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
```

Enhancing Your Test

After you create a test, you can use a variety of options to enhance it.

You can add checkpoints to your test. A **checkpoint** is a step in your test that compares the a specified item during a run session with the values stored for the same item within the test. This enables you to identify whether or not your application is functioning correctly. There are several different checkpoint types. For more information on creating checkpoints, see Chapter 8, “Understanding Checkpoints.”

Tip: You can also use the **CheckProperty** method, which enables you to verify the property value of an object without using the checkpoint interface.

You can parameterize your test to replace fixed values with values from an external source during your run session. The values can come from a Data Table, environment variables you define, or values that QuickTest generates during the run session. For more information, see Chapter 16, “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 17, “Outputting Values.”

You can divide your test into actions to streamline the testing process of your application. For more information, see Chapter 18, “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 21, “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 34, “Working in the Expert View and Function Library Windows.”

Managing Your Test

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

Tip: As the content of your application or Web site changes, you can update the selected Active Screen display and use the Active Screen to add new steps to your test instead of re-recording steps on new or modified objects. For more information, see “Changing the Active Screen” on page 223.

Creating a New Test



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

Opening an Existing Test

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

Note: You cannot open a test that was created with a later version of QuickTest on a computer running an earlier version of QuickTest. For example, you cannot open a test created in QuickTest 9.1 on a computer running QuickTest 8.0.

Notes for users of previous QuickTest versions:

When you open a test that was created using a version of QuickTest earlier than version 9.1, you are asked whether you want to convert it or view it in read-only format. If the test contains objects from external add-ins in its actions' local object repositories, the relevant add-in must be installed to convert the test to the current format. Otherwise, it is opened in read-only format.

If you choose to view the test in read-only format, it appears as it did previously, using all of its original settings, but you cannot modify it. If you choose to convert the test, it is updated to the current format and you can modify it as needed. If you save the converted test, it cannot be used with earlier versions of QuickTest.

To open an existing test:



- 1 If your test is not already open, choose **File > Open > Test**, or click the **Open** arrow and choose **Test**, or press CTRL+O.
- 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 45, "Working with Quality Center."

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

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 and any associated shared object repositories.

Tip: If changes are made to an existing test, an asterisk (*) is displayed in the title bar until the test is saved.

Note: You must use the **Save As** option in QuickTest if you want to save a test under another name or create a copy of a test. You cannot copy a test or change its name directly in the file system or in Quality Center.

To save a new test:



- 1 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.

Note: QuickTest suggests a default folder called **Tests**. For all supported operating systems except Windows Vista, this folder is located under your QuickTest Professional installation folder. For Windows Vista, this folder is located under MyDocuments\Mercury Interactive\QuickTest Professional.

- 3 Type a name for the test in the **File name** box. Note that the test name cannot exceed 220 characters (including the path), cannot begin or end with a space, and cannot contain the following characters:
\\ : * ? " < > | % ' ;

- 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” on page 616.

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 “Setting Active Screen Options” on page 715.

- 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 45, “Working with Quality Center.”

Zippping 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 Test 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 Test 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

You can print your entire test from the Keyword View (in table format). You can also print a single action either from the Keyword View (in table format) or the Expert View (in statement format). When printing from the Expert View, you can also specify additional information that you want to be included in the printout.

To print from the Keyword View:



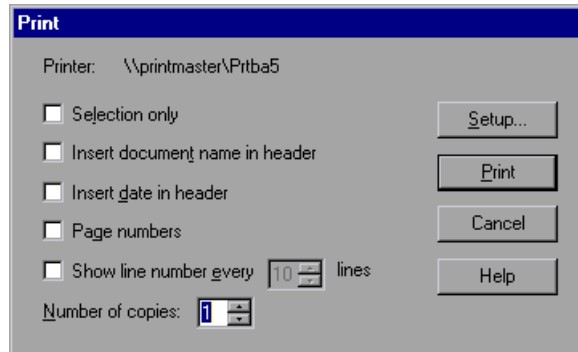
- 1 Click the **Print** button or choose **File > Print**. A standard Print dialog box opens.
- 2 Click **OK** to print the content of the Keyword View to your default Windows printer.

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 which the print job will be sent. You can change the printer by clicking the **Setup** button.
 - **Selection only.** Prints only the text that is currently selected (highlighted) in the Expert View.
 - **Insert document name in header.** Includes the name of the active test or function library at the top of the printout.
 - **Insert date in header.** Includes today's date at the top of the printout. The date format is taken from your Windows regional settings.
 - **Page numbers.** Includes page numbers on the bottom of the printout (for example, page 1 of 3).
 - **Show line numbers every __ lines.** Displays line numbers to the left of the script lines, as specified.
 - **Number of copies.** Specifies the number of times to print the document.
- 3 If you want to print to a different printer, or change your printer preferences, click **Setup** to display the Print Setup dialog box.
- 4 Click **Print** to print according to your selections.

5

Working with the Keyword View

The Keyword View provides an easy way to create, view, and modify tests in a graphical easy-to-use format.

This chapter describes:	On page:
About Working with the Keyword View	112
Understanding the Keyword View	113
Understanding the QuickTest Recorded Object Hierarchy	118
Adding a Standard Step to Your Test	120
Adding Other Types of Steps to Your Test	133
Modifying the Parts of a Step	136
Working with Comments	136
Managing Action Steps	137
Using Keyboard Commands in the Keyword View	140
Defining Keyword View Display Options	141
Viewing Properties of Step Elements in the Keyword View	147
Working with Breakpoints in the Keyword View	148

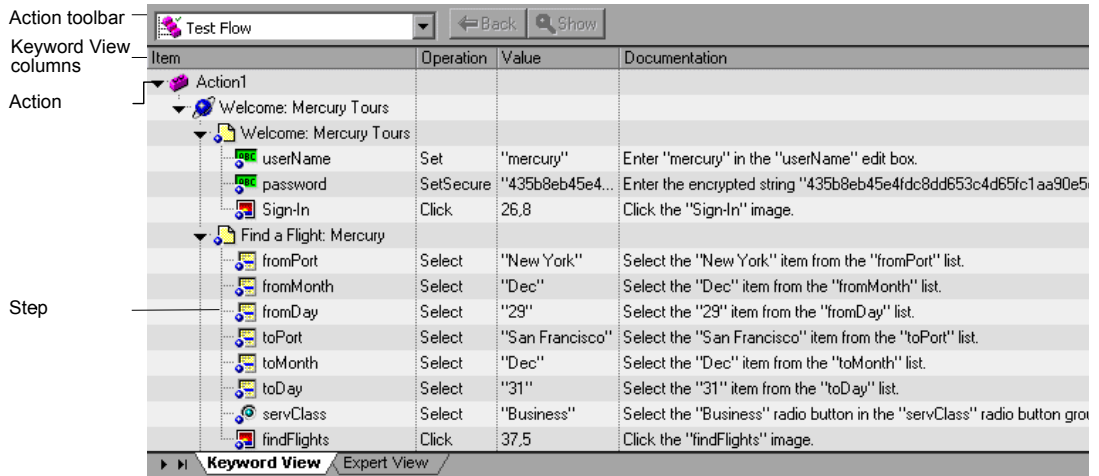
About Working with the Keyword View

The Keyword View enables you to create and view the steps of your test in a modular, table format. Each step is a row in the Keyword View that is comprised of individual, modifiable parts. You create and modify steps by selecting items and operations in the Keyword View and entering information as required. Each step is automatically documented as you complete it, enabling you to view a description of your test in understandable sentences. You can also use these descriptions as instructions for manual testing, if required.

You can use the Keyword View to add new steps to your test and to view and modify existing steps. When you add or modify a step, you select the test object or other step type you want for your step, select the method operation you want to perform, and define any necessary values for the selected operation or statement. Working in the Keyword View does not require any programming knowledge. The programming required to actually perform each test step is done automatically behind the scenes by QuickTest.

Understanding the Keyword View

The Keyword View is comprised of a table-like view, in which each step is a separate row in the table, and each column represents the different parts of the steps. The columns displayed vary according to your selection. For more information, see “Defining Keyword View Display Options” on page 141.






Actions are the highest level of the test hierarchy. They 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 **Action** toolbar.

You can insert a new action, a call to an action, or a copy of an action, to your test. 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. The Action toolbar is available only if at least one of the actions in your test is reusable. For more information on inserting and using actions in the Keyword View, see Chapter 18, “Working with Actions.”

Tip: You can copy and paste or drag and drop actions to move them to a different location within a test. For more information, see “Managing Action Steps” on page 137.

Each action is comprised of steps. During a recording session, each step is recorded as a row in the Keyword View. For example, the Keyword View could contain the following rows:

Welcome: Mercury Tours				
	userName	Set	"mercury"	Enter "mercury" in the "userName" edit box.
	password	SetSecure	"3ee35"	Enter the encrypted string "3ee35" in the "password" edit box.
	Sign-In	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.

You can use the Keyword View to add steps at any point in your test. After you add steps, you can modify or delete them using standard editing commands and drag-and-drop functionality. You can print the contents of the Keyword View to your Windows default printer (and even preview the contents prior to printing). For more information, see “Printing a Test” on page 108.

In the Keyword View, 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, and insert breakpoints to assist you in debugging it.

The Keyword View can contain any of the following columns: **Item**, **Operation**, **Value**, **Assignment**, **Comment**, and **Documentation**. A brief description of each column is provided below.

Item Column

The item on which you want to perform the step (test object, utility object, function call, or statement). This column displays a hierarchical icon-based tree. The highest level of the tree are actions, and all steps are contained within the relevant branch of the tree. Steps performed within the same parent object are displayed under that same object. Function calls, utility objects, and statements are placed in the tree hierarchy at the same level as the item above them (as a sibling).

You can collapse or expand an item in the item tree to change the level of detail that the tree displays.

- To collapse an item and its sub-items, click the arrow ▼ to the left of the item's icon, press the minus key (-) on your keyboard number pad, press the left arrow key on your 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 item's icon) and that a specific column is not selected, before pressing the required key. Otherwise, the keys will not work.

Operation Column

The operation to be performed on the item. This column contains a list of all available operations (methods 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.

Comment Column

A free text edit box for any information you want to add regarding the step. These are also displayed as inline comments in the Expert View.

Note: You can also enter a comment on a new line below the currently selected step by choosing **Insert > Comment**. For more information, see “Adding Comments” on page 578.

Documentation Column

Read-only auto-documentation of what the step does in an easy-to-understand sentence, for example, Click the “Sign-in” image. or Select “San Francisco” in the “toPort” list. If you want to print or view only the steps, you can choose to display only this column. For example, you may want to print or view manual testing instructions.

Tips:

You can display only the **Documentation** column of a test by right-clicking the column header row and choosing **Documentation Only** from the displayed menu.

You can also copy the documentation by selecting **Edit > Copy Documentation to Clipboard**, or right-clicking the column header row and choosing **Copy Documentation to Clipboard** from the displayed menu, and then paste it into a different application, as required.

Note: If you do not see one or more of these columns described below in the Keyword View, you can use the Keyword View Options dialog box to display them. For more information, see “Defining Keyword View Display Options” on page 141.

Tips for Working with the Keyword View

- You can use the left and right arrow keys to move the focus one cell to the left or right, with the following exceptions:
 - In the **Item** column, the left and right arrow keys collapse or expand the item (if possible). If not possible, the arrow keys behave as in any other column.
 - When a cell is in edit mode, for example, when modifying a value or comment, the left and right arrow keys move within the edited cell.

- When a **Value** cell is selected, press CTRL+F11 to open the Value Configuration Options dialog box.
- When the entire step is selected (by clicking to its left), use the + key (expands a specific branch), - key (collapses a specific branch), and * key (expands all branches) to expand and collapse the **Item** tree.
- When a row is selected (not a specific cell), you can type a letter to jump to the next row that starts with that letter.

Note: In addition to the above commands, you can also use QuickTest menu shortcuts. For more information, see “Performing Commands Using Shortcut Keys” on page 45.

Understanding the QuickTest Recorded Object Hierarchy

When you add a step to your test in the Keyword View, the step is added as a sibling step or sub-step of the selected step, according to the QuickTest recorded object hierarchy.

The recorded object hierarchy comprises two or more levels of test objects. The top level is the 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 a third level object, for example, Browser > Page > WebButton.

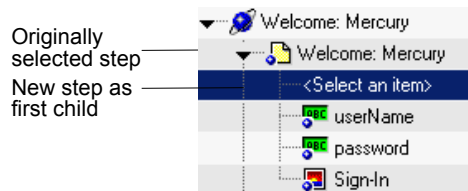
Note: When testing ActiveX objects in a browser, the top-level ActiveX object is recorded within the standard Web object hierarchy, for example, Browser > Page > ActiveX.

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**.

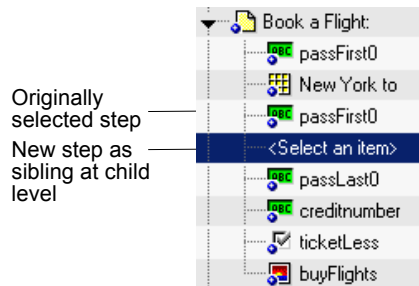
An object that can potentially contain a lower-level object is called a container object. All top-level objects in the recorded hierarchy are container objects. If a second-level object contains third-level objects according to the QuickTest recorded object hierarchy, then that object is also considered a container object. For example, in the step **Browser > Page > Edit > Set “David”**, Browser and Page are both container objects.

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, the new step is inserted as the first sub-step of the container object.



- If the selected step is at the lowest level of the recorded hierarchy, the new step is inserted as a sibling step immediately after the selected step.



Adding a Standard Step to Your Test

You can use the Keyword View to add a step at any point in your test. You can add a step below the currently selected step, at the end of a test, or at the beginning of a new test. You can also add a new step immediately after a conditional or loop block, as described in “Adding a Standard Step After a Conditional or Loop Block” on page 135.

Tip: You can also add a step using the Step Generator. For more information, see “Inserting Steps Using the Step Generator” on page 541.

To add a standard step:

1 Perform one of the following:

- Click anywhere in the Keyword View (below the existing steps, if any) to add a step at the end of the test. If no steps are defined yet, this adds the first step to the test.
- Choose **Insert > New Step** to add a new step after the existing steps (if any). If the test does not contain any steps, this adds the first step to the component.
- Select an existing step and choose **Insert > New Step** to add a new step between existing steps. (If you select the last step, QuickTest adds a step at the end of the test.)
- Right-click an existing step and choose **Insert New Step** from the context-sensitive menu.

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”.

Note: The **Select an item** list is generally expanded to display all applicable test objects, as well as the **StepGenerator** and **Statement** items.

- 2** Define the step by clicking in the cell for the part of the step you want to modify and specifying its contents, as described below. Each cell in the step row represents a different part of the step. For each step, you can define the following:
- **Item.** A test object on which you perform a step. You must select an option from the **Item** column before you can add additional content to a step. For more information, see “Selecting an Item for Your Step” on page 122.
 - **Operation.** The operation to be performed on the item. For more information, see “Selecting the Operation for Your Step” on page 128.
 - **Value.** (If relevant.) The argument values for the selected operation. For more information, see “Defining Values for Your Step Arguments” on page 129.
 - **Assignment.** (If relevant.) The variable value. Double-click in the left part of the **Assignment** cell if you want to create or edit an assignment to or from a variable. Click the arrow button to select either **Get from** or **Store in**, depending on whether you want to retrieve the value from a variable or store the value in a variable. Click in the right part of the **Assignment** cell to specify or modify the name of the variable.
 - **Comment.** (If relevant.) Textual notes about the step. For more information, see “Working with Comments” on page 136.

Note: The **Documentation** cell is read-only. This cell displays an explanation of what the step does in an easy-to-understand sentence, for example, Click the “Sign-in” image. or Select “San Francisco” in the “toPort” list. In most cases, QuickTest can generate the description displayed in this cell.

If you created a function library and associated it with the test, QuickTest can display documentation for it only if you defined the relevant text in the function library. For more information, see “Documenting the Function” on page 1066 and “Managing Function Libraries” on page 83.

Tip: 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 define or modify your steps. You can also drag and drop steps to move them to a different location within your test. For more information, see “Managing Action Steps” on page 137 and “Using Keyboard Commands in the Keyword View” on page 140.

- 3** After you make your changes, save the test. For more information, see “Saving a Test” on page 105.

Selecting an Item for Your Step

An item can be any of the following:

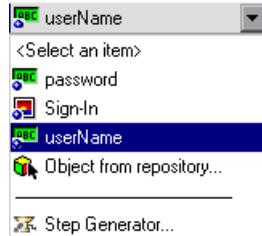
- A test object in the object repository. You can either choose a test object from the list, or choose **Object from repository** to open the Select Object for Step dialog box in which you can select a test object from the object repository or an object from your application. The test objects available 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.
- 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 541.

To select an item:

Click in the **Item** cell. Then click the down arrow and select the item on which you want to perform the step from the displayed list. When you insert a new step, the list is displayed automatically.

Selecting a Test Object from the Item List

The test objects available in the **Item** list are the sibling and child test objects of the previous step's test object, as defined in the shared object repository. The example below shows the objects available for the step following a **userName** test object.



To select a test object from the displayed Item list:

- 1 Click in the **Item** cell, then click the arrow button to display the **Item** list. If you have just created a new step, the list is displayed automatically as soon as you create the new step.
- 2 In the **Item** list, select the test object on which you want to perform the step. The item you select is displayed in the **Item** cell. You now need to specify an operation for the step. For more information, see “Selecting the Operation for Your Step” on page 128.

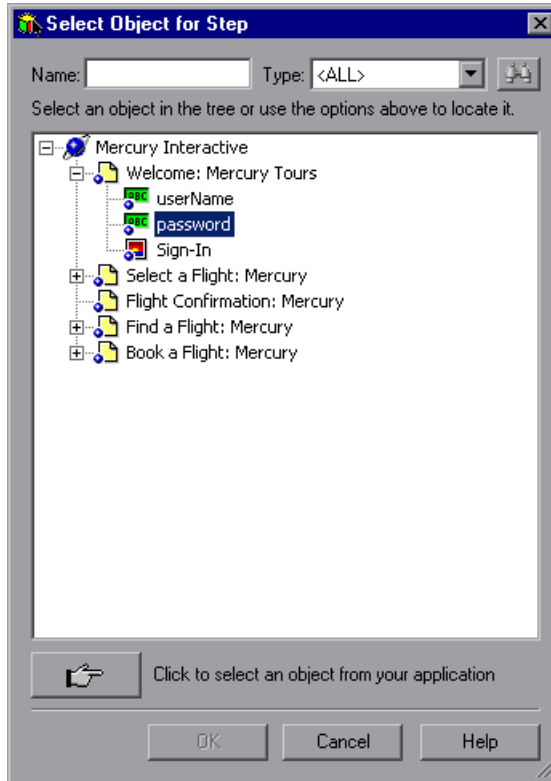
Selecting a Test Object from the Shared Object Repository

You can select any object in the object repository tree for your new step, or you can select the **Insert run-time object** option to enter an **Object** statement for the selected test object in your test. If the object repository is very large, you can search for the object. For example, you may want to add a **password** object that you know is an Edit box. You can search all the **Edit** type objects for one called **password**, or even one containing the letter **p**.

For more information on the object repository, see Chapter 6, “Working with Test Objects.” For more information on **Object** statements, see “Accessing Run-Time Object Properties and Methods” on page 1029.

To select a test object from the shared object repository:

- 1 Click in the **Item** cell, then click the arrow button to display the **Item** list. If you have just created a new step, the list is displayed automatically as soon as you create the new step.
- 2 In the **Item** list, choose **Select another object**. The Select Object for Step dialog box opens.



- 3 Select an object from the object repository tree. If the object repository is very large, you can search for the object, as described below. If a search is not required, proceed to step 8.

- 4 In the **Name** box, enter the name of the object, or any part of the name. For example, you can enter **p** to search for all object names containing the letter **p**.

Note: If the **Name** box is left empty, all objects of the selected object type are considered matching criteria.

- 5 In the **Type** box, select the type of object for which to search, or select **<All>** to search for the object in all the object types.

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



- 6 Click the **Find Next** button. The search starts at the currently selected node, and the number of objects that match your criteria is displayed. The first object in the list that matches your criteria is highlighted.
- 7 If required, click the **Find Next** button to navigate through all the objects that match your search criteria. The search continues to the end of the tree, then wraps to the beginning of the tree, and continues.

Tip: Press **F3** to find the next object that matches your search criteria, or **SHIFT+F3** to find the previous match.

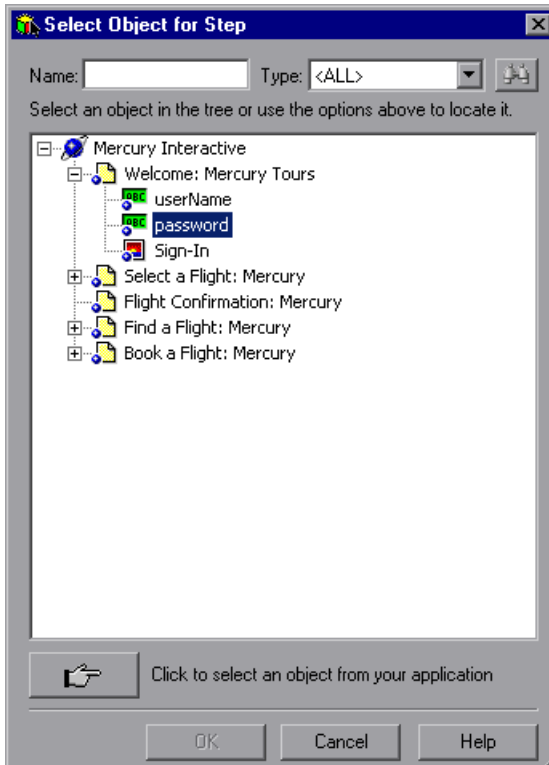
- 8 Click **OK**. The object is displayed in the **Item** column of the Keyword View, and is also added to the **Item** list. You can now specify the operation for the selected object. For more information, see “Selecting the Operation for Your Step” on page 128.

Selecting a Test Object from Your Application

If the shared object repository does not include the test object that you need for this step, you can select it directly from your application and add it to the shared object repository so that you can use it in this and other steps.

To add a test object from your application:

- 1 Click in the **Item** cell, then click the arrow button to display the **Item** list. If you have just created a new step, the list is displayed automatically as soon as you create the new step.
- 2 In the **Item** list, choose **Select another object**. The Select Object for Step dialog box opens.

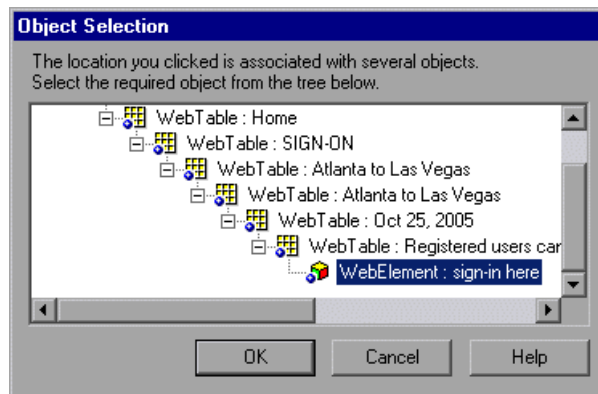


- 3 Click the pointing hand button. QuickTest is minimized.

- 4 Use the pointing hand to click on the required object in your application.

Tip: You can hold the left CTRL key to change the window focus or perform operations such as right-clicking or moving the pointer over an object to display a context menu. If the window containing the object you want to 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. Additionally, if the window containing the object you want to select is minimized, you can display it by holding the left CTRL key, right-clicking the application in the Windows task bar, and choosing **Restore** from the context menu.

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



- 5 Select the object for the new step and click **OK**. The object is displayed in the shared object repository tree in the Select Object for Step dialog box.

- 6 Click **OK**. The object is displayed in the **Item** column in the Keyword View. You can now specify the operation for the selected object. For more information, see “Selecting the Operation for Your Step” on page 128.

Tips: If you select an object in your application that is not in the shared object repository, a test object is added to the local object repository when you insert the new step. After you add a new test object to the local object repository, it is recommended to rename it, if its name does not clearly indicate its use. For example, you may want to rename a test object named **Edit** (that is used for entering a username) to **UserName**. This will enable other users to select the appropriate test object when adding steps using test objects located in this shared object repository.

After you add the required objects to the local object repository, you can use the Object Repository Merge Tool to update the shared object repository and make the new objects available to other tests. For more information, see “Updating a Shared Object Repository from Local Object Repositories” on page 1168.

If you are adding a container test object, it is also recommended to specify its context, for example, if you are adding a confirmation message box from a Login page, you may want to name it **Login > Confirm**. For more information, see “Renaming Test Objects” on page 174.

Selecting the Operation for Your Step

The **Operation** cell specifies the operation to be performed on the item listed in the **Item** column. The available operations vary according to the item selected in the **Item** column. When you select an item, all operations associated with that item are listed.

For example, if you selected a browser test object, such as a **WebButton** object, the list contains all of the available methods, such as **Click** or **Exist**.

To select an operation for the step:

Click in the **Operation** cell. Then click the down arrow button and select the operation to be performed on the item. The available operations vary according to the item selected in the **Item** column. For example, if you selected a browser test object, the list contains all of the methods and properties available for the browser object. If you selected a test object in the **Item** column, the default operation (most commonly-used operation) for the test object is automatically displayed in the **Operation** column. This cell is not applicable if you chose to insert a statement in the **Item** column.




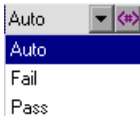
Note: Even if the **Item** column in the Keyword View is displayed to the right of the **Operation** column, you must still first select an item to view the list of available operations in the **Operation** column.

Defining Values for Your Step Arguments

The **Value** cell lists the value(s) for the operation argument(s). You can insert a constant value or a parameter.

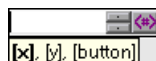
You can also encode password values. For more information, see “Inserting Encoded Passwords into Method Arguments and Data Table Cells” on page 132.

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. If you modify a cell that contains a string enclosed by quotes by removing the quotes, QuickTest will not restore the quotes and the value will be treated as a variable name.
	Integer	Enables you to enter any number, or use the up and down arrows to select a number.
	Boolean	Enables you to select a True or False value from the list.
	Predefined Constant	Enables you to select a predefined value from the list.

To define or modify a value:

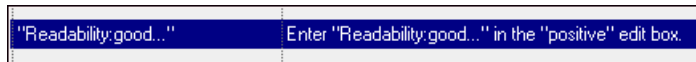
Click in each partition of the **Value** cell and enter the argument values for the selected operation. Note that when you click in the **Value** cell, a tooltip displays information for each argument. In the tooltip, the argument for the partition that is currently highlighted is displayed in bold, and any optional arguments are enclosed in square brackets.



Note: After you enter the initial value, you can edit the value at any time in the Keyword View for a test object, utility object, function call, conditional statement, or loop statement. You cannot edit the value of a regular statement, such as `x=10`, in the Keyword View after you define its initial value. You can edit the previously defined value of a regular statement only in the Expert View.


To add multi-line arguments:

You can also add multi-line argument values by pressing **SHIFT+ENTER** to add line breaks to your argument value. After you enter a multi-line argument value, QuickTest automatically converts it to a string, and displays only the first line of the argument, followed by an ellipsis (...). This format for multi-line argument values is also displayed in the Documentation column of the Keyword View.



Tip: Select the cell to display the entire argument value to be used in the step. Note that the argument value is used during the run session exactly as it appears in the step. For example, if you enter quotation marks as part of the argument value, they will be included in the argument value used during the run session. QuickTest automatically interprets a multi-line value as a string, so you do not need to add quotation marks for this purpose.

To parameterize the value for an argument:

Click the  button in the required **Value** cell. For more information, see “Parameterizing Values” on page 365.

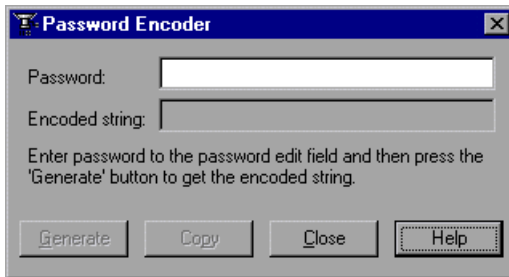
Inserting Encoded Passwords into Method Arguments and Data Table Cells

You can encode passwords to use the resulting strings as method arguments or Data Table parameter values. For example, your Web site may include a form in which the user must supply a password. You may want to test how your site responds to different passwords, but you also want to ensure the integrity of the passwords. The **Password Encoder** enables you to encode your passwords and place secure values into the Data Table.

Tip: You can also encrypt strings in Data Table cells using the Encrypt option in the Data Table menu. For more information, see “Data Menu” on page 527.

To encode a password:

- 1 From the Windows menu, choose **Start > Programs > QuickTest Professional > Tools > Password Encoder**. The Password Encoder dialog box opens.



- 2 Enter the password in the **Password** box.
- 3 Click **Generate**. The Password Encoder encrypts the password and displays it in the **Encoded String** box.
- 4 Use the **Copy** button to copy and paste the encoded value into the Data Table.
- 5 Repeat the process for each password you want to encode.
- 6 Click **Close** to close the Password Encoder.








Adding Other Types of Steps to Your Test

In addition to adding standard statement steps to your test using the Keyword View, you can also insert the following special types of steps using the relevant options from the **Insert** menu. Each step is entered as a row in the Keyword View, and you can then modify it as described in “Modifying the Parts of a Step” on page 136.

- You can insert a checkpoint step. For more information see “Understanding Checkpoints” on page 225.
- You can insert an output value step. For more information see “Outputting Values” on page 411.
- You can insert comments in steps to separate parts of an action or a test and to add details about a specific part. For more information, see “Adding Comments” on page 578.
- You can insert a step that sends information to the results, a step that puts a comment line in your test, a step that synchronizes your test with your application, or a step that measures a transaction in your test. For more information see “Adding Steps Containing Programming Logic” on page 539.
- You can insert a step that calls a WinRunner test or function. For more information see “Working with WinRunner” on page 1249.
- You can control the flow of your test by using conditional statements and loop statements. For more information, see “Using Conditional and Loop Statements in the Keyword View” on page 134.

Using Conditional and Loop Statements in the Keyword View

Using conditional statements, you can incorporate decision making into your tests. Using loop statements, you can run a group of steps repeatedly, either while or until a condition is true. You can also use loop statements to repeat a group of steps a specific number of times. Each statement type is indicated by one of the following icons in the Keyword View:

Icon	Type
	If...Then statement
	ElseIf...Then statement
	Else statement
	While...Wend statement
	For...Next statement
	Do...While statement
	Do...Until statement

After you insert a conditional or loop statement in the Keyword View, you can insert or record steps after the statement to include them in the conditional or loop block.

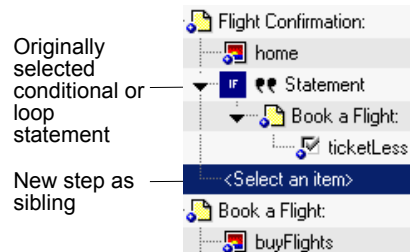
For information on including conditional and loop statements in your test, see Chapter 21, “Adding Steps Containing Programming Logic.”

Adding a Standard Step After a Conditional or Loop Block

After you add a conditional or loop statement to your test, all steps that you add or record are automatically inserted within the conditional or loop statement block. After you have finished adding steps to the block, you can add a step outside of the block, at a sibling level to the conditional or loop statement step, as described below. For more information on conditional and loop statements, see Chapter 21, “Adding Steps Containing Programming Logic.”

To add a standard step after a conditional or loop block:

- 1 Select the conditional or loop statement step after and outside of which you want to add the new step, and 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 “Adding a Standard Step to Your Test” on page 120.

Modifying the Parts of a Step

You can modify any part of a step in the Keyword View. For example, you can change the test object on which the step is performed, change the operation to be performed in the step, or add information regarding a step in the **Comment** column.

When working in the Keyword View, you can use the standard editing commands (**Cut**, **Copy**, **Paste** and **Delete**) in the **Edit** menu or in the context-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 in a test or in an action. For more information, see “Managing Action Steps” on page 137.

To modify a step, click in the cell containing the part of the step you want to modify and specify the content of the call. Each cell in the step row represents a different part of the step. For more information, see “Adding a Standard Step to Your Test” on page 120.

Working with Comments

A **Comment** is free text entry. You can insert a comment in the **Comment** cell of a step, or you can add a comment in a separate step. Using comments can help improve readability and make a test easier to update. For example, you may want to add a comment step at the beginning of each action to specify what that section includes.

After you add a comment, it is always visible as long as one or more columns are displayed. For information on selecting columns to display, see “Defining Keyword View Display Options” on page 141. QuickTest does not process comments when it runs a test.

To add a comment to an existing step:

Select the step and type your comment in the **Comment** column.

Note: You can also insert a comment step. For more information, see “Adding Comments” on page 578.

To modify an existing comment:

Double-click the comment in the **Comment** column. The cell becomes a free text field.

Managing Action Steps

You can move an action step before or after any other step in an action. You can also delete it if it is no longer required.

Moving an Action Step

You can move an action or step to a different location within a test, as needed.

To move a step in the Keyword View:

- In the **Item** column, drag the step up or down and drop it at the required location. When you drag a selected step, a line is displayed, enabling you to see the location to which the step will be moved. If you drag a step within its parent object, the step is displayed in the new position under its parent. If you move the step to a different parent object, the parent is duplicated, and the step is moved below it.
- Copy or cut the step to the Clipboard and then paste it in the required location. You can use **Edit > Copy** or CTRL + C to copy the step, and **Edit > Cut** or CTRL + X to cut the step, and **Edit > Paste** or CTRL + V to paste the step. When you move, copy, or cut an action or step, you also move, copy, or cut all of its sub-steps, if any.

Notes:

Conditional and loop blocks can only be copied or cut in their entirety. QuickTest does not enable you to copy or cut only the child nodes of conditional or loop blocks. After you copy or cut conditional or loop blocks to the Clipboard, QuickTest enables you to paste them only in valid locations.

You cannot copy or cut a parent object together with only some of its child objects. You must either select only the parent (which automatically includes all its child objects) or the parent object together with all of its children.

If you copy an action (**Insert > Call to Copy of Action**, right-click an action icon and select **Insert Call to Copy of Action**, or right-click any step and select **Action > Insert Call to Copy**) the Select Action dialog opens, which enables you to insert a call to a copy of an action. For more information on inserting a call to a copy of an action, see “Inserting Calls to Copies of Actions” on page 858.

Deleting an Action Step

You can delete an action step, if required. Before you delete a step, make sure that removing it will not prevent the action from running correctly. When an item has both an operation and sub-steps defined for it, as in the example below, you can choose whether to delete only the operation of the item, or to delete the item and all of its sub-steps.

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 a step if one of its cells is in edit mode.

To delete a step:

- 1 Select the row for the item you want to delete.
- 2 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).

Using Keyboard Commands in the Keyword View

If you prefer to use your keyboard, you can use the following keyboard commands to navigate within the Keyword View:

- Press F8 to add a new step below the currently selected step.
- Press SHIFT+F8 to add a new step after a conditional or loop block.
- Press F7 to use the Step Generator to add a new step below the selected step.
- The TAB and SHIFT+TAB keys move the focus left or right within a single row, unless you are in a cell that is in edit mode. If so, press ENTER to exit edit mode, and then you can use the TAB keys.
- When a cell containing a list is selected:
 - You can press SHIFT+F4 to open the list for that cell.
 - You can change the selected item by using the up and down arrow keys. In the **Item** column, the list must be open before you can use the arrow keys.
 - You can type a letter or sequence of letters to move to a value that starts with the typed letter(s). The typed sequence is highlighted white.

Defining Keyword View Display Options

You can choose how you want to display the information in the Keyword View using the Keyword View Options dialog box. You can customize the display of the Keyword View columns, fonts, and colors. The options you set remain in effect for all tests in all subsequent sessions on your computer.

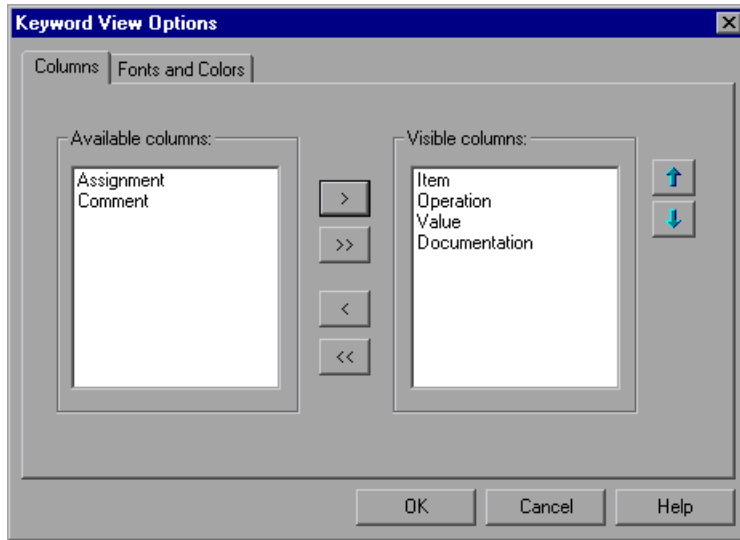
Displaying Keyword View Columns

You can use the Columns tab of the Keyword View Options dialog box to specify which columns you want to display in the Keyword View. You can also specify the order in which the columns are displayed.

Tip: You can display only the **Documentation** column by right-clicking the column header row and choosing **Documentation Only** from the displayed menu. You can then print the Keyword View for use as instructions for manual testing. For more information on printing from the Keyword View, see “Printing a Test” on page 108.

To specify the Keyword View columns to display:

- 1 Choose **Tools > View Options**. The Keyword View Options dialog box opens.



The **Available columns** list shows columns not currently displayed in the Keyword View. The **Visible columns** list shows columns currently displayed in the Keyword View.

- 2 Double-click column names or choose column names and click the arrow buttons (> and <) to move them between the **Available columns** and **Visible columns** lists.

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



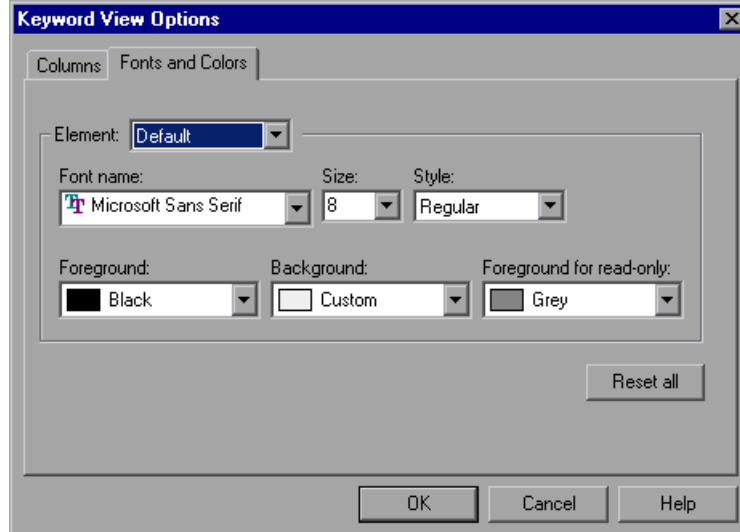
- 3 In the **Visible columns** list, set the order in which columns appear in the Keyword View by selecting one or more columns and then using the up and down arrow buttons.

Note: The order of the columns in the Keyword View does not affect the order in which the cells need to be completed for each step. For example, if you choose to display the **Operation** column to the left of the **Item** column, you still need to select the item first, and only then is the **Operation** column list refreshed to match the selection you made in the **Item** column.

- 4 Click **OK** to close the dialog box and apply the new column display.

Setting Keyword View Fonts and Colors

You can use the Fonts and Colors tab of the Keyword View Options dialog box to specify different text and color display options for different elements in the Keyword View.



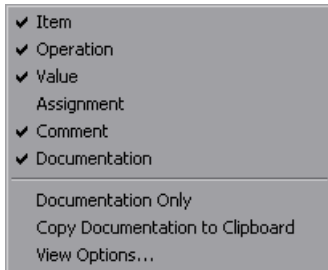
The Fonts and Colors tab includes the following options:

Option	Description
Element	<p>You can specify different font and color options for each of these Keyword View elements. Select one of the following elements to see the current definitions and modify them:</p> <ul style="list-style-type: none"> • Alternate Rows. The background color of every other row. The font and text color for the alternate rows is the same as the font and text color defined for the Default element. • Comment. The row and text of comment lines. Note that all of the available formatting options apply to entire comment rows, not to comments within a regular step row. For comments within a step row, only the specified Foreground color applies (all other settings are taken from the Alternate Rows, Default, or Selected Row settings, as appropriate). • Default. All rows and text in the Keyword View (except for the elements listed below). • Selected Row. The row and text currently selected (highlighted).
Font Name	<p>Enables you to modify the font used for text in the selected element. You cannot change the font for Alternate Rows or Selected Row elements.</p> <p>Note: When testing in a Unicode environment, you must select a Unicode-compatible font. Otherwise, elements in your test may not be correctly displayed in the Keyword View. However, the test will still run in the same way, regardless of the font you choose.</p>
Size	<p>Enables you to modify the font size used for text in the selected element. You cannot change the font size for Alternate Rows or Selected Row elements.</p>

Option	Description
Style	Enables you to modify the font style used for text in the selected element. You can select Regular , Bold , Italic , or Underline font styles. You cannot change the font style for Alternate Rows or Selected Row elements.
Foreground	Enables you to modify the text color for the selected element. You cannot change the foreground color for Alternate Rows .
Background	Enables you to modify the row color for the selected element.
Foreground for read-only	Enables you to modify the text color for rows that are read-only. This option cannot be changed for Alternate Rows .
Reset all	Resets all Fonts and Colors tab options to the default settings.

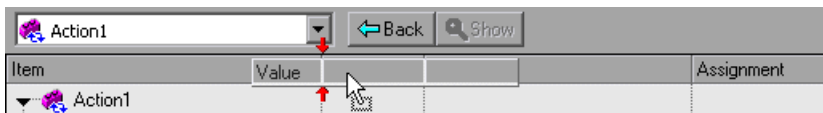
Tips for Working with the Keyword View

- You can display or hide specific columns by right-clicking the column header row in the Keyword View and then selecting or deselecting the required column name from the displayed menu.



You can quickly display only the **Documentation** column, for example, if you want to print the steps for use as instructions for manual testing, by selecting **Documentation Only**.

- You can rearrange columns by dragging a column header to its new location in the Keyword View. Red arrows are displayed when the column header is dragged to an available location.



Viewing Properties of Step Elements in the Keyword View

You can view properties for different parts of a step in the Keyword View. For example, you can view object properties, action properties, action call properties, checkpoint properties, and output value properties. Right-click the item whose properties you want to view, and select the relevant option from the displayed menu.

The property options available in the **Step** menu or the context (right-click) menu change according to the currently selected step. For example, if you right-click a step that contains a checkpoint or output value on a test object, you can view object properties and checkpoint or output value properties for the current object and checkpoint or output value. If you right-click an action, you can choose to view action properties or action call properties for the current action.

Working with Breakpoints in the Keyword View

You can 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.

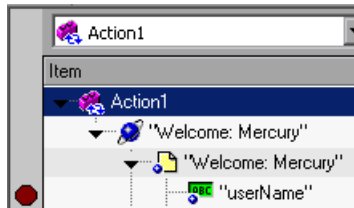
To insert a breakpoint in the Keyword View:

- Click in the left margin at the point where you want to insert the breakpoint.
- Select a step and press F9.
- Choose **Debug > Insert/Remove Breakpoint**.

A red breakpoint icon  is displayed.

To remove a breakpoint from the Keyword View:

- Click the breakpoint icon.
- Select a step and press F9.
- Choose **Debug > Insert/Remove Breakpoint**.



Note: QuickTest automatically places the breakpoint next to the appropriate item for the step. In the example shown above, even if you click next to the **Welcome: Mercury** browser or page item, the breakpoint is automatically inserted next to the **userName** edit item, on which the step is actually performed. When you collapse items, the breakpoint icons remain in the left margin next to the closest visible item, so you can see that the test contains breakpoints.

For more information on breakpoints, see “Using Breakpoints” on page 596.

6

Working with Test Objects

This chapter explains how to manage and maintain the test objects in your test. 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.

This chapter describes:	On page:
About Working with Test Objects	150
Understanding Object Repository Types	151
Understanding the Object Repository Window	156
Viewing and Modifying Test Object Properties	163
Mapping Repository Parameter Values	185
Adding Objects to the Object Repository	189
Defining New Test Objects	200
Copying, Pasting, and Moving Objects in the Object Repository	202
Deleting Objects from the Object Repository	205
Locating Objects	206
Working with Test Objects During a Run Session	213
Managing Shared Object Repository Associations	214
Exporting Local Objects to an Object Repository	217

About Working with Test Objects

When QuickTest runs a test, it simulates a human user by moving the pointer 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**.

As QuickTest learns the test objects, it stores them in the action's **local object repository**. You can choose to keep the stored objects in the local object repository, or you can choose to store the objects in a **shared object repository**. Storing the objects in the local object repository makes them available only to the specific action, but not to other actions. Storing the objects in one or more shared object repositories enables multiple tests to use them. You can also work with a combination of local and shared object repositories, as needed. For more information on local and shared object repositories, see “Understanding Object Repository Types” on page 151.

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

You can modify objects stored in a local object repository using the Object Repository window, as described in this chapter. You can modify objects in a shared object repository using the Object Repository Manager. For information on the Object Repository Manager, see Chapter 38, “Managing Object Repositories.” You can also copy objects from a shared object repository to a local object repository and then modify the local copy of the object using the Object Repository window, as described in this chapter.

You can also manipulate some aspects of a local object repository using the QuickTest Object Repository automation object model. For example, you can add, remove, and rename test objects in the local object repository. For more information, see “Manipulating Object Repositories Using Automation” on page 1148.

Understanding Object Repository Types

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

When you plan and create tests, you must consider how you want to store the objects in your tests. You can store the objects for each action in its corresponding local object repository, or you can store the objects in your tests in one or more shared object repositories. By storing objects in shared object repositories and associating these repositories with your actions, you enable multiple actions to use the objects. For each action, you can use a combination of objects from your local and shared object repositories, according to your needs. You can also transfer local objects to a shared object repository, if required. This reduces maintenance and enhances the reusability of your tests because it enables you to maintain the objects in a single, shared location instead of multiple locations. For more information, see “Deciding Whether to Use Local or Shared Object Repositories” on page 153.

If you are new to using QuickTest, you may want to use local object repositories. In this way, you can record and run tests without creating, choosing, or modifying shared object repositories because all objects are automatically saved in a local object repository that can be accessed by its corresponding action. If you modify an object in the local object repository, your changes do not have any effect on any other action or any other test (except tests that call the action, as described in “Inserting Calls to Existing Actions” on page 856).

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

If an object with the same name is located in both the local object repository and in a shared object repository associated with the same action, the action uses the local object definition. If an object with the same name is located in more than one shared object repository associated with the same action, the object definition is used from the first occurrence of the object, according to the order in which the shared object repositories are associated with the action. For more information on associating shared object repositories, see “Associating Object Repositories with Actions” on page 488.

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

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

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

Note for users of previous QuickTest versions: When you open a test that was created using a version of QuickTest earlier than version 9.1, you are asked whether you want to convert it or view it in read-only format. Whether you choose to open it in read-only format or convert it, the object repositories are associated to the test as follows:

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

Deciding Whether to Use Local or Shared Object Repositories

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

In general, the local object repository is easiest to use when you are creating simple 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 is generally the preferred option when:

- You are creating tests using keyword-driven methodologies (not using record).
- 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.

Understanding the Local Object Repository

When you use a local object repository, QuickTest uses a separate object repository for each action. (You can also use one or more shared object repositories if needed. For more information, see “Understanding the Shared Object Repository” on page 155.) The local object repository is fully editable from within its action.

When working with a local object repository:

- QuickTest creates a new (empty) object repository for each action.
- As you record operations on objects in your application, QuickTest automatically stores the information about those objects in the corresponding local object repository (if the objects do not already exist in an associated shared object repository).

QuickTest adds all new objects to the local object repository even if one or more shared object repositories are already associated with the action. (This assumes that an object with the same description does not already exist in one of the associated shared object repositories).

- If a child object is added to a local object repository, and its parents are in a shared object repository, its parents are automatically moved to the local object repository.
- Every time you create a new action, QuickTest creates a new, corresponding local object repository and begins adding test objects to the local object repository as you record or learn objects.
- If you learn or record on the same object in your application in two different actions, the object is stored as a separate test object in each of the local object repositories.
- When you save your test, all of the local object repositories are automatically saved with the test (as part of each action within the test). The local object repository is not accessible as a separate file (unlike the shared object repository).

Understanding the Shared Object Repository

When you use shared object repositories, QuickTest uses the shared object repositories you specify for the selected action. You can use one or more shared object repositories. (You can also save some objects in a local object repository for each action if you need to access them only from the specific action. For more information, see “Understanding the Local Object Repository” on page 154.)

After you begin creating your test, you can specify additional shared object repositories. You can also create new ones and associate them with your action. Before running the test, you must ensure that the object repositories being used by the test contain all the objects in your test. Otherwise, the test may fail. For more information, see “Adding Objects to the Object Repository” on page 189.

You modify a shared object repository using the Object Repository Manager. For more information, see Chapter 38, “Managing Object Repositories.”

When working with a shared object repository:

- If you record operations on an object that already exists in either the shared or local object repository, QuickTest uses the existing information and does not add the object to the object repository.
- If a child object is added to a local object repository, and its parents are in a shared object repository, its parents are automatically moved to the local object repository.
- QuickTest does not add an object to the shared object repository as you record operations on it. Instead, it adds new objects to the local object repository (not the shared object repository) as you learn objects or record steps on them (unless those same objects already exist in an associated shared object repository).

You can export the local objects to a shared object repository. For more information, see “Exporting Local Objects to an Object Repository” on page 217.

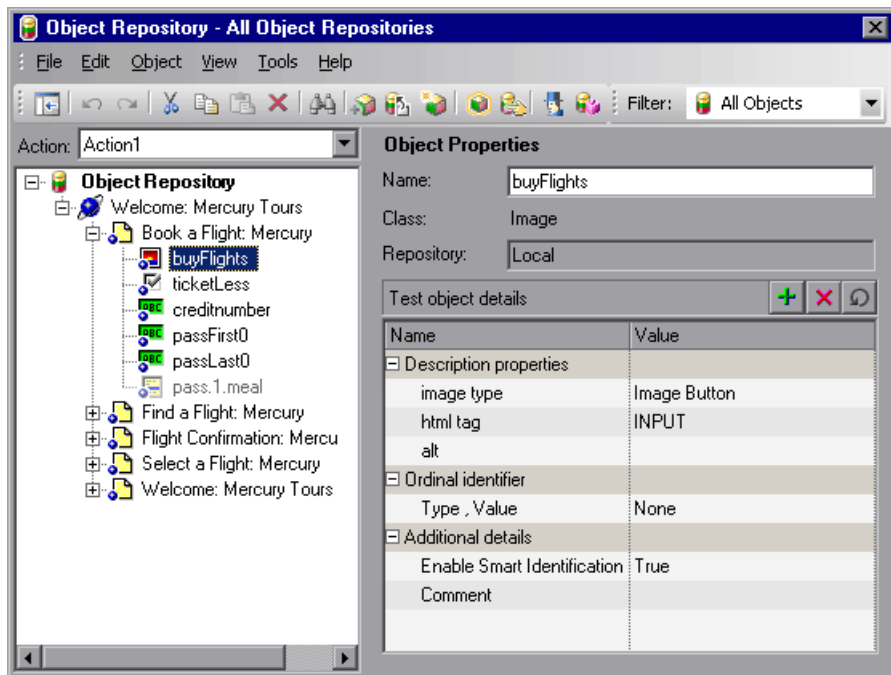
You can also merge the local objects directly to a shared object repository that is associated with the same action. This can reduce maintenance since you can maintain the objects in a single shared location, instead of multiple locations. For more information, see “Updating a Shared Object Repository from Local Object Repositories” on page 1168.

Understanding the Object Repository Window



You open the Object Repository window for a specific action by choosing **Resources > Object Repository** or clicking the **Object Repository** button.

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



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

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

Note: You can choose whether to show only the object repository tree, or the object repository tree together with the test object details area. For more information, see “Showing and Hiding the Test Object Details Area” on page 162.

You can use the Object Repository window to view the test object description of any test object in the repository (in local and shared object repositories), to modify local test objects and their properties, and to add objects to your local object repository.



Note: All changes you make to a local object are automatically updated in all steps that use the local object as soon as you make the change. You can use the **Edit > Undo** and **Edit > Redo** options or **Undo** and **Redo** buttons to cancel or repeat your changes. When you save the current test, you cannot undo or redo operations that were performed before the save operation.

For more information on viewing and modifying test object properties, see “Modifying Test Object Properties” on page 168.

Note: Even when steps containing a test object are deleted from your action, the objects remain in the object repository. You can delete objects from the local object repository using the Object Repository window. You can delete objects from a shared object repository using the Object Repository Manager. For more information, see “Managing Object Repositories” on page 1115.

The Object Repository window contains the following information:

Information	Description
Action	Enables you to select the action whose objects you want to view.
Object repository tree	<p>Contains all objects in the selected action (all local objects and all objects in any shared object repositories associated with the selected action).</p> <p>Note: If there are test objects in different associated object repositories with the same name, object class, and parent hierarchy, the object repository tree shows only the first one it finds based on the priority order defined. For information on object repository priorities, see “Associating Object Repositories with Actions” on page 488.</p> <p>You can filter the objects shown in the object repository tree. For more information, see “Filtering the Object Repository Window” on page 162.</p>
Name	The name that QuickTest assigns to the test object. You can change the name of a test object in the local object repository. For more information, see “Renaming Test Objects” on page 174.
Class	The class of the object.

Information	Description
Repository	The location (file name and path) of the object repository in which the object is located. If the object is located in the local object repository, Local is displayed.
Test object details	Enables you to view the properties and property values used to identify an object during a run session. You can also modify the test object details for a test object in the local object repository. For more information, see “Understanding the Test Object Details Area” on page 159. You can choose whether to show or hide the test object details area. For more information, see “Showing and Hiding the Test Object Details Area” on page 162.

Understanding the Test Object Details Area

The **Test object details** area of the Object Repository window enables you to view and modify the properties and property values used to identify an object during a run session.

Tip: You can choose whether to show or hide the test object details area. For more information, see “Showing and Hiding the Test Object Details Area” on page 162.

In the Object Repository window, objects in a shared object repository are displayed in the Object Properties pane (including the **Test object details** area) in read-only format. To modify objects in a shared object repository, open the shared object repository using the Object Repository Manager. For more information, see Chapter 38, “Managing Object Repositories.” You can also modify an object in a shared object repository by copying to the local object repository and then modifying the local copy. For more information, see “Copying an Object to the Local Object Repository” on page 164.

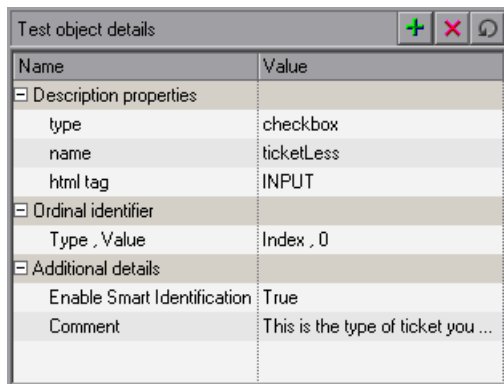
Tips:

You can view object properties and property values using the Object Properties dialog box. For more information, see “Viewing Object Properties and Property Values” on page 166.



You can use the Object Spy at any time to view run-time or test object properties and values of the objects in the application you are testing. You open the Object Spy by choosing **Tools > Object Spy** or clicking the **Object Spy** toolbar button. For more information, see “Viewing Object Properties Using the Object Spy” on page 70.

You can modify test object details for objects saved in the local object repository. You can also copy an object from a shared object repository to the local object repository, and then modify it.



Name	Value
[-] Description properties	
type	checkbox
name	ticketLess
html tag	INPUT
[-] Ordinal identifier	
Type , Value	Index , 0
[-] Additional details	
Enable Smart Identification	True
Comment	This is the type of ticket you ...



Note: All changes you make to a local object are automatically updated in all steps that use the local object as soon as you make the change. You can use the **Edit > Undo** and **Edit > Redo** options or **Undo** and **Redo** buttons to cancel or repeat your changes. When you save the current test, you cannot undo and redo operations that were performed before the save operation.

The **Test object details** area contains the following items:

Item	Description
Description properties	<p>The properties and property values used to identify the object during a run session.</p> <p>You can add and remove properties to or from the test object description. For more information, see “Adding Properties to a Test Object Description” on page 177.</p> <p>You can specify a property value as a constant, or you can parameterize the value. For more information, see “Specifying or Modifying Property Values” on page 170.</p>
Ordinal identifier	<p>A numerical value that indicates the object’s order or location relative to other objects with an otherwise identical description (objects that have the same values for all properties). For more information, see “Specifying Ordinal Identifiers” on page 183.</p>
Additional details	<p>Contains the following options:</p> <ul style="list-style-type: none"> • Enable Smart Identification. Enables you to select True or False to specify whether QuickTest should use Smart Identification to identify the test object during the run session if it is not able to identify the object using the test object description. <p>Note: This option is available only if Smart Identification properties are defined for the test object’s class in the Object Identification dialog box. For more information on Smart Identification, see “Configuring Smart Identification” on page 959.</p> <ul style="list-style-type: none"> • Comment. Enables you to add textual information about the test object.

Showing and Hiding the Test Object Details Area

You can choose to work with the Object Repository window in Compact View mode or Full View mode. Compact View mode displays only the object repository tree, while Full View mode displays the object repository tree together with the test object details area.

To change the Object Repository window view mode:

Perform one of the following, depending on the mode you want to show:



- Choose **View > Compact View** or click the **Compact View** button.

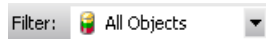


- Choose **View > Full View** or click the **Full View** button.

The Object Repository window switches to the selected view mode.

Filtering the Object Repository Window

You can use the Filter toolbar to filter the objects shown in the Object Repository window.



You can choose to show objects that meet one of the following criteria:

- All objects in the selected action (all local objects and all objects in any shared object repositories associated with the selected action)
- Only the local objects in the selected action
- Only the objects in a specific shared object repository associated with the current action

To filter the Object Repository window:

In the **Filter** toolbar list, select one of the following options:

- **All Objects**
- **Local Objects**
- The name of a specific shared object repository associated with the current action

The object repository tree is filtered to display only the objects from the location that you selected. The title bar of the Object Repository window indicates the current filter.

Viewing and Modifying Test Object Properties

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

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

For more information on different ways in which you can view and modify test object properties, see:

- “Copying an Object to the Local Object Repository,” below
- “Viewing Object Properties and Property Values” on page 166
- “Modifying Test Object Properties” on page 168
- “Specifying or Modifying Property Values” on page 170
- “Updating Test Object Properties from an Object in Your Application” on page 172
- “Restoring Default Properties for a Test Object” on page 174
- “Renaming Test Objects” on page 174
- “Adding Properties to a Test Object Description” on page 177
- “Defining New Test Object Properties” on page 180
- “Removing Properties from a Test Object Description” on page 182
- “Specifying Ordinal Identifiers” on page 183

Copying an Object to the Local Object Repository

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

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

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

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

To copy an object to the local object repository:

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

Viewing Object Properties and Property Values

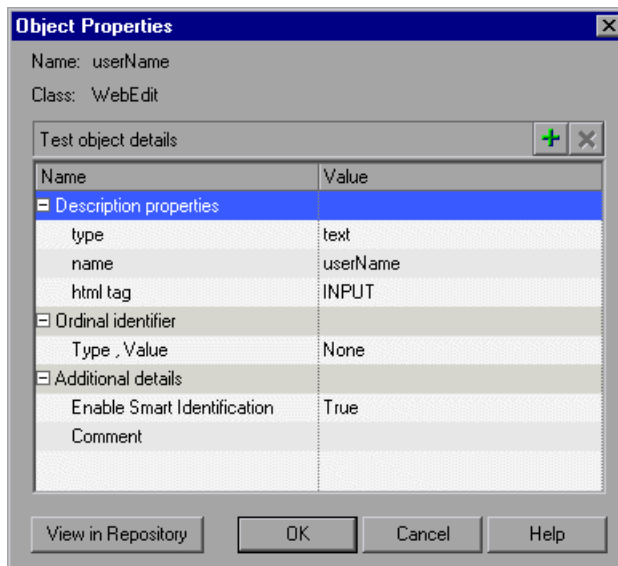
You can view test object properties and property values for objects in your test steps. You can also view test object properties and property values for objects in the Active Screen, regardless of whether the objects are stored in the object repository.

To view object properties and property values:

In your test, perform one of the following:

- Click in the step of the object whose properties you want to view and choose **Edit > Step Properties > Object Properties**
- In the Active Screen, right-click the object whose properties you want to view and choose **View / Add Object**

The Object Properties dialog box opens.



Note: There are slight differences in the Object Properties dialog box, depending on whether the selected object is currently stored in the local object repository, the shared object repository, or not stored in any object repository associated with the current test. This section describes options shown in the dialog box for options in the local object repository or not in any associated object repository. For objects stored in a shared object repository, this dialog box appears as for local objects (as shown above), but is in read-only format.

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

- View the object's properties and property values—its description properties, ordinal identifier, and other settings.
- Modify the properties and property values used to identify the object (for objects that are stored in the local object repository). You modify the properties and values in the Object Properties dialog box in the same way as you modify the test object details in the Object Repository window. For more information, see “Modifying Test Object Properties” on page 168.
- Click the **View in Repository** button (for objects that are stored in the object repository) to open the Object Repository window and display the selected object in the object hierarchy.
- Click the **Add to Repository** button (for objects that are not stored in the object repository) to add the selected object to the local object repository.

Modifying Test Object Properties

You can modify an object by modifying one or more of the object's property values or by changing the set of properties used to identify that object. You can do this for objects in the local object repository using the Object Repository window, and for objects in the shared object repository using the Object Repository Manager.

You can also automatically update the description of one or more test objects in your object repository based on the actual updated object properties in your application. For more information, see “Updating Test Object Properties from an Object in Your Application” on page 172.



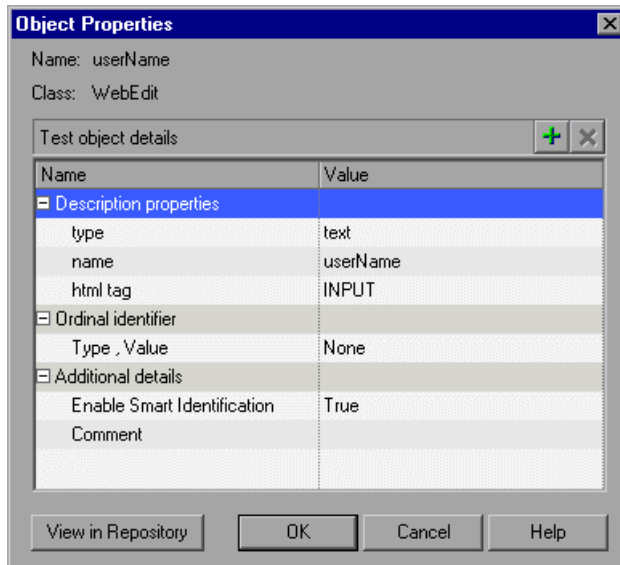
Tip: You can use the Object Spy at any time to view run-time or test object properties and values of the objects in the application you are testing. You open the Object Spy by choosing **Tools > Object Spy** or clicking the **Object Spy** toolbar button. For more information, see “Viewing Object Properties Using the Object Spy” on page 70.

To modify an object property:

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



Tips:

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



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

- 2 Modify the properties and values as required. You modify the properties and values in the Object Properties dialog box in the same way as you modify the test object details in the Object Repository window. For more information, see “Understanding the Test Object Details Area” on page 159 and “Viewing and Modifying Test Object Properties” on page 163.
- 3 Click **OK** to close the dialog box.

Specifying or Modifying Property Values

You can specify or modify values for properties in the test object description. You can specify a value using a constant value (either a simple value or a constant value that includes regular expressions) or you can parameterize it. You can do this for objects in the local object repository using the Object Repository window or Object Properties dialog box, and for objects in the shared object repository using the Object Repository Manager.

You can also find and replace specific object property values. For more information, see “Finding Objects in an Object Repository” on page 206.

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.”




Tip: You can use the Object Spy at any time to view run-time or test object properties and values of the objects in the application you are testing. You open the Object Spy by choosing **Tools > Object Spy** or clicking the **Object Spy** toolbar button. For more information, see “Viewing Object Properties Using the Object Spy” on page 70.

To specify a property value:

- 1 Select the test object whose property value you want to specify.
- 2 In the **Test object details** area, click in the value cell for the required property.

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






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

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



For information on specifying property values, see “Configuring a Selected Value” on page 350.

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

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

Updating Test Object Properties from an Object in Your Application


You can update an object in your object repository by selecting the corresponding object in your application and relearning its properties and property values from the application. When you update a test object description in this way, all currently defined properties and values are overwritten, including description properties and values, and ordinal identifier and Smart Identification information. Any object-specific comments that you may have entered are not removed.

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

You can also use this option to update an object that you defined (using the **Object > Define New Test Object** option) before the application was completely developed, and as a result some of the object properties and values are missing in the test object description, or are no longer sufficient to identify the object. For more information on the **Define New Test Object** option, see “Defining New Test Objects” on page 200.

You can do this for objects in the local object repository using the Object Repository window, and for objects in the shared object repository using the Object Repository Manager.

To update test object properties from an object in your application:

- 1 In the object repository tree, select the test object whose description you want to update.
- 2  Choose **Object > Update from Application** or click the **Update from Application** button. QuickTest is minimized and the cursor changes to a pointing hand, so that you can point to and click on any object in the open application.

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

Notes:

If the object you want to select is in a window that 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 25, “Setting Global Testing Options.” You can also hold the left CTRL key to change the window focus. Additionally, if the window containing the object you want to select is minimized, you can display it by holding the left CTRL key, right-clicking the application in the Windows task bar, and choosing **Restore** from the context menu.

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


If the location you click is associated with more than one object, the Select an Object dialog box opens. Select an object from the object tree and click **OK**.

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

Restoring Default Properties for a Test Object

You can restore the default properties for a selected test object. When you restore the default properties, it restores the mandatory property set defined for the selected object class in the Object Identification dialog box. Any changes that you have made to the description property set for the test object will be overwritten. However, if property values were defined for any of the mandatory properties they are not modified. In addition, restoring the default mandatory property set does not change the values for the ordinal identifier or Smart Identification settings for the test object.

To restore the mandatory property set:

- 1 In the object repository tree, select the test object whose description you want to restore.
- 2  In the **Test object details** area, click the **Restore mandatory property set** button.
- 3 Click **Yes** to confirm the operation. The test object's description properties are restored to the mandatory property set for the selected object class.

Renaming Test Objects

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

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

You rename objects in the local object repository using the Object Repository window. You rename objects in the shared object repository using the Object Repository Manager.

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

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

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

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

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

To rename a test object:

In the object repository tree, select the test object that you want to rename and perform one of the following:

- Choose **Edit > Rename** and enter the new name for the object in the selected node in the tree. Then press ENTER or click anywhere else to remove the focus from the object.
- Press F2 and enter the new name for the object.
- In the **Name** box in the Object Properties pane, enter the new name for the object. Then click anywhere else to remove the focus from the object.

Note: The name you assign to the object must be unique within the same class and hierarchy in the object repository. Object names are not case-sensitive.

Adding Properties to a Test Object Description

You can add to the list of properties that QuickTest uses to identify an object. For each object class, QuickTest has a default property set that it uses for the object description for a particular object. You can use the Add Properties dialog box to change the properties that are included in the object description. You can do this for objects in the local object repository using the Object Repository window or Object Properties dialog box, and for objects in the shared object repository using the Object Repository Manager.

Note: You can also add any valid test object property to a test object description, even if it does not appear in the Add Properties dialog box. For more information, see “Defining New Test Object Properties” on page 180.

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

You can also change the properties that identify an object if you want to reference objects using properties that 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 additional hypertext links to all past newsletters. The text in the first hypertext link on the page changes as the current newsletter changes, but it always links to a page called **current.html**. Suppose you want to create a step in your test in which you always click the first hypertext link in your archive page. Since the news is always changing, the text in the hypertext link keeps changing. You need to modify how QuickTest identifies this hypertext link so that it can continue to find it.

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

You can modify the default properties for a hypertext link for the object that was recorded 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.



Tip: You can use the Object Spy at any time to view run-time or test object properties and values of the objects in the application you are testing. You open the Object Spy by choosing **Tools > Object Spy** or clicking the **Object Spy** toolbar button. For more information, see “Viewing Object Properties Using the Object Spy” on page 70.

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 learn or record after you make the change. For more information, see “Configuring Object Identification” on page 943. You can also apply the changes you make in the Object Identification dialog box to the descriptions of all objects in an existing test using the **Update Run Mode** option. For more information, see “Updating a Test” on page 616.

To add properties to a test object description:

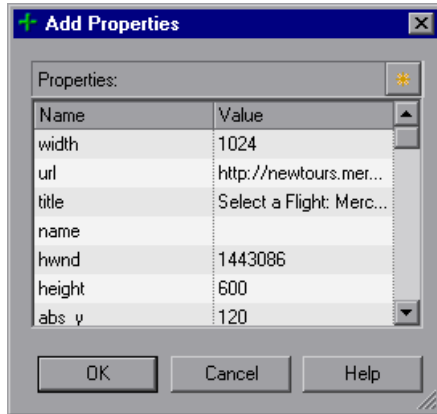
- 1 In the object repository tree, select the test object whose description you want to modify.
- 2 In the **Test object details** area, click the **Add description properties** button.



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

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

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



Notes: Values for all properties are displayed only if the application that contains the object is currently open. If the application is closed, only values for properties that were part of the object description when the object was learned are shown.

You can resize the Add Properties dialog box to enable you to view long property values.



You can click the **Define new property** button to add valid test object properties to this properties list. For more information, see “Defining New Test Object Properties” on page 180.

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

Tip: After you add a new property to the object description, you can modify its value. For more information on modifying object property values, see “Specifying or Modifying Property Values” on page 170.

Defining New Test Object Properties

You can add any valid test object property to a test object description, even if it does not appear in the Add Properties dialog box. You can do this for objects in the local object repository using the Object Repository window or Add Properties dialog box, and for objects in the shared object repository using the Object Repository Manager. For example, suppose you want QuickTest to use a specific property to identify your object, but that property is not listed in the Add Properties dialog box. You can open the Add Properties dialog box and add that property to the list.



Tip: You can use the Properties tab of the Object Spy to view a complete list of valid test object properties for a selected object. You open the Object Spy by choosing **Tools > Object Spy** or clicking the **Object Spy** toolbar button. For more information, see “Viewing Object Properties Using the Object Spy” on page 70.

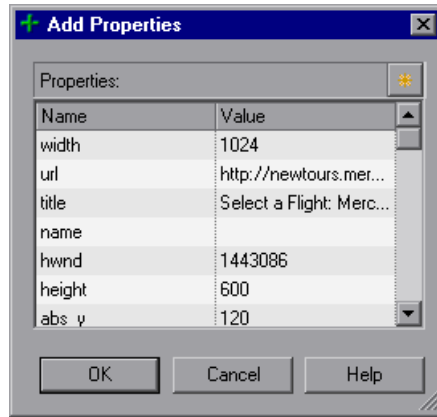
To define a new test object property:

- 1 In the object repository tree, select the test object for which you want to define a new property.
- 2 In the **Test object details** area, click the **Add description properties** button.

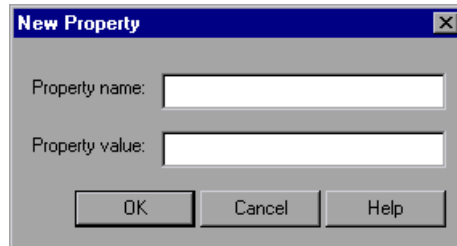


Tip: For an object in the local object repository, you can also select the required test object and choose **Edit > Step Properties > Object Properties**, click the **Add description properties** button, and then perform the following steps in the Add Properties dialog box.

The Add Properties dialog box opens.



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



- 4 Specify a valid 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.

- 5 Click **OK** to add the property to the list and close the New Property dialog box. The new property is highlighted in the Add Properties dialog box.
- 6 Click **OK** while the new property is highlighted to include it in the object description and close the Add Properties dialog box.

Removing Properties from a Test Object Description

You can remove properties from the description of a test object if you no longer want them to be part of the description. You can do this for objects in the local object repository using the Object Repository window or Object Properties dialog box, and for objects in the shared object repository using the Object Repository Manager.

To remove a property from a test object description:

- 1 In the object repository tree, select the test object whose description you want to modify.
- 2 In the **Test object details** area, select one or more properties that you want to remove from the test object description.

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



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

Specifying Ordinal Identifiers

An ordinal identifier assigns a numerical value to a test object that indicates its order or location relative to other objects with an otherwise identical description (objects that have the same values for all properties). This ordered value provides a backup mechanism that enables QuickTest to create a unique description to recognize an object when the defined properties are not sufficient to do so. You can specify the ordinal identifier for objects in the local object repository using the Object Repository window or Object Properties dialog box, and for objects in the shared object repository using the Object Repository Manager.

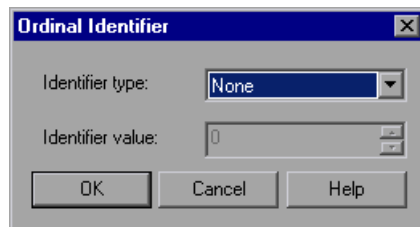
For more information on ordinal identifiers, see “Selecting an Ordinal Identifier” on page 952.

To specify an ordinal identifier:

- 1 Select the test object whose ordinal identifier you want to specify.
- 2 In the **Test object details** area, click in the cell to the right of the **Type, Value** cell under the **Ordinal identifier** row.

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

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



- 4** In the **Identifier type** box, select one of the following options:
 - **Location.** Indicates the order in which the object appears within the parent window, frame, or dialog box relative to other objects with an otherwise identical description.
 - **Index.** Indicates the order in which the object appears in the application code relative to other objects with an otherwise identical description.
 - **CreationTime** (Browser objects only). Indicates the order in which the browser was opened relative to other open browsers with an otherwise identical description. This identifier type is only available if more than one Browser object was open when the test object was learned.
 - **None.** Does not specify an ordinal identifier. This is the default value if no ordinal identifier was recorded or learned.
- 5** In the **Identifier value** box, enter the numeric value of the ordinal identifier.
- 6** Click **OK**. The ordinal identifier appears in the relevant row of the **Test object details** area for the selected object.

Mapping Repository Parameter Values

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

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

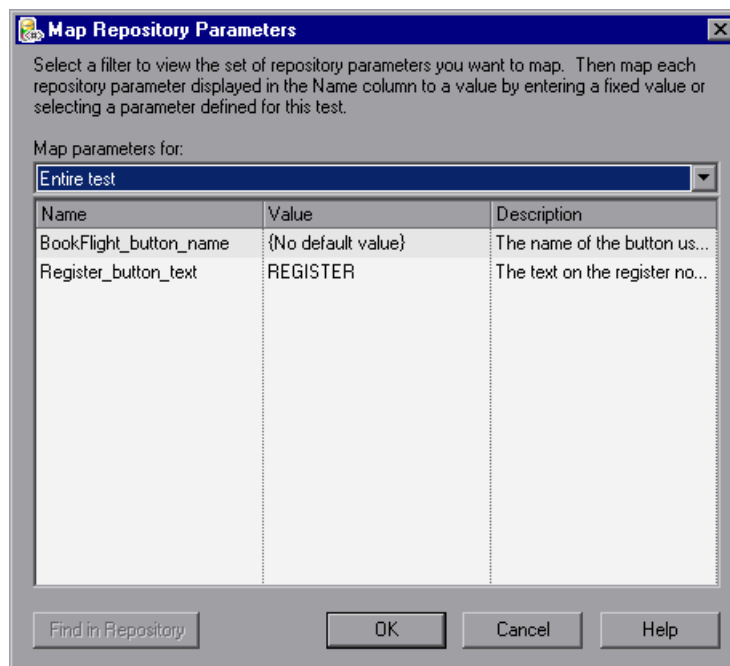
Before you map repository parameters, if you have more than one repository parameter with the same name in different shared object repositories that are associated with the same test, the repository parameter from the shared object repository with the highest priority (as defined in the shared object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

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

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



To map repository parameter values:

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






Tip: If you have unmapped repository parameters (repository parameters without a default value) in your test, you can also open this dialog box by double-clicking the **Repository Parameters** row in the Missing Resources pane. For more information, see Chapter 19, “Handling Missing Resources.”

The Map Repository Parameters dialog box contains the following options:

Option name	Description
Map parameters for filter	<p>Enables you to filter the list of parameters that is displayed. You can choose to display:</p> <ul style="list-style-type: none"> • All unmapped parameters. Displays all of the parameters in your test with unmapped values • Entire test. Displays all of the parameters in your test (with mapped or unmapped values) • <Action name>. (For example, LogIn) Displays all of the parameters in the specified action (with mapped or unmapped values)
Name column	The name of the repository parameter.
Value column	<p>The parameter's current value, if any. This column shows either the new value you defined, or the default value that was defined when the parameter was created. If no default value was defined, then the parameter is currently unmapped, and the text {No Default Value} is shown.</p> <p>You can perform one of the following:</p> <ul style="list-style-type: none"> • Enter a new constant value. • Parameterize the value by clicking in the Value cell of the relevant parameter and then clicking the parameterization button . • Reset a parameter to its default value by clicking in the Value cell of the relevant parameter and then clicking the Reset to Default Value button .
Description column	A textual description of the parameter, if any.
Find in Repository button	Opens the Object Repository window and highlights the first test object in the object repository tree that uses the selected repository parameter. You can click this button again to find the next occurrence of the selected parameter, and so forth.

Note: The repository parameter names, default values, and descriptions are defined in the Manage Repository Parameters dialog box. In addition, the names and descriptions can only be modified there. For more information, see “Managing Repository Parameters” on page 1134.

- 2 Click the **Map parameters for** arrow to select the list of parameter groups for which you want to define values. You can choose to display:
 - **All unmapped parameters.** Displays all of the parameters in your test with unmapped values
 - **Entire test.** displays all of the parameters in your test (with mapped or unmapped values)
 - **<Action name>.** (For example, LogIn) Displays all of the parameters in the specified action (with mapped or unmapped values)
- 3 Click in the **Value** cell of the parameter you want to map. You can choose to map the value in one of the following ways:
 -  ➤ Enter a new constant value or modify an existing constant value by typing directly in the **Value** cell. You can also enter a constant value in the Value Configuration Options dialog box by clicking the parameterization button. For information on using this dialog box, see “Configuring a Selected Value” on page 350.
 -  ➤ Parameterize the value by clicking the parameterization button. The Value Configuration Options dialog box opens. You can parameterize the value using a Data Table (Global sheet only), random number, environment, or test parameter. For information on using this dialog box, see “Configuring a Selected Value” on page 350.
 -  ➤ Restore the default value by clicking the **Clear Default Value** button. The default value, if any, that was defined in the Add Repository Parameter dialog box is displayed in the cell. For information on the Add Repository Parameter dialog box, see “Adding Repository Parameters” on page 1136.
- 4 Repeat step 3 for any additional parameter values that you want to map. Then click **OK** to close the Map Repository Parameter dialog box.

Adding Objects to the Object Repository

When you create a shared object repository for your keyword-driven testing infrastructure, you can add objects to it in a number of different ways. You can choose whether to add only a selected object, or to add all objects of a certain type, such as all button objects, or to add all objects of a specific class, such as all **WebButton** objects. In addition, if you record a test, QuickTest adds each object on which you perform an operation to the local object repository (for objects that do not already exist in an associated shared object repository). You can also add objects to the local object repository while editing your test.

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

Note: You can add an object to the local object repository only if that object does not already exist in a shared object repository that is associated with the action. If an object already exists in an associated shared object repository, you can add it to the local object repository using the **Copy to Local** option. For more information, see “Copying an Object to the Local Object Repository” on page 164.

If needed, you can merge test objects from the local object repository into a shared object repository. For more information, see Chapter 39, “Merging Shared Object Repositories.”

You can also add objects to a shared object repository while navigating through your application. For more information, see “Adding Objects Using the Navigate and Learn Option” on page 1131.

Tip: You can also add an object to the local object repository by choosing it from your application in the Select Object for Step dialog box (from a new step in the Keyword View or from the Step Generator).

Adding an Object Using the Add Objects to Local or Add Objects Option

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

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

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

1 Perform one of the following:



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



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

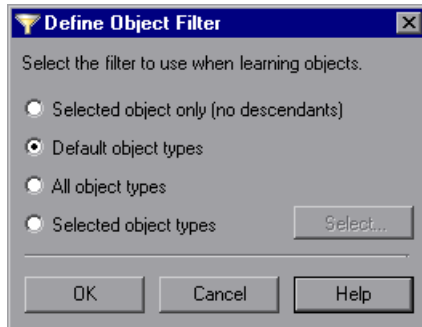
QuickTest and the Object Repository window or Object Repository Manager are minimized and the cursor becomes a pointing hand.

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 25, “Setting Global Testing Options.” You can also hold the left CTRL key to temporarily deactivate the pointing hand mechanism while you change the window focus. Additionally, if the window containing the object you want to select is minimized, you can display it by holding the left CTRL key, right-clicking the application in the Windows task bar, and choosing **Restore** from the context menu.

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

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

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

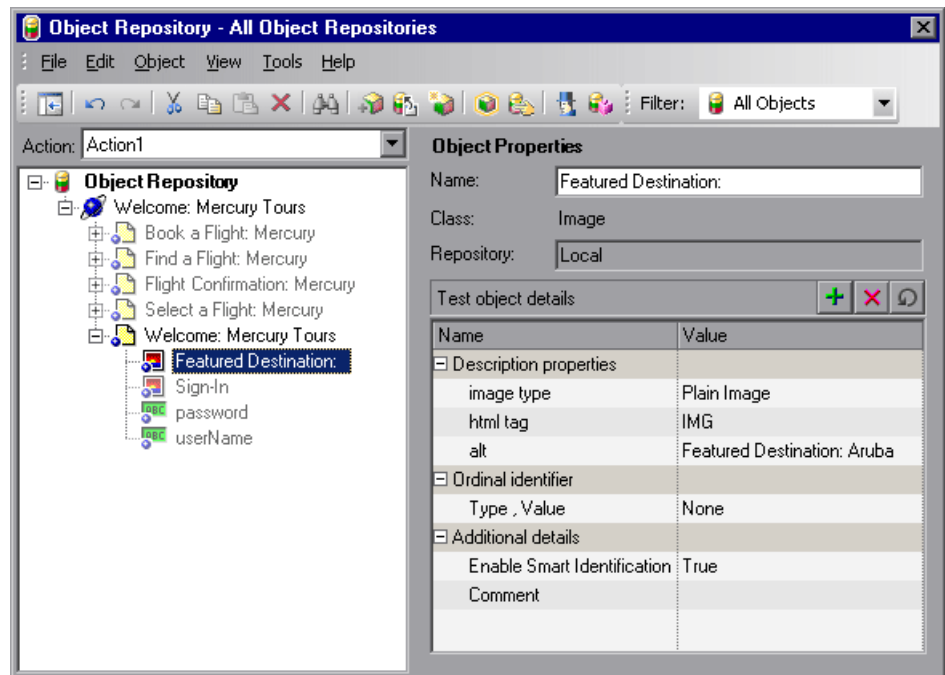


You can choose from the following options:

- **Selected object only (no descendants).** Adds to the object repository the previously selected object's properties and values, without its descendant objects.
- **Default object types.** Adds to the object repository the previously selected object's properties and values, with the properties and values of its descendant objects according to the object types specified by the default filter. You can see which objects are in the default filter by clicking the **Select** button and then clicking the **Default** button.
- **All object types.** Adds to the object repository the previously selected object's properties and values, together with the properties and values of all of its descendant objects.
- **Selected object types.** Adds to the object repository the previously selected object's properties and values, as well as the properties and values of its descendant objects according to the object types and classes you specify in the object filter. You specify the objects and classes in the filter by clicking the **Select** button and selecting the required items in the Select Object Types dialog box. For more information on the Select Object Types dialog box, see "Understanding the Select Object Types Dialog Box" on page 198.

- 4 Select the required option and click **OK** to close the Define Object Filter dialog box and add the specified objects to the object repository according to the selected object filter.
- 5 The Object Repository window is redisplayed, showing the new local objects and their properties and values in the object repository. If you chose to add the objects from the Object Repository Manager, the objects are added to the active shared object repository.

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



You can edit the new test object's details just as you would edit any other object in a local or shared object repository. For more information, see "Viewing and Modifying Test Object Properties" on page 163.

Adding Objects from the Active Screen

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

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 715.

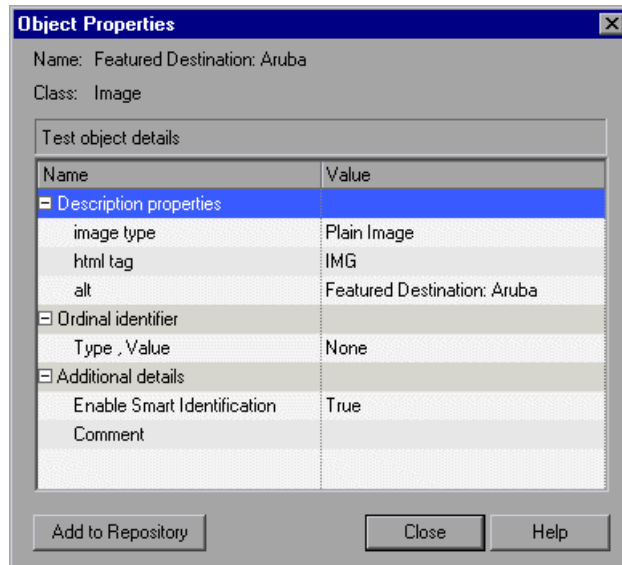
When you add an object to the object repository in one of the ways described in this section, the object is added to the local object repository and can only be used by the current action. If you want to add the object to the shared object repository, so that it can be used in multiple actions, add it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

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



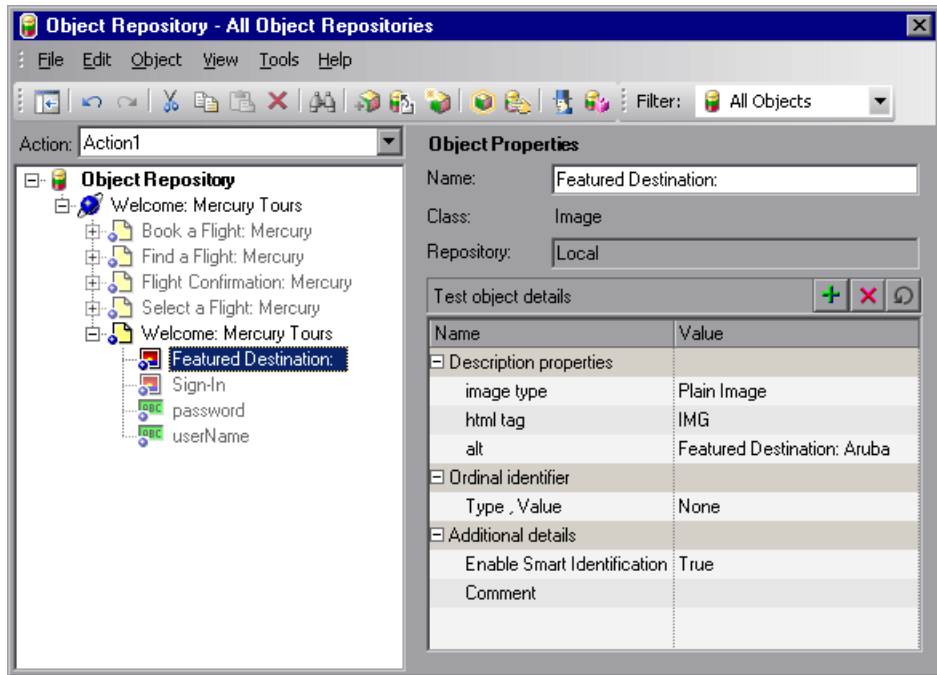
- 1 If the Active Screen is not displayed, choose **View > Active Screen** or click the Active Screen toolbar button to display it.
- 2 Select a step in your test whose Active Screen contains the object that you want to add to the object repository.
- 3 In the Active Screen, right-click the object you want to add and select **View/Add Object**.
- 4 If the location you clicked is associated with more than one object, the Object Selection dialog box opens. Select the object 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 selected object is added to the local object repository for the current action with the default test object properties and values. The **Add to Repository** button changes to **View in Repository**.

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



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

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

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

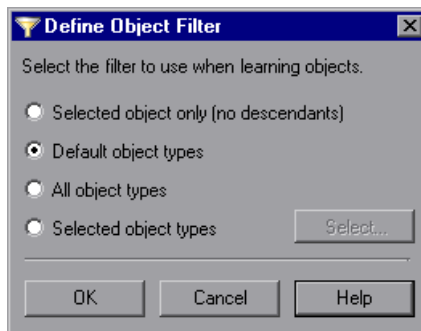
The appropriate dialog box opens, enabling you to configure your preferences for the step you want to insert.

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

Understanding the Define Object Filter Dialog Box

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

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



You can choose from the following options:

- **Selected object only (no descendants)**. Adds to the object repository the previously selected object's properties and values, without its descendant objects.

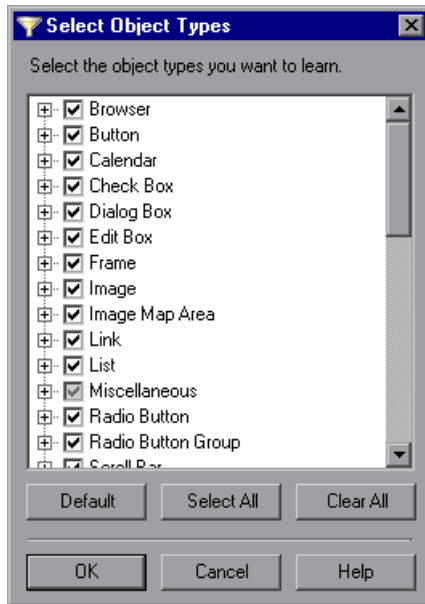
- **Default object types.** Adds to the object repository the previously selected object's properties and values, with the properties and values of its descendant objects according to the object types specified by the default filter. You can see which objects are in the default filter by clicking the **Select** button and then clicking the **Default** button.
- **All object types.** Adds to the object repository the previously selected object's properties and values, together with the properties and values of all of its descendant objects.
- **Selected object types.** Adds to the object repository the previously selected object's properties and values, as well as the properties and values of its descendant objects according to the object types and classes you specify in the object filter. You specify the objects and classes in the filter by clicking the **Select** button and selecting the required items in the Select Object Types dialog box. For more information on the Select Object Types dialog box, see "Understanding the Select Object Types Dialog Box," below.

Understanding the Select Object Types Dialog Box

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

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

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



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

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

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

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

Defining New Test Objects

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

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

When you define a new object in the object repository as described in this section, the object is added to the local object repository and can only be used by the current action. If you want to add the object to the shared object repository so that it can be used in multiple actions, you must add it using the Object Repository Manager. For more information, see Chapter 38, “Managing Object Repositories.”

After you have defined the new test object, if the properties of the object in your application do not match the test object description that you defined, or if an object has been updated in your application, you can update the object description at any time. For more information, see “Updating Test Object Properties from an Object in Your Application” on page 172.

To define a new test object:

- 1 Select the object under which you want to define the new object, according to the correct object hierarchy.
- 2 Click the **Define New Test Object** button or choose **Object > Define New Test Object**. The Define New Test Object dialog box opens.



Define New Test Object

Environment: All

Class: ActiveX

Name: *

Test object details

Name	Value
Description properties	
progid	
Ordinal identifier	
Type, Value	None
Additional details	
Enable Smart Identific...	False
Comment	

Add Close Help

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

Note: The environments included in the **Environment** box correspond to the loaded add-in environments. For more information on loading add-ins, see “Working with QuickTest Add-Ins” on page 809.

- 4 In the **Class** box, select the class of the test object you want to define.
- 5 In the **Name** box, enter a name for the new test object. After you enter a name, the **Test object details** area is enabled.

- 6 In the **Test object details** area, define the properties and values for your test object. The **Test object details** area automatically contains the mandatory properties defined for the object class in the Object Identification dialog box. You can add or remove properties as required, and define values for the properties. For more information, see “Viewing and Modifying Test Object Properties” on page 163.
- 7 Click **Add**. The new test object is added to the local object repository in the selected location.
- 8 Repeat steps 3 to 7 to define additional test objects, or click **Close** to close the Define New Test Object dialog box.

Copying, Pasting, and Moving Objects in the Object Repository

You can copy, paste, and move objects in the local object repository using the Object Repository window, and copy, paste, and move objects both within a shared object repository and between shared object repositories using the Object Repository Manager. You can also copy objects from a shared object repository to the local object repository to modify them locally. For more information, see “Copying an Object to the Local Object Repository” on page 164.



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

You can copy, paste, and move objects in the following ways.

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

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

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

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

To move or copy an object without its child objects:

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

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

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

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

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

To copy objects from one shared object repository to another:

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

To move objects from one shared object repository to another:

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

Guidelines for Copying, Pasting, and Moving Objects

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

- You cannot modify the root node of an object repository.
- If you change the object hierarchy, ensure that the new hierarchy is a valid recorded hierarchy.
- If you paste or move an object to a different hierarchical level, you can choose whether to copy all objects up to the shared parent object (in the message displayed when you perform such an operation).
- In the Object Repository window, when you copy, paste, and move objects from a shared object repository associated with a test, the objects are copied, pasted, or moved to the local object repository of the test.
- If you move an object to its immediate parent, QuickTest creates a copy of the object (renamed with an incremental suffix) and pastes it as a sibling of the original object.
- If you cut or copy an object, and then paste it on its parent object, QuickTest creates copy of the object (renamed with an incremental suffix) and inserts it at the same level as the original object.
- You cannot move an object to any of its descendants.
- You cannot copy or move an object to be a child of a bottom-level object (an object that cannot contain a child object) in the object hierarchy.
- You cannot copy, paste, or move objects that have unmapped repository parameters from a shared object repository to the local object repository. If you copy, paste, or move an object from a shared object repository to the local object repository and the object or one of its parent objects are parameterized using one or more repository parameters, the repository parameter values are converted when you copy, paste, or move the object. For example, if the repository parameter is mapped to a Data Table parameter, the property is parameterized using a Data Table parameter. If the value is a constant value, the property receives the same constant value.

Deleting Objects from the Object Repository

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

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

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

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

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

To delete an object from the object repository:

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



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

Locating Objects

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

Finding Objects in an Object Repository

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

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

Note: You cannot use the Find and Replace dialog box to replace property or object names. You cannot replace property values in a read-only test.

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

- 1 Make sure that the relevant object repository is open (in the Object Repository window or Object Repository Manager).
- 2 Click the **Find & Replace** button or choose **Edit > Find & Replace**. The Find & Replace dialog box opens.



Find & Replace

Find **Find Next**

Object name:

Object type:

Object class:

Property name:

Property value:

Replace **Replace**

New property value: **Replace All**

Options

☐ Match case Direction: ☐ Up

☐ Match whole word ☒ Down

Close **Help**

- 3 Specify one or more criteria by which you want to search for the object, property, or property value:
 - **Object name.** Enter the name or partial name of the object you want to find.
 - **Object type.** Select the type of object you want to find, for example, **Button**.

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

- **Object class.** Select the class of object you want to find, for example, **WebButton**. The classes available depend on the selection you made in the **Object type** box.
 - **Property name.** Specify the name or partial name of the property you want to find.
 - **Property value.** Specify the property value or partial property value you want to find.
- 4 If you specified a property value and want to replace it with a different value, enter the new property value in the **New property value** box.
 - 5 Specify the search parameters, as follows:
 - If you want the search to distinguish between upper and lower case letters, select **Match case**.
 - If you want the search to find only complete words that exactly match the single word you entered, select **Match whole word**.
 - Specify the direction in which you want to search: **Up** or **Down**.

- 6 Perform the find or replace operation in one of the following ways. The search is performed on the entire object repository, starting with the currently selected object and in the direction you specified. To find the next instance, click **Find Next** again.
- To find the specified object, property, or property value, click **Find Next**. The first instance of the searched word is displayed.
 - To individually find and replace each instance of the property value for which you are searching, click **Find Next**. When an instance is found, click **Replace**. The property value is replaced, and the next instance of the property value, if any, is highlighted.
 - To replace all instances of the specified property value with the new property value, click **Replace All**. Instances in shared object repositories that are not editable are not changed.

Highlighting an Object in Your Application

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

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

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

To highlight an object in your application:

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



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

Locating an Object in the Object Repository

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

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

To locate an object in the object repository:

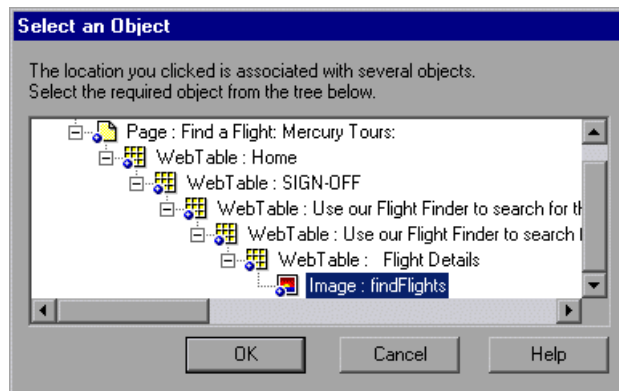
- 1 Make sure your application or Web site is open to the correct window or page.
- 2 Click the **Locate in Repository** button or choose **View > Locate in Repository**. QuickTest is minimized, and your cursor changes to a pointing hand.



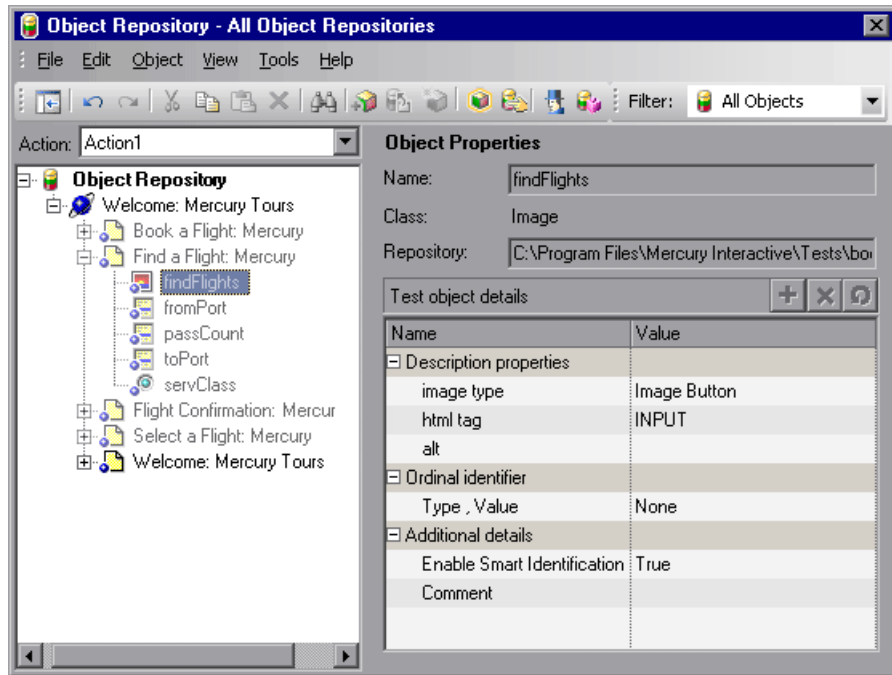
- 3 Use the pointing hand to click on the required object in your application or Web site.

Tip: You can hold the left CTRL key to change the window focus or perform operations such as right-clicking or moving the pointer over an object to display a context menu. If the window containing the object you want to 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. Additionally, if the window containing the object you want to select is minimized, you can display it by holding the left CTRL key, right-clicking the application in the Windows task bar, and choosing **Restore** from the context menu.

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



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



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

Working with Test Objects During a Run Session

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

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

Creating Test Objects During a Run Session

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 or in a user-defined function 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 1005.

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 Keyword View, Expert View, or in a user-defined function.

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

Object(description).SetTOProperty Property, Value

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

Managing Shared Object Repository Associations

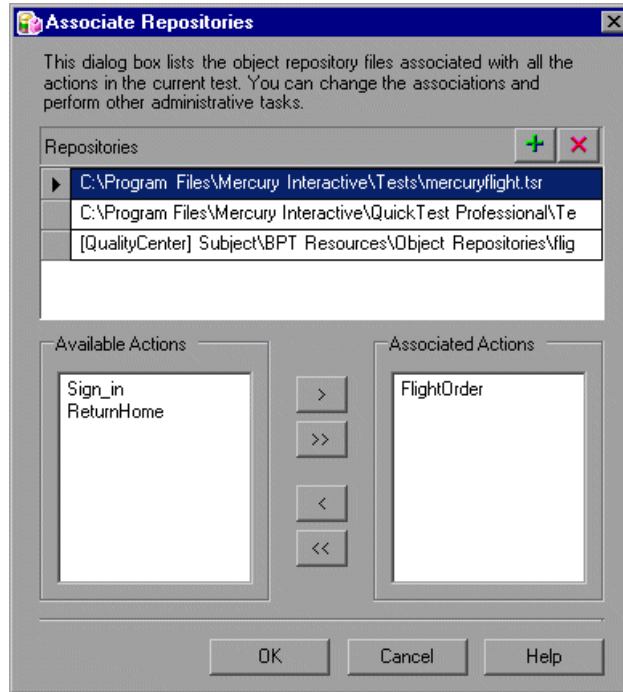
You can manage the shared object repository associations of a selected test using the Associate Repositories dialog box. The Associate Repositories dialog box enables you to associate one or more shared object repositories with one or more actions in a test. You can also remove object repository associations from selected actions, or from all actions in the test. For more information on shared object repository associations, see “Associating Object Repositories with Actions” on page 488.

To manage object repository associations:

- 1** Perform one of the following:
 - Choose **Resources > Associate Repositories**
 - In the Object Repository window, choose **Tools > Associate Repositories**
 - In the Object Repository window, click the **Associate Repositories** button.



The Associate Repositories dialog box opens.



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

Note: If an associated shared object repository is stored in the file system on the network, its path must always be specified as a mapped drive letter (for example, C:), and not as a UNC path (for example, \\myserver\...).



- 2** To add a shared object repository to the list so you can associate it to one or more actions in the current test, click the **Add Repository** button. The Add Shared Object Repository dialog box opens. Navigate to the object repository you want to add, and click **Open** or **OK** (depending on whether you are adding it from the file system or a Quality Center project). The new object repository is displayed at the bottom of the **Repositories** list.
- 3** To modify the name or path of an associated shared object repository, click a shared object repository in the **Repositories** list and then click the browse button to open a file selection dialog box in which you can select a different shared object repository. Alternatively, you can modify the shared object repository name or path directly in the **Repositories** list. The modified shared object repository remains associated with the same actions as the previous shared object repository.

Note: If you are connected to Quality Center, the dialog box that opens is different from the standard file system dialog box. You can switch between the two dialog box versions by clicking the **File System** and **Quality Center** buttons in the Add Shared Object Repository dialog box.

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

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

Note: You cannot define the priorities of the object repositories associated with an action using the Associate Repositories dialog box. You prioritize the object repositories using the Associated Repositories tab of the Action Properties dialog box. For more information, see “Associating Object Repositories with Actions” on page 488.



- 5** To remove an object repository from the list and thereby remove all of its associations to any actions in the current test, select the object repository and click the **Remove Repository** button.
- 6** Click **OK**. The changes you made to the object repository associations are applied. You can view the new associations and change the object repository priorities in the Associated Repositories tab of the Action Properties dialog box. For more information, see “Associating Object Repositories with Actions” on page 488.

Exporting Local Objects to an Object Repository

You can export all of the objects contained in an action’s local object repository to a new shared object repository in the file system or to a Quality Center project (if QuickTest is connected to Quality Center). This enables you to make the local objects accessible to other actions. You export local objects to a new shared object repository using the Object Repository window.

When you export local objects to a shared object repository, the parameters of any parameterized objects are converted to repository parameters using the same name as the source parameter. The default (mapped) value of each repository parameter is the corresponding source parameter. You can modify the mapping used within your action using the Map Repository Parameters dialog box (described in “Mapping Repository Parameter Values” on page 185). For more information on repository parameters, see Chapter 38, “Managing Object Repositories.”

Tip: After you export the local objects, you can use the Object Repository Merge Tool to merge the shared object repository containing the exported objects with another shared object repository. For more information, see Chapter 39, “Merging Shared Object Repositories.”

To export local objects to a new shared object repository:

- 1** Open the test that has the local objects you want to export.
- 2** Make sure that the Object Repository window is open.
- 3** In the Object Repository window, in the **Action** box, choose the action whose local objects you want to export.
- 4** Choose **File > Export Local Objects**. The Export Object Repository dialog box opens.

Note: If you are connected to Quality Center, the dialog box that opens is different from the standard file system dialog box. You can switch between the two dialog box versions by clicking the **File System** and **Quality Center** buttons in the Export Object Repository dialog box.

- 5** Select the location in which to save the file, specify the file or attachment name, and click **Save** or **OK** (depending on whether you are saving it to the file system or a Quality Center project).

The object repository is exported to the specified shared object repository (a file with a **.tsr** extension). You can now use the new shared object repository like any other shared object repository.

7

Working with the Active Screen

The Active Screen provides a snapshot of your application in a recording session and enables you to insert additional steps on the test objects captured in that screen after the recording session.

This chapter describes:	On page:
About Working with the Active Screen	219
Increasing or Decreasing the Active Screen Information Saved with a Test	221
Changing the Active Screen	223
Tips for Improving Active Screen Performance	224

About Working with the Active Screen



The Active Screen provides a snapshot of your application as it appeared when you performed the corresponding step during a recording session. An Active Screen can be captured for every step you record. Additionally, depending on the Active Screen capture options that you used while recording, the page displayed in the Active Screen can contain detailed property information on each object displayed on the page. To view the Active Screen pane, click the **Active Screen** button or choose **View > Active Screen**. For information on setting Active Screen recording options, see “Choosing the Recording Mode” on page 93.

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

You can specify the level at which QuickTest captures and stores information on objects while recording tests. For example, you can instruct QuickTest to capture all properties for all test objects on the captured screen, or only the properties of the recorded objects and their parents. For more information, see “Increasing or Decreasing the Active Screen Information Saved with a Test” on page 221.

If QuickTest captured object information while recording your test, you can use the Active Screen to add these objects to the local object repository. For information on configuring the Active Screen capture settings, see “Setting Active Screen Options” on page 715. For information on adding objects to the object repository from the Active Screen, see “Adding Objects to the Object Repository” on page 189.

When QuickTest creates an Active Screen page for a Web-based application, it stores the path to images and other resources on the page, rather than downloading and storing the images with your test. Therefore, you may need to provide login information to view password-protected resources. For information on accessing password-protected resources in the Active Screen of a Web-based application, see “Accessing Password-Protected Resources in the Active Screen” on page 845.

When working with Web-based applications, you can specify Active Screen display criteria for captured Web pages. For example, you can specify whether QuickTest should load ActiveX controls or Java applets. For more information, see “Web Page Appearance” on page 721.

Active Screen pages for non-Web-based applications are based on a single bitmap capture of the visible part of the application window (or other top-level object), with context sensitive areas representing each object displayed in the Active Screen.

You can choose whether or not to save the content of the Active Screen with your test. Saving the content of the Active Screen with your test is especially useful if you want to be able to edit the saved test directly from the Active Screen. Later, if you need to conserve disk space after you finish editing the test, and you plan to use your test only for test runs, you can save the test without the content of the Active Screen. (Tests without Active Screen files use significantly less disk space.) For more information, see “Increasing or Decreasing the Active Screen Information Saved with a Test”, below.

Increasing or Decreasing the Active Screen Information Saved with a Test

You can decide if and how much information you want to capture and save in the Active Screen. The more information you capture, the easier it is to add steps to your test using the many Active Screen options. However, more captured information also leads to slower recording and editing times. Removing or decreasing Active Screen information can be especially useful for conserving disk space after you have finished designing the test and you are using the test only for test runs.

If you find that the information saved in the Active Screen after recording is not sufficient for your test editing needs, or if you no longer need Active Screen information, and you want to decrease the size of your test, you can change the amount of Active Screen information saved with your test.

To increase or decrease the Active Screen information saved with your test:

- 1** Confirm that the Active Screen capture preference in the Active Screen tab of the Options dialog box is set to capture the amount of information you need. For more information, see “Setting Active Screen Options” on page 715.
- 2** Perform one of the following:
 - Perform an **Update Run Mode** operation to save the required amount of information in the Active Screen for all existing steps. For more information on the **Update Run Mode** options, see “Updating a Test” on page 616.

- Re-record the step(s) containing the object(s) you want to add to the Active Screen by performing one of the following:
 - Select the step after which you want to record your step, position your application to match the selected location in your test, and then begin recording.
 - Place a breakpoint in your test at the step before which you want to add a step and run your test to the breakpoint. This brings your application to the point from which to record the step. For more information on setting breakpoints, see “Setting Breakpoints” on page 597.

To stop saving Active Screen information (and reduce the disk space used by your test):

- 1** Open the relevant test in QuickTest.
- 2** Choose **File > Save As** and clear the **Save Active Screen files** check box.

Note: If you clear this check box, your Active Screen files will not be saved, and you will not be able to edit your test using the options that are normally available from the Active Screen.

- 3** Click **Save** to apply your changes. For more information, see “Saving a Test” on page 105.

Tip: If you need to recover Active Screen files after you save a test without Active Screen files, re-record the necessary steps or use the **Update Run Mode** option to recapture screens for all steps in your test. For more information, see “Updating a Test” on page 616.

Changing the Active Screen

As the content of your application or Web site changes, you can continue to use the Active Screen from tests that you recorded previously. To do this, you update the selected Active Screen display so that you can use the Active Screen to add new steps to your test rather than re-recording steps on new or modified objects.

For example, suppose that one of the pages in your Web site now includes a new object and you want to add a checkpoint that checks this object. You can use the **Change Active Screen** command to replace the page in your Active Screen pane and then proceed to create a checkpoint for this object.

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 pane.
- 2 In the Keyword View, click a step that you want to change. The window or page is displayed in the Active Screen pane.
- 3 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.

Tip: You can also hold the left CTRL key to change the window focus or perform operations such as right-clicking or moving the pointer over an object to display a context menu to choose the new Active Screen display. Additionally, if the window containing the object you want to select is minimized, you can display it by holding the left CTRL key, right-clicking the application in the Windows task bar, and choosing **Restore** from the context menu.

- 5 When a message prompts you to change your current Active Screen display, click **Yes**.

Tips for Improving Active Screen Performance

You can choose from the following Active Screen options to improve performance:

- If you are testing Windows-based applications, you can choose to save all Active Screen information in every step, save information only in certain steps, or to disable Active Screen captures entirely. You set this preference in the Active Screen tab of the Options dialog box. The less information saved, the faster your recording times will be. For more information, see “Setting Active Screen Options” on page 715.
- If you are testing Web-based applications, you can disable screen capture of all steps in the Active Screen. From the Active Screen pane, click **Custom Level** to open the Custom Active Screen Capture Settings dialog box. Select the **Disable Active Screen Capture** option. This will improve recording time. For more information on the Active Screen tab of the Options dialog box, see “Setting Active Screen Options” on page 715.
- If you are testing an application using an external QuickTest add-in, refer to the add-in documentation to determine whether special Active Screen screen capture options exist for that environment.
- When you save a new test, or when you save a test with a new name using **Save As**, you can choose not to save the captured Active Screen files with the test as described in step 2 of the procedure describing how to stop saving Active Screen information on page 222. Tests without Active Screen files use significantly less disk space.

8

Understanding Checkpoints

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

This chapter describes:	On page:
About Understanding Checkpoints	225
Adding Checkpoints to a Test	227
Understanding Types of Checkpoints	228

About Understanding Checkpoints

QuickTest enables you to add checks to your test. 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. By default, the checkpoint name receives the name of the test object on which the checkpoint is being performed. You can choose to specify a different name for the checkpoint or accept the default name.

When you run the test, QuickTest compares the expected results of the checkpoint to the current results. If the results do not match, the checkpoint fails. You can view the results of the checkpoint in the Test Results window.

Tip: You can also use the **CheckProperty** method and the **CheckItemProperty** method to check specific property or item property values. For more information, refer to the *Mercury QuickTest Professional Object Model Reference*.

Note: If you want to retrieve the return value of a checkpoint (a boolean value that indicates whether the checkpoint passed or failed), you must add parentheses around the checkpoint argument in the statement in the Expert View. For example:

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

For more information on Expert View syntax, see “Understanding Basic VBScript Syntax” on page 996.

Adding Checkpoints to a Test

You can add checkpoints during a recording session or while editing your test. It is generally more convenient to define checkpoints after recording the initial test.

To add checkpoints while recording or editing:



Use the commands in the **Insert > Checkpoint** menu, or click the **Insert Checkpoint** button on the toolbar. This displays a menu of checkpoint options that are relevant to the selected step.

To add a checkpoint while editing only:

- Right-click the step where you want to add the checkpoint and choose **Insert Standard Checkpoint**.
- Select the step where you want to add the checkpoint and choose **Insert > Checkpoint > Standard Checkpoint**.
- Right-click any object in the Active Screen and choose the relevant checkpoint option:
 - **Insert Standard Checkpoint**
 - **Insert Bitmap Checkpoint**
 - **Insert Accessibility Checkpoint**

These options can be used to create checkpoints for any object in the Active Screen (even if the object is not part of any step in the Keyword View).

Notes:

If you use the Active Screen 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 715.

Throughout this guide, procedures for creating checkpoints may be described using only one of the above methods. However, you can choose any of the methods described above.

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, and so forth. For example, you can check that a radio button is activated after it is selected or you can check the value of an edit box.

Standard checkpoints are supported for all add-in environments (see “Supported Checkpoints” on page 231).

For more information on standard checkpoints, see Chapter 11, “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 add-in environment (see “Supported Checkpoints” on page 231).

For more information on image checkpoints, see Chapter 11, “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 231).

For more information on bitmap checkpoints, see Chapter 12, “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 all Web and ActiveX add-in environments (see “Supported Checkpoints” on page 231), as well as for many external add-in environments. Table checkpoints are also supported for some list view objects, such as WinListView and VbListView, as well as other list view objects in external add-in environments.

For more information on table checkpoints, see “Checking Tables” on page 233.

- **Text Checkpoint** checks that a text string is displayed in the appropriate place on a Web page or application. For example, suppose a Web page displays the sentence Flight departing from New York to San Francisco. You can create a text checkpoint that checks that the words “New York” are displayed between “Flight departing from” and “to San Francisco”.

Text checkpoints are supported for the Web add-in environment, plus some Web-based add-in environments (see “Supported Checkpoints” on page 231).

For more information on text checkpoints, see Chapter 10, “Checking Text.”

- **Accessibility Checkpoint** identifies areas of your Web site that may not conform to the World Wide Web Consortium (W3C) Web Content Accessibility Guidelines. For example, guideline 1.1 of the W3C Web Content Accessibility Guidelines requires you to provide a text equivalent for every non-text element. You can add an **Alt** property check to check whether objects that require the **Alt** property under this guideline, do in fact have this tag.

Accessibility checkpoints are supported for the Web add-in environment (see “Supported Checkpoints” on page 231).

For more information on accessibility checkpoints, see Chapter 29, “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 add-in environment (see “Supported Checkpoints” on page 231).

For more information on page checkpoints, see Chapter 29, “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 for all add-in environments (see “Supported Checkpoints” on page 231).

For more information on database checkpoints, see Chapter 13, “Checking Databases.”

- **XML Checkpoint** checks the data content of XML documents in XML files or XML documents in Web pages and frames. For more information on XML checkpoints, see Chapter 14, “Checking XML.”

The **XML Checkpoint (Web Page/Frame)** option is supported for the Web add-in environment. The **XML Checkpoint** option is supported for all add-in environments (see “Supported Checkpoints,” below).

Supported Checkpoints

The following table shows the types of checkpoints supported for each add-in environment that is supported by default with the QuickTest Professional installation.

Checkpoint Type	Web	Standard Windows	VB	ActiveX
Standard	S	S	S	S
Image	S	NS	NS	NS
Table	S	S (WinListView)	S (VbListView)	S
Text	S	NS	NS	NS
Bitmap	S	S	S	S
Accessibility	S	NS	NS	NS
XML (Web Page/Frame)	S	NA	NA	NA
XML (From Resource)	S	S	S	S
Page	S	NA	NA	NA
Database	S	S	S	S

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.

9

Checking Tables

You can add table checkpoints to check the content of tables displayed in your application.

This chapter describes:	On page:
About Checking Tables	233
Creating a Table Checkpoint	234
Understanding the Table Checkpoint Properties Dialog Box	238
Checking Table Content	239
Checking Table Properties	249
Modifying a Table Checkpoint	252

About Checking Tables

By adding table checkpoints to your test, you can check the content of tables displayed in your application. For example, you can check that a specified value is displayed in a certain cell. For some environments, you can also check the properties of the table object. For example, you can check that a table has the expected number of rows and columns.

When you run the test, the table checkpoint compares the actual data to the expected data, as defined in the checkpoint. If the results match, the checkpoint passes. You can view the results of the checkpoint in the Test Results window. For more information, see Chapter 24, “Analyzing Test Results.”

Table checkpoints are supported for Web and ActiveX table objects, as well as for table objects in a variety of external add-in environments. Table checkpoints are also supported for some list view objects, such as WinListView and VbListView, as well as other list view objects in external add-in environments.

Creating a Table Checkpoint

You can add a table checkpoint while recording or editing your test. To add a table checkpoint, you use the Table Checkpoint Properties dialog box.

To add a table checkpoint while recording:

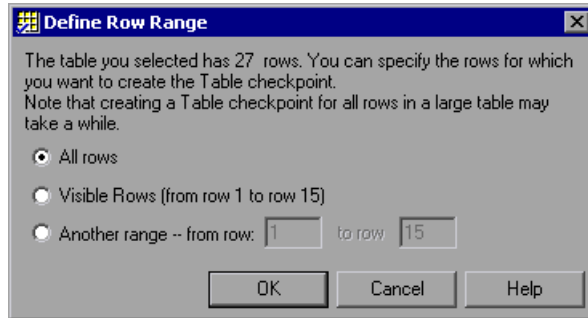


- 1** Choose **Insert > Checkpoint > Standard Checkpoint** or click the **Insert Checkpoint or Output Value** button. The QuickTest window is minimized, and the pointer turns into a pointing hand.

Tip: You can hold the left CTRL key to change the pointing hand into a standard pointer, and then change the window focus or perform operations such as right-clicking or moving the pointer over an object to display a context menu. Additionally, if the window containing the object you want to select is minimized, you can display it by holding the left CTRL key, right-clicking the application in the Windows task bar, and choosing **Restore** from the context menu.

- 2** Click the table you want to check. The Object Selection - Checkpoint Properties dialog box opens.
- 3** Select a table item from the displayed object tree and click **OK**. If the Table Checkpoint Properties dialog box opens, skip to step 4.

Otherwise, for certain objects in certain environments, the Define Row Range dialog box opens.



Select the range of rows you want to include in your checkpoint. You can include:

- **All rows.** Includes all of the rows in the table. Note that capturing all of the data for large table or list view objects may take some time.
- **Visible Rows (from row X to row Y).** Includes only the rows visible on the screen. Note that this option may not be available for some environments or object types.
- **Another range -- from row X to row Y.** You can specify any row range between 1 and the number of rows listed in the table.

Click **OK**. The Define Row Range dialog box closes, and the Table Checkpoint Properties dialog box displays the rows you specified (above the grid area).

- 4 In the Table Checkpoint Properties dialog box, specify the settings for the checkpoint. For more information, see “Understanding the Table Checkpoint Properties Dialog Box” on page 238.

Note: For some environments, the Table Checkpoint Properties dialog box contains two tabs: Table Content and Properties. For other environments, the Table Checkpoint Properties dialog box displays only the options available in the Table Content tab, but does not contain any tabs.

- 5 Click **OK** to close the dialog box. A checkpoint statement is added for the selected object in the Keyword View and Expert View.

To add a table checkpoint while editing:

- 1 Depending on whether the object on which you want to perform a check is already in a step, do one of the following:

- If you have already recorded a step on the object you want to check, right-click the step and choose **Insert Standard Checkpoint**. Alternatively, select the step and choose **Insert > Checkpoint > Standard Checkpoint**.

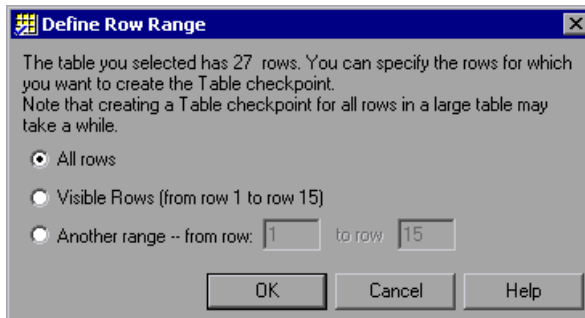


- If you have not recorded a step on the object you want to check, make sure the **Active Screen** button is selected and the Active Screen is visible. Click a step in your test where you want to add a checkpoint. The Active Screen displays the Web page or application screen corresponding to the highlighted step. Right-click the table in the Active Screen and choose **Insert Standard Checkpoint**. The Object Selection - Checkpoint Properties dialog box opens. Select a table item from the displayed object tree and click **OK**.

Note: In some environments, you must have the table open in your application to insert a checkpoint on it.

- 2 If the Table Checkpoint Properties dialog box opens, skip to step 3.

Otherwise, for certain objects in certain environments, the Define Row Range dialog box opens.



Select the range of rows you want to include in your checkpoint. You can include:

- **All rows.** Includes all of the rows in the table. Note that capturing all of the data for large table or list view objects may take some time.
- **Visible Rows (from row X to row Y).** Includes only the rows visible on the screen. Note that this option may not be available for some environments or object types.
- **Another range -- from row X to row Y.** You can specify any row range between 1 and the number of rows listed in the table.

Click **OK**. The Define Row Range dialog box closes, and the Table Checkpoint Properties dialog box displays the rows you specified (above the grid area).

- 3** In the Table Checkpoint Properties dialog box, specify the settings for the checkpoint. For more information, see “Understanding the Table Checkpoint Properties Dialog Box” on page 238.

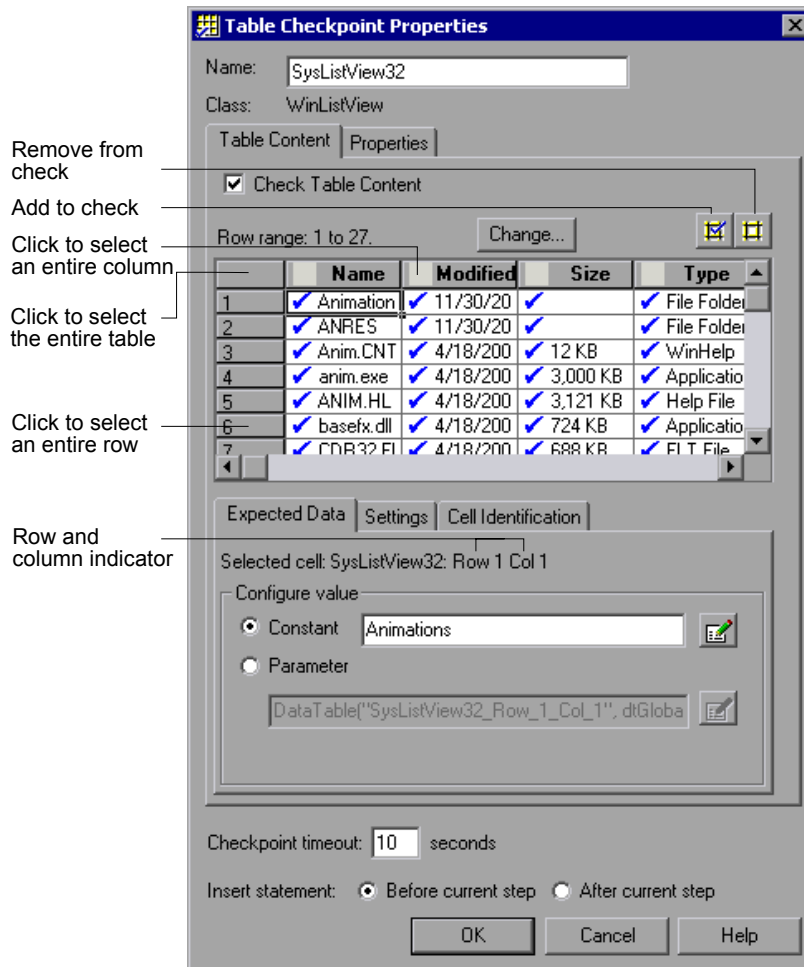
Note: For some environments, the Table Checkpoint Properties dialog box contains two tabs: Table Content and Properties. For other environments, the Table Checkpoint Properties dialog box displays only the options available in the Table Content tab, but does not contain any tabs.

- 4** Click **OK** to close the dialog box. A checkpoint statement is added for the selected object.

Understanding the Table Checkpoint Properties Dialog Box

The Table Checkpoint Properties dialog box enables you to specify which cell contents of your table to check and which verification method and type to use. You can also edit or parameterize the expected data for the cells included in the check.

For some environments, the Table Checkpoint Properties Dialog Box also enables you to check the properties of the object (using the Properties tab), in addition to checking the content (using the Table Content tab).



Note: Some of the options shown in this example are available only in certain environments and only for certain objects.

For information on the options in the Table Content tab (or the entire dialog box if no tab is displayed), see the sections below. For information on the options in the Properties tab, see “Checking Table Properties” on page 249.

Checking Table Content

The Table Checkpoint Properties dialog box enables you to check table content.

Note: If the Table Checkpoint Properties dialog box contains tabs, you use the Table Content tab to check table content.

You can:

- understand and set general table checkpoint options
- specify which cells to check
- specify the expected data (Expected Data tab)
- specify the value type criteria (Settings tab)
- specify how QuickTest should locate the cells to be checked (Cell Identification tab)

Understanding and Setting General Table Checkpoint Options

This section describes the general settings and options displayed in the Table Checkpoint Properties dialog box. Most of the options described in this section are available regardless of whether the Table Checkpoint Properties dialog box contains tabs.

Descriptive Information

The top part of the Table Checkpoint Properties dialog box contains the following options:



The screenshot shows a dialog box with two fields. The 'Name' field contains the text 'True DBGrid Control' and the 'Class' field contains the text 'AcxTable'.

Name	<p>The name that QuickTest assigns to the checkpoint. By default, the checkpoint name is the name of the test object on which the checkpoint is being performed. You can specify a different name for the checkpoint or accept the default name.</p> <p>If you rename the checkpoint, make sure that the name is unique, does not begin or end with a space, and does not contain the following character/combination of characters:</p> <p>" := @@</p>
Class	<p>Specifies the type of object (read-only). This may be a table-type object or a list view-type object.</p>

Tabs (If Available)

If the Table Checkpoint Properties dialog box contains tabs, each tab displays a check box. You can select one or both of these check boxes to specify the type of data to check.



Check Table Content check box	(Table Content tab) Selecting the Check Table Content check box instructs QuickTest to check the content of the table object. (Selected by default.)
Check Properties check box	(Properties tab) Selecting the Check Properties check box instructs QuickTest to check the properties of the table object. (Cleared by default.)

Note: These check boxes are displayed only if the Table Checkpoint Properties dialog box contains tabs. If the Table Checkpoint Properties dialog box does not contain tabs, QuickTest automatically checks table content as defined in the dialog box.

Timeout and Statement Location

The bottom part of the Table Checkpoint Properties dialog box contains the following options:

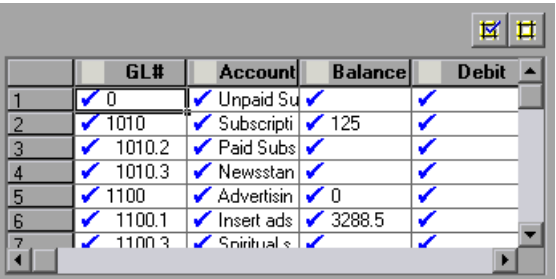
Checkpoint timeout: seconds

Insert statement: ☒ Before current step ☐ After current step

Checkpoint timeout	<p>Specifies the time interval (in seconds) during which QuickTest attempts to perform the checkpoint successfully. QuickTest continues to perform the checkpoint until the checkpoint passes or until the timeout occurs. If the checkpoint does not pass before the timeout occurs, the checkpoint fails.</p> <p>For example, if it takes a long time for data to load in a table, increasing the checkpoint timeout value can help ensure that the data has sufficient time to load. This enables the checkpoint to pass (if the data matches) before the end of the timeout period is reached.</p> <p>If you specify a checkpoint timeout other than 0, and the checkpoint fails, the Test Results window displays information on the checkpoint timeout.</p>
Insert statement	<p>Specifies when to perform the checkpoint in the test. Choose Before current step if you want to check the table content before the highlighted step is performed. Choose After current step if you want to check the table content after the highlighted step is performed.</p> <p>Note: The Insert statement option is available only when adding a new checkpoint while editing an existing test. (This option is not available during recording.)</p>

Specifying Which Cells to Check

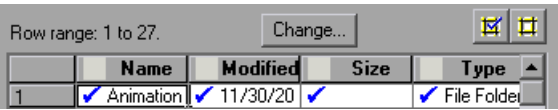
The grid area of the Table Checkpoint Properties dialog box represents the cells in the table. The column header names are captured from the table you selected for your checkpoint.



	GL#	Account	Balance	Debit
1	✓ 0	✓ Unpaid Su	✓	✓
2	✓ 1010	✓ Subscripti	✓ 125	✓
3	✓ 1010.2	✓ Paid Subs	✓	✓
4	✓ 1010.3	✓ Newsstan	✓	✓
5	✓ 1100	✓ Advertisin	✓ 0	✓
6	✓ 1100.1	✓ Insert ads	✓ 3288.5	✓
7	✓ 1100.3	✓ Spiritual s	✓	✓

Tip: You can change the column widths and row heights of the grid by dragging the column and row header dividers.



Note: Some environments and objects support selecting a row range. This enables you to specify which rows are displayed in the grid area. If row range selection is supported, the row range you specify when creating the checkpoint is displayed above the grid:



	Name	Modified	Size	Type
1	✓ Animation	✓ 11/30/20	✓	✓ File Folder

Clicking the **Change** button enables you to modify the row range. For more information, see “Modifying a Table Checkpoint” on page 252.

When you create a new table checkpoint, all cells contain a blue check mark, indicating they are all selected for verification. You can instruct QuickTest to check the entire table, specific rows, specific columns, or specific cells. QuickTest checks only cells containing a check mark.

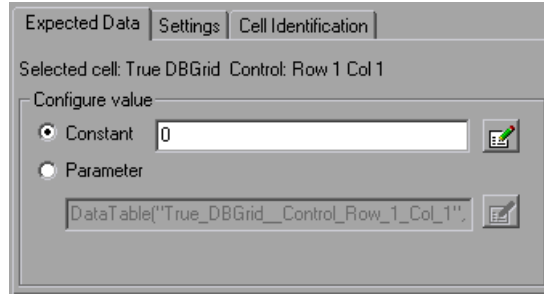
To:	Do this:
Add a single cell to or remove it from the check	Double-click the cell
Add an entire row to or remove it from the check	Double-click the row header
Add an entire column to or remove it from the check	Double-click the column header
Add all cells to or remove all cells from the check	Double-click the top-left corner of the grid
Add a range of cells to the check	Select the cells to add to the check and click the Add to Check button 
Remove a range of cells from the check	Select the cells to remove from the check and click the Remove from Check button 

Notes:

- Double-clicking on the grid toggles the settings for all selected cells. Therefore, if you double-click a row header, a column header, or the top left corner of the grid, any cells that were previously included in the check are removed from it, and any cells that were not previously included in the check are added to it.
 - When more than one cell is selected, the options in the Expected Data tab are disabled.
-

Specifying the Expected Data

The Expected Data tab displays options for setting the expected value of the selected cell in the table.



You can modify the value of a cell or you can parameterize it to use a value from an external source, such as the Data Table or an environment variable. During the run session, QuickTest compares the value specified in this tab with the actual value that it finds during the run session. If the expected value and the actual value do not match, the checkpoint fails.

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

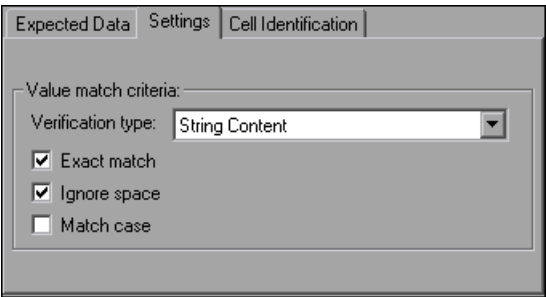
The Expected Data tab includes the following:

Selected cell	Indicates the table name and the row and column numbers of the selected cell.
Configure value	Enables you to set the expected value of the cell as a constant or parameter. For more information on modifying values, see “Setting Values in the Configure Value Area” on page 347.

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

Specifying the Value Type Criteria

The Settings tab includes options that determine how the actual cell values are compared with the expected cell values. The settings in this tab apply to all selected cells.



The default setting is to treat cell values as strings and to check for the exact text, while ignoring spaces.

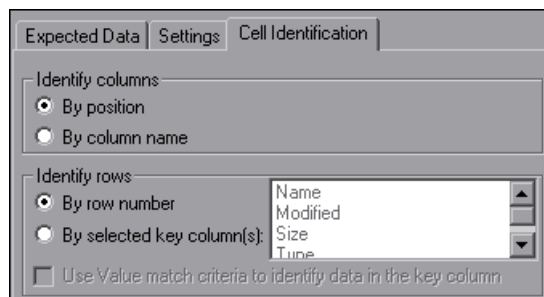
The Settings tab includes the following options:

Option	Description
Verification type	<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 table data is compared against the range that you defined and not against a specific expected value.


Option	Description
Exact match	Default. Checks that the exact text, and no other text, is displayed in the cell. Clear this check box if you want to check that a value is displayed in a cell as part of the contents of the cell. Note: QuickTest displays this option only when String Content is selected as the Verification type .
Ignore space	Default. Ignores spaces in the captured content when performing the check. The presence or absence of spaces does not affect the outcome of the check. Note: QuickTest displays this option only when String Content is selected as the Verification type .
Match case	Conducts a case sensitive search. Note: QuickTest displays this option only when String Content is selected as the Verification type .
Min / Max	Specifies the numeric range against which the content of the cell is compared. The range values can be any real number. Note: QuickTest displays this option only when Numeric Range is selected as the Verification type .

Specifying the Cell Identification Settings

The settings in the Cell Identification tab determine how QuickTest locates the cells to be checked. The settings in this tab apply to all selected cells.

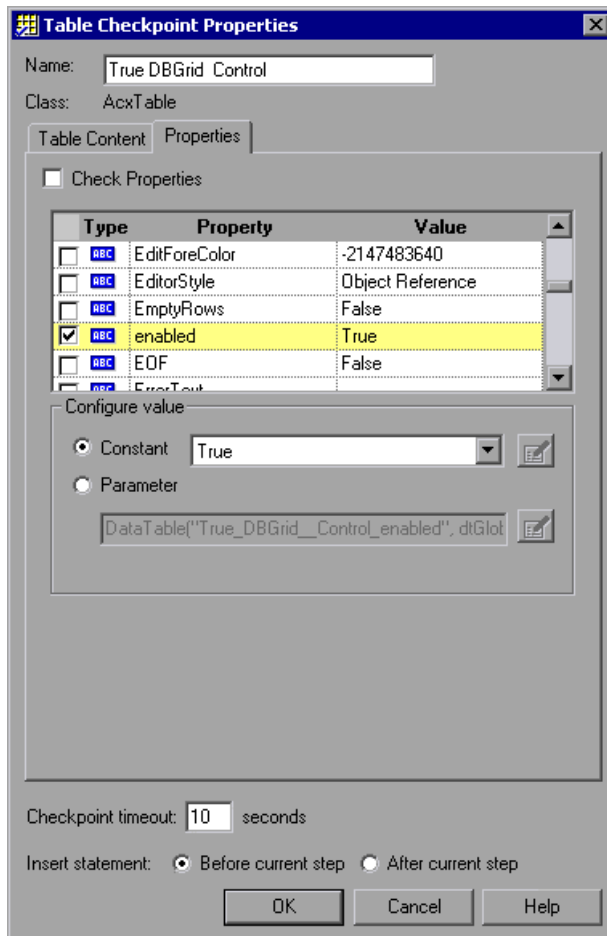


The Cell Identification tab includes the following options:

Identify columns	<p>Specifies the location of the column (in your actual table) containing the cell(s) to which you want to compare the expected data.</p> <ul style="list-style-type: none"> • By position. Default. Locates cells according to the column position. A shift in the position of the columns within the table results in a mismatch. • By column name. Locates cells according to the column name. A shift in the position of the columns within the table does not result in a mismatch. (Enabled only when the table contains more than one column.)
Identify rows	<p>Specifies the location of the row (in your actual table) containing the cell(s) 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 table results in a mismatch. • By selected key column(s). Locates the row(s) containing the cells to be checked by matching the value of the cell whose column was previously selected as a key column. A shift in the position of the row(s) does not result in a mismatch. If more than one row is identified, QuickTest checks the first matching row. You can use more than one key column to uniquely identify any row. <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>

Checking Table Properties

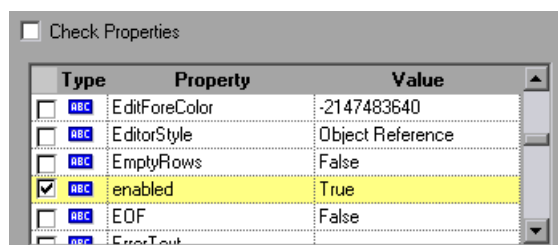
For certain environments, you can specify which table (or grid) properties you want to check. By default, when you create a table checkpoint on an object, QuickTest captures all the object's properties, but does not select any properties to check.



Note: For information on general table checkpoint options, such as **Name** and **Checkpoint timeout**, see “Understanding and Setting General Table Checkpoint Options” on page 240.






Selecting Properties to Check

When you create a table checkpoint, the Properties pane displays the table object’s default properties, including the properties, their values, and their types.



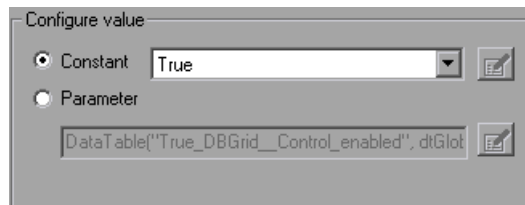
You instruct QuickTest to perform a properties check by selecting the **Check Properties** check box. (This check box is cleared by default.)

The Properties pane for the object contains the following:

Check box	<p>For each object class, QuickTest recommends default property checks. You can accept the default checks or modify them accordingly.</p> <ul style="list-style-type: none"> • To check a property, select the corresponding check box. • To remove a property from the check, clear the corresponding check box
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 or action 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 347.

Editing the Expected Value of a Table Property

The **Configure value** area enables you to define the expected value of the property as a **Constant** or a **Parameter**.



For information on modifying property values, see “Setting Values in the Configure Value Area” on page 347.

Modifying a Table Checkpoint

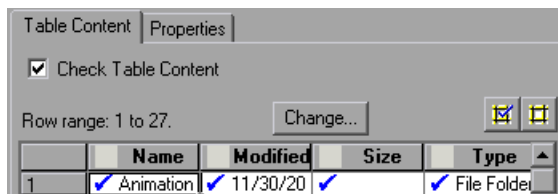
You can change the expected data, settings and cell identification options for an existing table checkpoint.

To modify the settings of the table checkpoint:

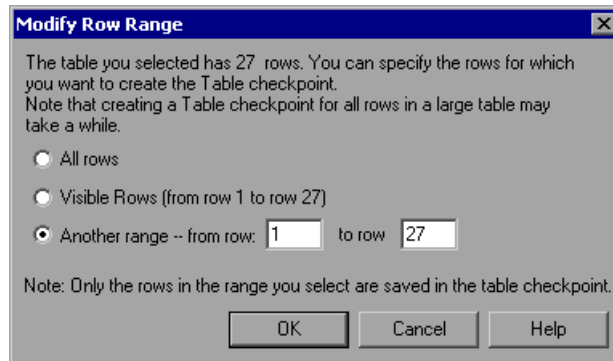
- 1 In the Keyword View or Expert View, right-click the table checkpoint that you want to modify and select **Checkpoint Properties**. Alternatively, select the step containing the checkpoint and choose **Edit > Step Properties > Checkpoint Properties**. The Table Checkpoint Properties dialog box opens.
- 2 Modify the settings as described in “Understanding the Table Checkpoint Properties Dialog Box” on page 238.

To modify the number of rows in an existing table checkpoint:

- 1 Open the application containing the table or list view object you want to check and display the object in the application.
- 2 In the Keyword View or Expert View, right-click the table checkpoint that you want to modify and select **Checkpoint Properties**. Alternatively, select the step containing the checkpoint and choose **Edit > Step Properties > Checkpoint Properties**. The Table Checkpoint Properties dialog box opens, displaying the currently selected row range.



- 3 In the Table Content tab, click the **Change** button at the top of the dialog box (above the grid area). The Modify Row Range dialog box opens.



- 4 Select the range of rows you want to include in your checkpoint. You can include all the rows, only the visible rows, or another range that you specify.

Note: The **Visible Rows** option may not be available for some environments or object types.

- 5 Click **OK**. The Modify Row Range dialog box closes, and the Table Checkpoint Properties dialog box displays the rows you specified in the Modify Row Range dialog box.
 - If your modified row range includes new rows, QuickTest captures the current values of the new rows from the open application.
 - If your modified row range includes some or all of the rows that were already included in the checkpoint, the expected values of those cells are not changed. This enables you to modify the row range without losing parameterization, regular expressions, or other changes you may have made to the expected cell values in your checkpoint.

Therefore, you cannot use the Modify Row Range dialog box to update the expected values of an existing table checkpoint. To update the expected values of your checkpoint, use the **Update Run** option. For more information, see “Updating a Test” on page 616.

- If your modified row range excludes some or all of the rows that were previously included in your checkpoint, those rows (and any modifications you made to the expected values) are deleted from the checkpoint.

10

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:	On page:
About Checking Text	256
Creating a Text Checkpoint	257
Understanding the Text Checkpoint Properties Dialog Box	259
Modifying a Text Checkpoint	269
Creating a Standard Checkpoint for Checking Text	269

About Checking Text

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

- **Text Checkpoint.** Enables you to check that the text is displayed in a Web page, according to specified criteria. Text checkpoints are supported for Web-based applications (as long as the Web Add-in is loaded). Text checkpoints are also supported for various external QuickTest Professional add-ins. Refer to your add-in documentation for details.

Note: QuickTest no longer supports the creation of text checkpoints for checking non-Web-based applications. QuickTest also no longer supports the creation of text area checkpoints. However, if you used an earlier version of QuickTest to create text area checkpoints or text checkpoints, you can still run and edit these text/text area checkpoints (as long as the relevant QuickTest add-in is loaded).

- **Standard Checkpoint.** Enables you to check the **text** property of an object. You can use standard checkpoints to check text in Windows-based and other types of applications (including Web-based applications). For more information on standard checkpoints, see “Creating Standard Checkpoints” on page 274.

Creating a Text Checkpoint

You can add a text checkpoint while recording or editing steps in a Web application.

Notes:

Text checkpoints are also supported for some external QuickTest Professional add-ins. Refer to your add-in documentation for details.

Text checkpoints are not supported for Windows-based objects. For more information, refer to the *QuickTest Professional Readme*.

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 **Insert Checkpoint or Output Value** button and choose **Text Checkpoint**.



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

Tip: Hold the left CTRL key to change the window focus or perform operations such as right-clicking or moving the pointer over an object to display a context menu. If the window containing the object you want to select is minimized, you can display it by holding the left CTRL key, right-clicking the application in the Windows task bar, and choosing **Restore** from the context menu.

- 3 Click the text string for which you want to create the checkpoint. The Text Checkpoint Properties dialog box opens.
- 4 Specify the checkpoint settings. For more information, see “Understanding the Text Checkpoint Properties Dialog Box” on page 259.
- 5 Click **OK** to close the dialog box. A checkpoint statement is added for the selected object.

To add a text checkpoint while editing a test:

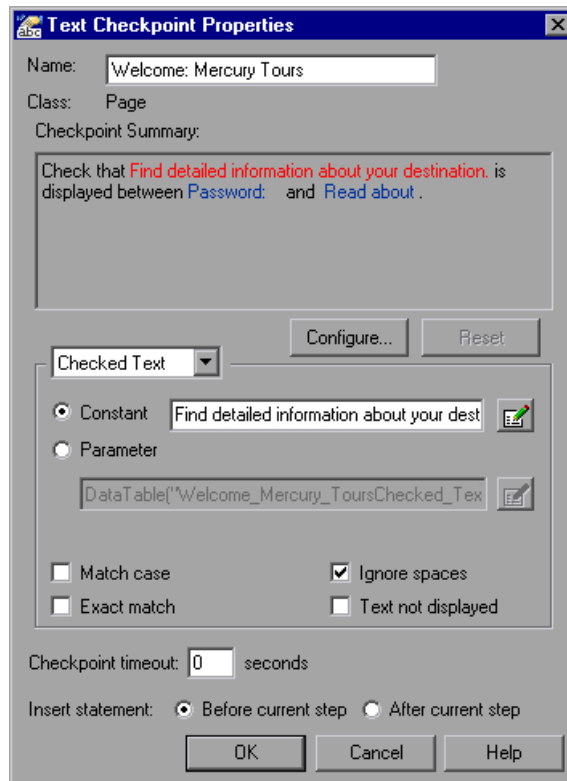


- 1** Make sure the **Active Screen** button is selected.
- 2** Click the step where you want to add a checkpoint. The Active Screen displays the page or screen corresponding to the highlighted step.
- 3** Highlight a text string on the Active Screen.
- 4** Right-click the text string and choose **Insert Text Checkpoint**. The Text Checkpoint Properties dialog box opens.
- 5** Specify the settings for the checkpoint. For more information, see “Understanding the Text Checkpoint Properties Dialog Box” on page 259.
- 6** Click **OK** to close the dialog box. A checkpoint statement is added for the selected object.

Understanding the Text Checkpoint Properties Dialog Box

In the Text 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. 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. It displays the text you selected when creating the checkpoint, plus the text before and after it. QuickTest automatically displays the Checked Text in red and the text before and after the Checked Text in blue.

In the Text Checkpoint Properties dialog box you can:

- view and specify general text checkpoint options. For more information, see “Understanding and Setting General Text Checkpoint Information” on page 261.
- designate parts of the captured string as **Checked Text** and other parts as **Text Before** and **Text After** by clicking the **Configure** button. For more information, see “Configuring the Text Selection” on page 261.
- set parameterization and other preferences for each of the string elements in your checkpoint by selecting the string element type (**Checked Text** / **Text Before** / **Text After**) from the list box and selecting your preferences. For more information, see “Setting Options for Checked Text” on page 264, “Setting Options for Text Displayed Before the Checked Text” on page 265, and “Setting Options for Text Displayed After the Checked Text” on page 266.
- specify when QuickTest should perform the checkpoint by specifying the timeout and location of the checkpoint. For more information, see “Setting Checkpoint Timeout and Statement Location Options” on page 268.

Understanding and Setting General Text Checkpoint Information

The top part of the Text Checkpoint Properties dialog box contains the following options:

Name:

Class:

Name	<p>The name that QuickTest assigns to the checkpoint. By default, the checkpoint name is the name of the test object on which the checkpoint is being performed. You can specify a different name for the checkpoint or accept the default name.</p> <p>If you rename the checkpoint, make sure that the name is unique, does not begin or end with a space, and does not contain the following character/combination of characters:</p> <p>" := @@</p>
Class	Specifies the type of object (read-only).

Configuring the Text Selection

You can view and modify the text selection displayed in the Checkpoint Summary pane.

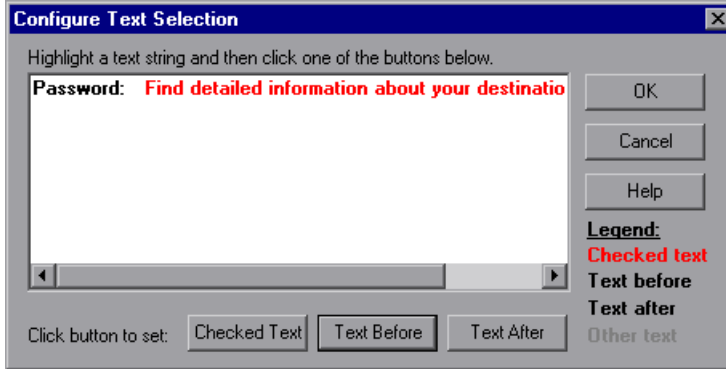
Checkpoint Summary:

Check that **Find detailed information about your destination.** is displayed between **Password:** and **Read about .**

The Checkpoint Summary pane contains the following options.

Option	Description
Configure	Opens the Configure Text Selection dialog box, where you can specify the checked text, the text before (if any), and the text after (if any).
Reset	Resets the text selection to the previous configuration.

To designate **Checked Text**, **Text Before**, and **Text After**, you use the Configure Text Selection dialog box (opened by clicking the **Configure** button). The Configure Text Selection dialog box displays the text you captured when creating the text checkpoint, as well as text before and after the selected text. QuickTest displays the checked text in red and the text before and after it in black (as indicated in the **Legend** displayed in the dialog box).



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 choose a larger text selection in your Web page.

You can modify 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.

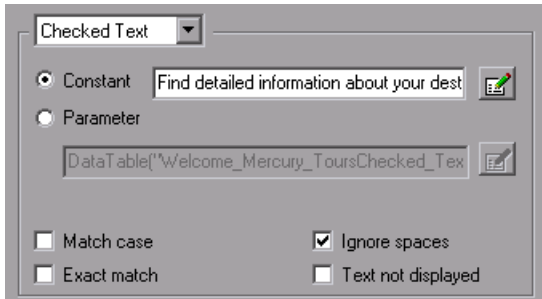
When you close the Configure Text Selection dialog box, the Checkpoint Summary pane displays the new text selection configuration.

Setting Options for the Text to be Checked

The middle area of the Text Checkpoint Properties dialog box enables you to set options for the checked text, text before, and text after, as described in the following sections.

Setting Options for Checked Text

You set options for the checked text by choosing **Checked Text** from the list box. In the Checked Text area, you can indicate whether you want the checked text to be a constant or a parameter, and you can set the criteria for a successful match.



You can choose from the following options for the checked text:

- **Constant.** Default. Sets the expected value of the checked text as a constant. For information on modifying values, see “Setting Values in the Configure Value Area” on page 347.

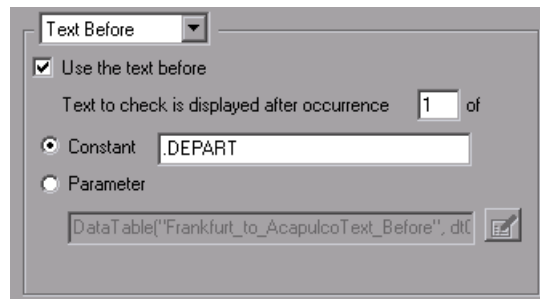
Tip: The **Constant** box displays the checked text. You can change the checked text by typing in the **Constant** box or by using the Configure Text Selection dialog box.

- **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 347.
- **Match case.** Conducts a case-sensitive check.
- **Exact match.** Checks for the exact expected text. For example, if you create a checkpoint with the following description, **Check that New York is displayed between Flight departing from and to San Francisco**, and select **Exact match**, if the actual text is **New York City**, the checkpoint fails. If you do not select **Exact match**, the checkpoint passes because the expected text is contained within the actual text.

- **Ignore spaces.** Ignores spaces in the captured text when performing the check. The presence or absence of spaces does not affect the outcome of the check.
- **Text not displayed.** Checks that the text string is not displayed. For example, if you create a checkpoint with the following description, Check that New York is displayed between Flight departing from and to San Francisco, and select **Text not displayed**, QuickTest checks that the text New York is not displayed.

Setting Options for Text Displayed Before the Checked Text

You set options for the text displayed before the checked text by choosing **Text Before** from the list box. In the Text Before area, you can set the text before the checked text as a constant or a parameter.



You can choose from the following options when setting the text displayed before the checked text:

- **Use the text before.** Checks the text before the checked text. To ignore this text, clear this check box.
- **Text to check is displayed after occurrence.** Checks that the checked text is displayed after the specified text.

If the identical text string you specify is displayed more than once on the page, you can specify the occurrence of the string to which you are referring.

If you accept the default text that QuickTest recommends, the number in the dialog box will be correct. If you modify the text, confirm that the occurrence number is accurate.

If you choose a non-unique text string, change the occurrence number appropriately. For example, if you want to check that the words **Mercury Tours** are displayed after the fourth occurrence of the word **the**, enter **4** in the **Text to check is displayed after occurrence** box.

- **Constant.** Default. Sets the expected value of the text before the checked text as a constant. For information on modifying values, see “Setting Values in the Configure Value Area” on page 347.

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

Tip: The **Constant** box displays the text before the checked text. You can change the text by typing in the **Constant** box or by using the Configure Text Selection dialog box.

- **Parameter.** Sets the expected value of the text before the checked text as a parameter. For information on modifying values, see “Setting Values in the Configure Value Area” on page 347.

Setting Options for Text Displayed After the Checked Text

You set options for or the text displayed after the checked text by choosing **Text After** from the list box. In the Text After area, you can set the text after the checked text as a constant or a parameter.

The screenshot shows a configuration window for 'Text After'. It includes a dropdown menu set to 'Text After', a checked checkbox for 'Use the text after', and a field for 'Text to check is displayed before occurrence' with the value '1'. Below this, the 'Constant' radio button is selected, and the text '03/09/2004' is entered in the corresponding text box. The 'Parameter' radio button is unselected. At the bottom, there is a text box containing the code 'DataTable(\"Frankfurt_to_AcapulcoText_After\", dtGlt)' and a small icon to its right.

You can choose from the following options when setting the text displayed after the checked text:

- **Use the text after.** Checks the text after the checked text. To ignore this text, clear this check box.
- **Text to check is displayed before occurrence.** Checks that the checked text is displayed before the specified text. If the identical text string you specify is displayed more than once on the page, you can specify to which occurrence of the string you are referring.

If you accept the default text that QuickTest recommends, the number in the dialog box will be correct. If you modify the text, confirm that the occurrence number is also accurate.

If you choose a non-unique text string, change the occurrence number appropriately. For example, if you want to check that the words **Mercury Tours** are displayed before the fourth occurrence of the word **the**, enter 4 in the **Text to check is displayed before occurrence** box.

- **Constant.** Default. Sets the expected value of the text after the checked text as a constant. For information on modifying values, see “Setting Values in the Configure Value Area” on page 347.

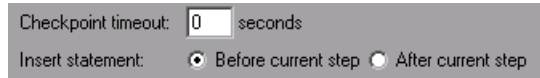
If you modify the text, whenever possible, use a string that is unique within the object so that the occurrence number is 1.

Tip: The **Constant** box displays the text after the checked text. You can change the text by typing in the **Constant** box or by using the Configure Text Selection dialog box.

- **Parameter.** Sets the expected value of the text after the checked text as a parameter. For information on modifying values, see “Setting Values in the Configure Value Area” on page 347.

Setting Checkpoint Timeout and Statement Location Options

You can specify the time interval during which QuickTest attempts to perform the checkpoint successfully by modifying the selections in the bottom part of the Text Checkpoint Properties dialog box. You can also specify when to perform the checkpoint.



- **Checkpoint timeout.** Specifies the time interval (in seconds) during which QuickTest attempts to perform the checkpoint successfully. QuickTest continues to perform the checkpoint until it passes or until the timeout occurs. If the checkpoint does not pass before the timeout occurs, the checkpoint fails.

For example, suppose it takes some time for an object to achieve an expected state. Increasing the checkpoint timeout value in this case can help ensure that the object has sufficient time to achieve that state, enabling the checkpoint to pass (if the data matches) before the maximum timeout is reached.

If you specify a checkpoint timeout other than **0**, and the checkpoint fails, the Test Results window displays information on the checkpoint timeout.

- **Insert statement.** Specifies when to perform the checkpoint. 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 during recording, or when modifying an existing checkpoint. It is available only when adding a new text checkpoint to an existing test while editing.

Modifying a Text Checkpoint

You can modify an existing text checkpoint.

To modify a text checkpoint:

- 1 In the Keyword View or Expert View, right-click the checkpoint that you want to modify and select **Checkpoint Properties**. Alternatively, select the step containing the checkpoint and choose **Edit > Step Properties > Checkpoint Properties**. The Text Checkpoint Properties dialog box opens.
- 2 Modify the settings. For more information, see “Understanding the Text Checkpoint Properties Dialog Box” on page 259.

Creating a Standard Checkpoint for Checking Text

You can check the text property of an object in Windows-based and other types of applications (including Web-based applications) by using a standard checkpoint.

To add a standard checkpoint for checking text while recording:



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

Tip: Hold the left CTRL key to change the window focus or perform operations such as right-clicking or moving the pointer over an object to display a context menu. If the window containing the object you want to select is minimized, you can display it by holding the left CTRL key, right-clicking the application in the Windows task bar, and choosing **Restore** from the context menu.

- 2 Click the object whose text you want to check. The Select an Object dialog box opens.

- 3 Select the item you want to check from the displayed object tree.
- 4 Click **OK**. The Checkpoint Properties dialog box opens.
- 5 In the **Name** box, either accept the name that QuickTest assigns to the checkpoint or specify another name for it. By default, the checkpoint name is the name of the test object on which the checkpoint is being performed.

If you rename the checkpoint, make sure that the name is unique, does not begin or end with a space, and does not contain the following character/combination of characters:

" := @@

Note: The **Class** area displays the type of test object on which the checkpoint is being performed.

- 6 Select the **text** property.
- 7 If necessary, edit the **text** value you want QuickTest to check. Note that you can parameterize this value.
- 8 If you want to check only text, clear the other check boxes in the dialog box.
- 9 Click **OK** to close the dialog box. A checkpoint statement is added for the selected object.

To add a standard checkpoint for checking text while editing:

- 1 Right-click the step for the object whose text you want to check, and choose **Insert Checkpoint**. The Checkpoint Properties dialog box opens.
- 2 In the **Name** box, either accept the name that QuickTest assigns to the checkpoint or specify another name for it. By default, the checkpoint name is the name of the test object on which the checkpoint is being performed.

If you rename the checkpoint, make sure that the name is unique, does not begin or end with a space, and does not contain the following character/combination of characters:

" := @@

Note: The **Class** area displays the type of test object on which the checkpoint is being performed.

- 3** Select the **text** property.
- 4** If necessary, edit the text value you want QuickTest to check. Note that you can parameterize this value.
- 5** If you want to check only text, clear the other check boxes in the dialog box.
- 6** Click **OK** to close the dialog box. A checkpoint statement is added for the selected object.

For more information on creating standard checkpoints, see Chapter 11, “Checking Object Property Values.”

11

Checking Object Property Values

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

This chapter describes:	On page:
About Checking Object Property Values	273
Creating Standard Checkpoints	274
Understanding the Checkpoint Properties Dialog Box	277
Understanding the Image Checkpoint Properties Dialog Box	281
Modifying Checkpoints	283

About Checking Object Property Values

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 while recording or editing the test. To set the options for a standard checkpoint, you use the Checkpoint Properties dialog box.

To add a standard checkpoint while recording:















- 1** Choose **Insert > Checkpoint > Standard Checkpoint**, or click the **Insert Checkpoint or Output Value** toolbar button.

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

Note: If the object you want to check can only be displayed by performing an event (such as right-clicking or moving the pointer over an object to display a context menu), hold the left CTRL key. You can also use the left CTRL key to change the window focus. The pointing hand temporarily turns into a standard pointer and you can perform the event. When the object you want to check is displayed, release the left CTRL key. The pointer 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
	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 277.
- 6** Click **OK** to close the dialog box. A checkpoint statement is added for the selected object in the Keyword View and Expert View.

To add a standard checkpoint while editing:

- 1** Perform one of the following:
 - Right-click the step on which you want to perform a checkpoint and choose **Insert Standard Checkpoint**.
 - Select the step where you want to add the checkpoint and choose **Insert > Checkpoint > Standard Checkpoint**.
 - Right-click any object in the Active Screen 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 ABC icon indicates that the value of the property to check is a constant.

This icon indicates that the value of the property to check is a Data Table parameter.

The selected check box indicates that this property will be checked.

Checkpoint Properties

Name:

Class: WebEdit

Type	Property	Value
<input type="checkbox"/> ABC	disabled	0
<input checked="" type="checkbox"/> ABC	html tag	INPUT
<input checked="" type="checkbox"/>	innertext	<userName_innertext>
<input checked="" type="checkbox"/> ABC	name	userName
<input type="checkbox"/> ABC	readonly	0
<input checked="" type="checkbox"/> ABC	type	text

Configure value:

☐ Constant

☒ Parameter

Checkpoint timeout: seconds

Insert statement: ☒ Before current step ☐ After current step

OK Cancel Help

The dialog box described above is used to configure most standard checkpoints. Certain standard checkpoint types, however, employ different dialog boxes, as follows:

For information on the Dialog Box for:	See:
Image checkpoint properties	“Understanding the Image Checkpoint Properties Dialog Box” on page 281
Page checkpoint properties	“Understanding the Page Checkpoint Properties Dialog Box” on page 831
Table checkpoint properties	“Understanding the Table Checkpoint Properties Dialog Box” on page 238






Identifying the Checkpoint

The top part of the dialog box displays information on the checkpoint:

Information	Description
Name	<p>The name that QuickTest assigns to the checkpoint. By default, the checkpoint name is the name of the test object on which the checkpoint is being performed. You can specify a different name for the checkpoint or accept the default name.</p> <p>If you rename the checkpoint, make sure that the name is unique, does not begin or end with a space, and does not contain the following character/combination of characters:</p> <p>" := @@</p>
Class	The type of object. In this example, the WebEdit class indicates that the object is an edit box.

Selecting the Object Property to Check

The properties for the object are listed in the Properties pane of the dialog box. The pane includes the properties, their values, and their types:

Pane Element	Description
Check box	For each object class, QuickTest recommends default property checks. You can accept the default checks or modify them accordingly. To check a property, select the corresponding check box. To exclude a property check, clear the corresponding check box.
Type	The  icon indicates that the value of the property is currently a constant. The  icon indicates that the value of the property is currently a test or action parameter. The  icon indicates that the value of the property is currently a Data Table parameter. The  icon indicates that the value of the property is currently an environment variable parameter. The  icon indicates that the value of the property is currently a random number parameter.
Property	The name of the property.
Value	The expected value of the property. For more information on modifying the value of a property, see “Setting Values in the Configure Value Area” on page 347.

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 347.

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 help ensure that the object has sufficient time to achieve that state, enabling the checkpoint to pass (if the data matches) before the maximum timeout is reached.

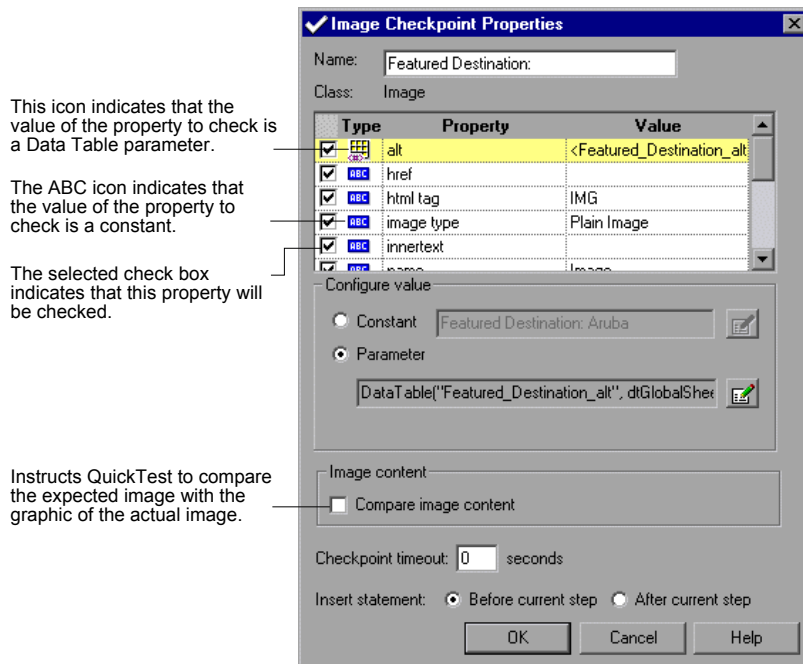
If you specify a checkpoint timeout other than **0**, and the checkpoint fails, the Test Results window displays information on the checkpoint timeout.

- **Insert statement.** Specifies when to perform the checkpoint in the test. 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 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 on the image to check:

Information	Description
Name	<p>The name that QuickTest assigns to the checkpoint. By default, the checkpoint name is the name of the test object on which the checkpoint is being performed. You can specify a different name for the checkpoint or accept the default name.</p> <p>If you rename the checkpoint, make sure that the name is unique, does not begin or end with a space, and does not contain the following character/combination of characters:</p> <p>" := @@</p>
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 279.

Editing the Expected Value of an Image Property

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

- **Configure value.** Enables you to define the expected value of the property as a **Constant** or **Parameter**. For information on modifying property values, see “Setting Values in the Configure Value Area” on page 347.
- **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 280.

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** In the Keyword View or Expert View, right-click the checkpoint that you want to modify and select **Checkpoint Properties**. Alternatively, select the step containing the checkpoint and choose **Edit > Step Properties > Checkpoint Properties**. The relevant checkpoint dialog box opens.
- 2** Modify the properties and click **OK**.

12

Checking Bitmaps

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

This chapter describes:	On page:
About Checking Bitmaps	285
Checking a Bitmap	286
Modifying a Bitmap Checkpoint	293

About Checking Bitmaps

You can check an area of a Web page or application as a bitmap. While creating a test, 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 **visible** part of the specified object as a bitmap (QuickTest does not capture any part that is scrolled off of the screen, for example), and inserts a checkpoint in the test. You can choose to save the entire object with your test, or you can choose to save only the selected area of the object with your test to save disk space.

When you run the test, 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 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 653.

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.

To create a bitmap checkpoint while recording:



- 1 Choose **Insert > Checkpoint > Bitmap Checkpoint**, or click the **Insert Checkpoint or Output Value** button and choose **Bitmap Checkpoint**.

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

Note: If the object you want to check can only be displayed by performing an event (such as right-clicking or moving the pointer over an object to display a context menu), hold the left CTRL key. The pointing hand temporarily turns into a standard pointer and you can perform the event. When the object you want to check is displayed, release the left CTRL key. The pointer becomes a pointing hand again. You can also use the left CTRL key to change the window focus. Additionally, if the window containing the object you want to select is minimized, you can display it by holding the left CTRL key, right-clicking the application in the Windows task bar, and choosing **Restore** from the context menu.

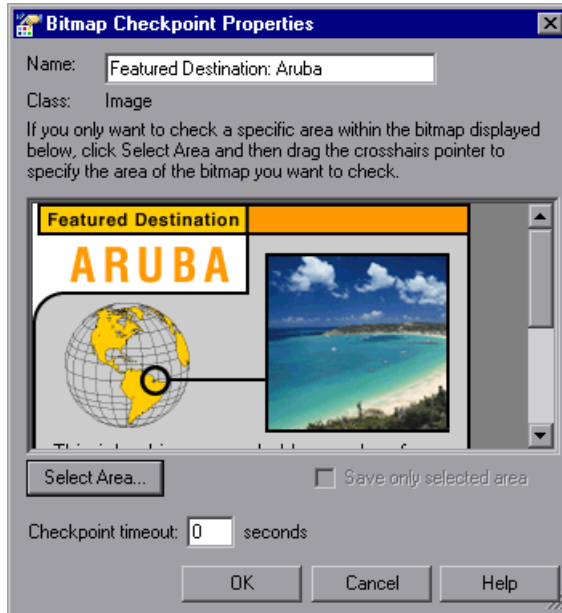
- 2 Click an object to check in your Web page or application. If the location you click is associated with more than one object, the Object Selection - Bitmap Checkpoint Properties dialog box opens.



- 3 Select an object from the tree on which to create a bitmap checkpoint.

Tip: 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.

- 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 In the **Name** box, either accept the name that QuickTest assigns to the checkpoint or specify another name for it. By default, the checkpoint name is the name of the test object on which the checkpoint is being performed.

If you rename the checkpoint, make sure that the name is unique, does not begin or end with a space, and does not contain the following character/combination of characters:

" := @@

Note: The **Class** area displays the type of test object on which the checkpoint is being performed. This is always **Image**.

- 6 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.
- 7 If you want to save only the selected area of the object with your test (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 (**Automation > Update Run Mode**) only updates the saved area of the bitmap, it does not update the original, full size object. To include more of the object in the checkpoint, create a new checkpoint.

- 8 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 help ensure that the object has sufficient time to achieve that state, enabling the checkpoint to pass (if the data matches) before the maximum timeout is reached.

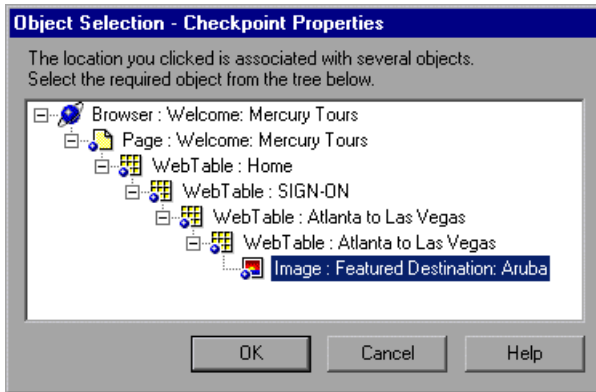
If you specify a checkpoint timeout other than **0**, and the checkpoint fails, the Test Results window displays information on the checkpoint timeout.

- 9 Click **OK** to add the bitmap checkpoint to your test. 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.

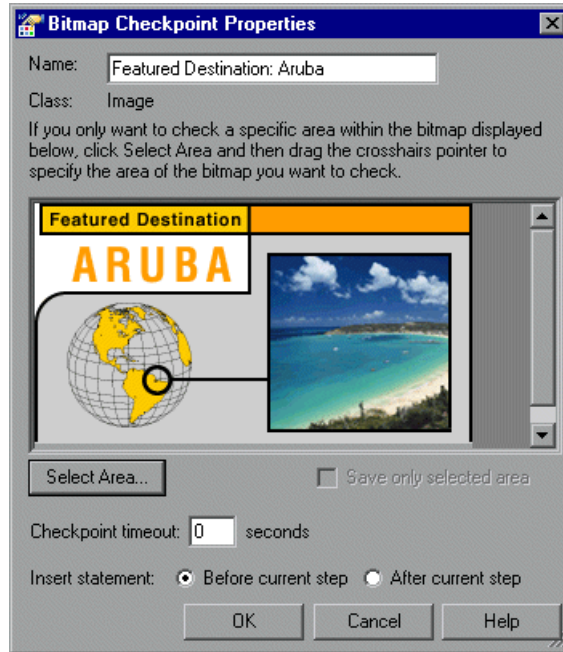


- 4 Select an object from the tree on which to create a bitmap checkpoint.

Note: 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.

- 5 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.

- 6 In the **Name** box, either accept the name that QuickTest assigns to the checkpoint or specify another name for it. By default, the checkpoint name is the name of the test object on which the checkpoint is being performed.

If you rename the checkpoint, make sure that the name is unique, does not begin or end with a space, and does not contain the following character/combination of characters:

" := @@

Note: The **Class** area displays the type of test object on which the checkpoint is being performed. This is always **Image**.

- 7 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.
- 8 If you want to save only the selected area of the object with your test (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 (**Automation > Update Run Mode**) only updates the saved area of the bitmap, it does not update the original, full size object. To include more of the object in the checkpoint, create a new checkpoint.

- 9 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 help ensure that the object has sufficient time to achieve that state, enabling the checkpoint to pass (if the data matches) before the maximum timeout is reached.

If you specify a checkpoint timeout other than **0**, and the checkpoint fails, the Test Results window displays information on the checkpoint timeout.

- 10 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.

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

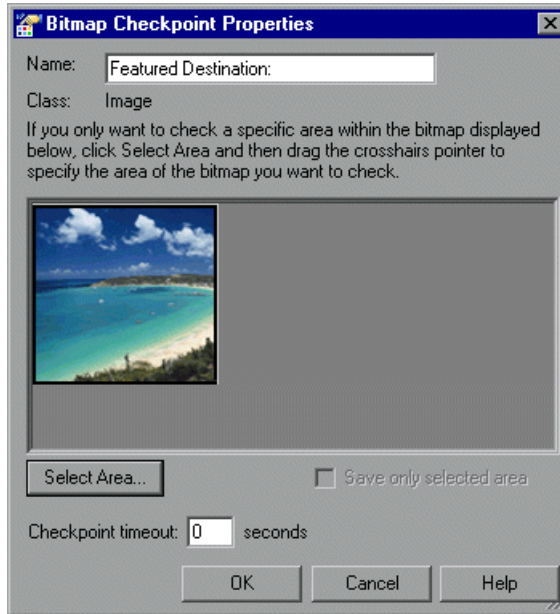
Modifying a Bitmap Checkpoint

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 (**Automation > Update Run Mode**) only updates the saved area of the bitmap, it does not update the original, full size object. To include more of the object in the checkpoint, create a new checkpoint.

To modify a bitmap checkpoint:

- 1 In the Keyword View or Expert View, right-click the checkpoint that you want to modify and select **Checkpoint Properties**. Alternatively, select the step containing the checkpoint and choose **Edit > Step Properties > Checkpoint Properties**. The Bitmap Checkpoint Properties dialog box opens and displays the object or area you saved with the checkpoint.



(This example illustrates the bitmap after it was modified, as described in the following steps.)

- 2 In the **Name** box, either accept the name that QuickTest assigns to the checkpoint or specify another name for it. By default, the checkpoint name is the name of the test object on which the checkpoint is being performed.

If you rename the checkpoint, make sure that the name is unique, does not begin or end with a space, and does not contain the following character/combination of characters:

" := @@

Note: The **Class** area displays the type of test object on which the checkpoint is being performed. This is always **Image**.

- 3** 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.
- 4** If you want to save only the newly selected area of the object with your test (to save disk space), select the **Save only selected area** check box. The Test Results window displays only the selected area of the bitmap.
- 5** 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 help ensure that the object has sufficient time to achieve that state, enabling the checkpoint to pass (if the data matches) before the maximum timeout is reached.

If you specify a checkpoint timeout other than **0**, and the checkpoint fails, the Test Results window displays information on the checkpoint timeout.

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

13

Checking Databases

By adding database checkpoints, you can check the contents of databases accessed by your Web site or application. Database checkpoints are supported by all environments.

This chapter describes:	On page:
About Checking Databases	297
Creating a Check on a Database	298
Understanding the Database Checkpoint Properties Dialog Box	303
Modifying a Database Checkpoint	312

About Checking Databases

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

You can define a database query in the following ways:

- Using Microsoft Query. You can install Microsoft Query from the custom installation of Microsoft Office.
- By manually defining an SQL statement.

Creating a Check on a Database

You create a database checkpoint based on the results of the query (**result set**) you defined on a database. You can create a check on a database to check the contents of the entire result set, or a part of it. QuickTest captures the current data from the database, saves this information as **expected data**, and inserts a database checkpoint into the test. This checkpoint is displayed in the Expert View as a **DbTable.Check CheckPoint** statement and as a step in the Keyword View, as follows:



When you run the test, the database checkpoint compares the current data in the database to the expected data defined in the Database Checkpoint Properties dialog box. If the expected data and the current results do not match, the database checkpoint fails. You can view the results of the checkpoint in the Test Results window. For more information, see Chapter 24, “Analyzing Test Results.”

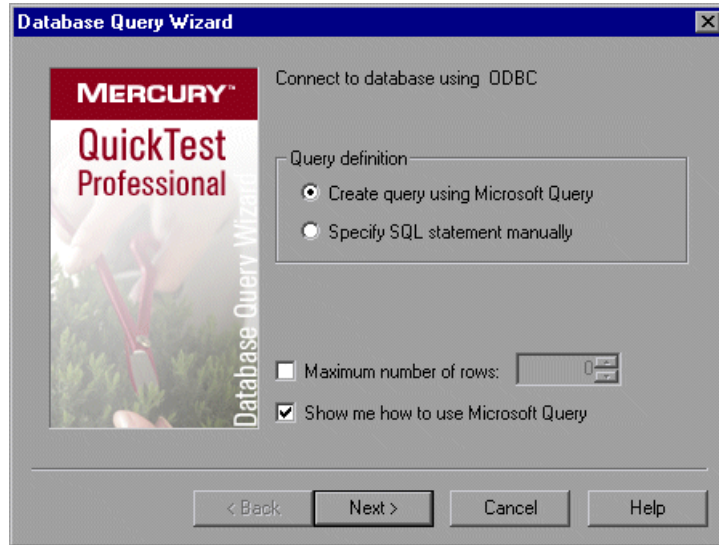
You can modify the expected data of a database checkpoint before you run your test. You can also make changes to the query in an existing database checkpoint. This can be useful if you move the database to a new location on the network.

Creating a Database Checkpoint

You can define the query for your checkpoint using Microsoft Query or by manually entering a database connection and SQL statement.

To create a database checkpoint:

- 1 Choose **Insert > Checkpoint > Database Checkpoint**. The Database Query Wizard opens.



- 2 Select your database selection preferences. You can choose from the following options:
 - **Create query using Microsoft Query.** Opens Microsoft Query, enabling you to create a new query. After you finish defining your query, you return to QuickTest. This option is available only if you have Microsoft Query installed on your computer.
 - **Specify SQL statement manually.** Opens the **Specify SQL statement** screen in the wizard, which enables you to specify the connection string and an SQL statement.

- **Maximum number of rows.** Select this check box if you would like to limit the number of rows and enter the maximum number of database rows to check. You can specify a maximum of 32,000 rows.
 - **Show me how to use Microsoft Query.** Displays an instruction screen when you click **Next** before opening Microsoft Query. (Enabled only when **Create query using Microsoft Query** is selected).
- 3** Click **Next**. The screen that opens depends on the option you selected in the previous step.
- If you chose **Create query using Microsoft Query** in the previous step, Microsoft Query opens. Choose a data source and define a query. For more information on creating a query, see “Creating a Query in Microsoft Query” on page 301.

Note: If you chose **Show me how to use Microsoft Query**, the Instruction for Microsoft Query screen opens. When you click **OK**, Microsoft Query opens.

- If you chose **Specify SQL statement manually** in the previous step, the Specify SQL statement screen opens. Specify the connection string and the SQL statement, and click **Finish**. For more information on specifying SQL statements, see “Specifying SQL Statements” on page 302.

The Database Checkpoint Properties dialog box opens.

- 4** Select the checks to perform on the result set as described in “Understanding the Database Checkpoint Properties Dialog Box” on page 303. You can also modify the expected data in the result set.
- 5** Click **OK** to close the dialog box. A checkpoint statement is added for the selected object in the Keyword View and Expert View.

Creating a Query in Microsoft Query

You can use Microsoft Query to choose a data source and define a query within the data source.

To choose a data source and define a query in Microsoft Query:

- 1** When Microsoft Query opens during the insert database checkpoint process, choose a new or an existing data source.
- 2** Define a query.
- 3** When you are done, in the Finish screen of the Query Wizard, select **Exit and return to QuickTest Professional** and click **Finish** to exit Microsoft Query. Alternatively, click **View data or edit query in Microsoft Query** and click **Finish**. After viewing or editing the data, choose **File > Exit and return to QuickTest Professional** to close Microsoft Query and return to QuickTest.
- 4** The Database Checkpoint Properties dialog box opens. Select the checks to perform on the result set as described in “Understanding the Database Checkpoint Properties Dialog Box” on page 303. You can also modify the expected data in the result set.
- 5** Click **OK** to close the dialog box. A checkpoint statement is added for the selected object in the Keyword View and Expert View.

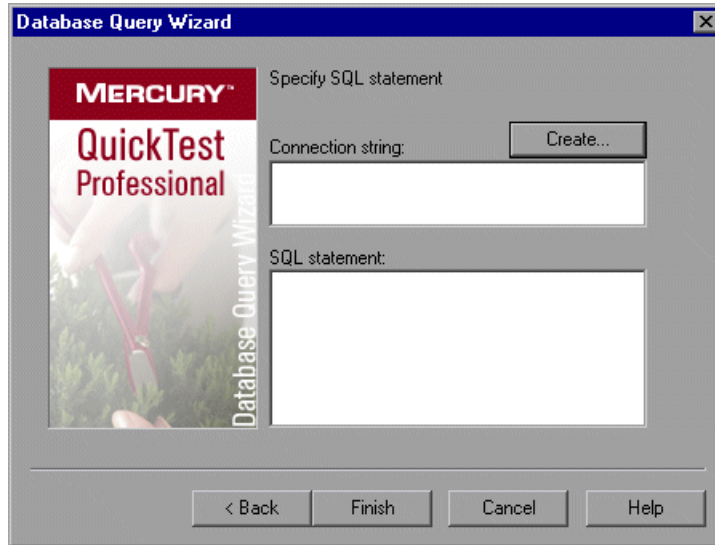
For more information on working with Microsoft Query, refer to your Microsoft Query documentation.

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. The following screen opens:



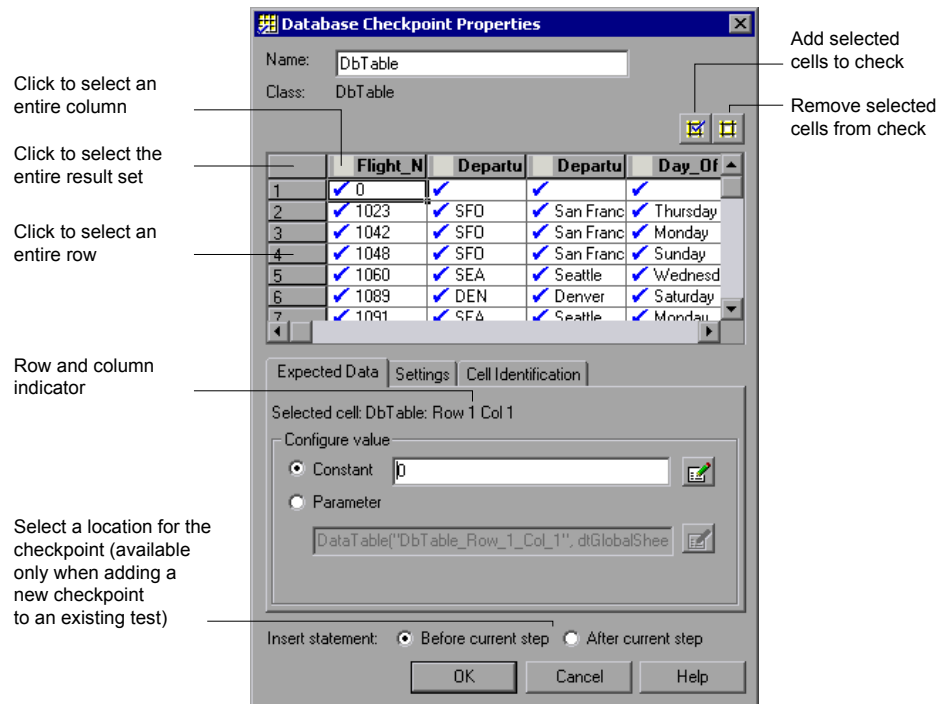
- 2 Specify the connection string and the SQL statement, and click **Finish**.
 - **Connection string.** Enter the connection string, or click **Create** to open the ODBC Select Data Source dialog box. You can select a **.dsn** file in the ODBC Select Data Source dialog box or create a new **.dsn** file to have the Database Query Wizard insert the connection string in the box for you.
 - **SQL statement.** Enter the SQL statement.

QuickTest takes several seconds to capture the database query and restore the QuickTest window.

- 3 Select the checks to perform on the result set as described in “Understanding the Database Checkpoint Properties Dialog Box” on page 303. You can also modify the expected data in the result set.
- 4 Click **OK** to close the dialog box. A checkpoint statement is added for the selected object in the Keyword View and Expert View.

Understanding the Database Checkpoint Properties Dialog Box

The Database Checkpoint Properties dialog box enables you to specify which cell contents of your database to check and which verification method and type to use. You can also edit or parameterize the expected data for the cells included in the check.



The Database Checkpoint Properties dialog box enables you to check database content.

- The information area at the top of the dialog box displays the name of the checkpoint and the class of the test object on which the checkpoint is being performed. You can rename the checkpoint, if required. For more information, see “Identifying the Database Checkpoint” on page 305.
- The grid area displays the data that was captured for the checkpoint. This is the expected data. You use this area to specify which cells you want to check. For more information, see “Specifying Which Cells to Check” on page 305.
- **Expected Data tab.** Enables you to set each checked cell as a constant or parameterized value. For more information, see “Specifying the Expected Data” on page 307.
- **Settings tab.** Enables you to set the criteria for a successful match between the expected and actual values. For more information, see “Specifying the Value Type Criteria in the Settings Tab” on page 308.
- **Cell Identification tab.** Enables you to instruct QuickTest how to locate the cells to be checked. For more information, see “Specifying the Cell Identification Settings” on page 310.

Tip: The value matching settings and cell identification criteria apply to all selected cells in the checkpoint. If you want to use different value matching or cell identification criteria for different cells in the database, create separate checkpoints and specify the relevant cells for each.

Identifying the Database Checkpoint

The top part of the Database Checkpoint Properties dialog box contains the following options:



A screenshot of the Database Checkpoint Properties dialog box. It shows two fields: 'Name' with the value 'DbTable' and 'Class' with the value 'DbTable'.



Name	<p>The name that QuickTest assigns to the checkpoint. By default, the checkpoint name is the name of the test object on which the checkpoint is being performed. You can specify a different name for the checkpoint or accept the default name.</p> <p>If you rename the checkpoint, make sure that the name is unique, does not begin or end with a space, and does not contain the following character/combination of characters:</p> <p>" := @@</p>
Class	Specifies the type of object (read-only).

Specifying Which Cells to Check

The grid area of the Database Checkpoint Properties dialog box represents the cells in the captured result set.

Tip: You can change the column widths and row heights of the grid by dragging the boundaries of the column and row headers.

When you create a new database checkpoint, all cells contain a blue check mark, indicating they are selected for verification. You can select to check the entire results set, specific rows, specific columns, or specific cells. QuickTest checks only cells containing a check mark.

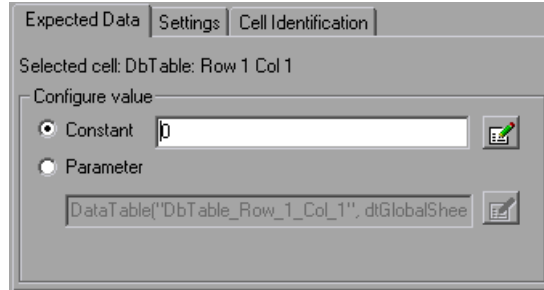
To:	Do this:
Add a single cell to or remove it from the check	Double-click the cell
Add an entire row to or remove it from the check	Double-click the row header
Add an entire column to or remove it from the check	Double-click the column header
Add the entire result set to or remove all cells from the check	Double-click the top-left corner of the grid
Add a range of cells to the check	Select the cells to add to the check and click the Add to Check button 
Remove a range of cells from the check	Select the cells to remove from the check and click the Remove from Check button 

Notes:

- Double-clicking on the grid toggles the settings for all selected cells. Therefore, if you double-click a row header, a column header, or the top left corner of the grid, any cells that were previously included in the check are removed from it, and any cells that were not previously included in the check are added to it.
 - When more than one cell is selected, the options on the Expected Data tab are disabled.
-

Specifying the Expected Data

The Expected Data tab displays options for setting the expected value of the selected cell in the result set.



You can modify the value of a cell or you can parameterize it to use a value from an external source, such as the Data Table or an environment variable. During the run session, QuickTest compares the value specified in this tab with the actual value that it finds. If the expected value and the actual value do not match, the checkpoint fails. For example, you can instruct QuickTest to use a value from the Data Table as the expected value for a particular cell.

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

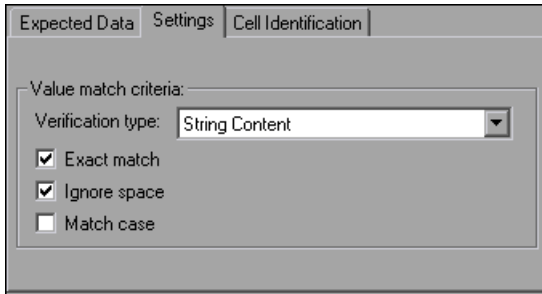
The Expected Data tab includes the following:

Selected cell	Indicates the table name and the row and column numbers of the selected cell.
Configure value area	Enables you to set the expected value of the cell as a constant or parameter. For more information on modifying values, see “Setting Values in the Configure Value Area” on page 347.

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

Specifying the Value Type Criteria in the Settings Tab

The Settings tab includes options that determine how the actual cell values are compared with the expected cell values. For example, you can instruct QuickTest to treat the value as a number so that 45 or 45.00 are treated as the same value, or you can instruct QuickTest to ignore spaces when comparing the values. The settings in this tab apply to all selected cells.



The default setting is to treat cell values as strings and to check for the exact text, while ignoring spaces.

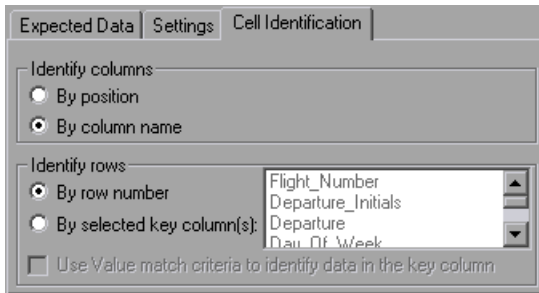
The Settings tab includes the following options:

Option	Description
Verification type	<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.
Exact match	<p>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.</p> <p>Note: QuickTest displays this option only when String Content is selected as the Verification type.</p>
Ignore space	<p>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.</p> <p>Note: QuickTest displays this option only when String Content is selected as the Verification type.</p>
Match case	<p>Conducts a case sensitive search.</p> <p>Note: QuickTest displays this option only when String Content is selected as the Verification type.</p>
Min / Max	<p>Specifies the numeric range against which the content of the cell is compared. The range values can be any real number.</p> <p>Note: QuickTest displays this option only when Numeric Range is selected as the Verification type.</p>


Specifying the Cell Identification Settings

The settings in the Cell Identification tab determine how QuickTest locates the cells to be checked. For example, suppose you want to check the data that is displayed in the first row and second column in the Database Checkpoint Properties dialog box. However, you know that each time you run your test, it is possible that the rows may be in a different order, depending on the sorting that was performed in a previous step. Therefore, rather than finding the data based on row and column numbers, you may want QuickTest to identify the cell based on the column name and the row containing a known value in a **key column**.

The settings in this tab apply to all selected cells.



The Cell Identification tab includes the following options:

Identify columns	<p>Specifies the location of the column (in your actual database) containing the cell(s) 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. • By column name. Default. Locates cells according to the column name. A shift in the position of the columns within the database does not result in a mismatch.
Identify rows	<p>Specifies the location of the row (in your actual database) containing the cell(s) 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 key column. A shift in the position of the row(s) does not result in a mismatch. If more than one row is identified, QuickTest checks the first matching row. You can use more than one key column to uniquely identify any row. <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 Database Checkpoint

You can modify the SQL query definition and the expected data in an existing database checkpoint.

To modify the SQL query definition:

- 1** In the Keyword View, right-click the database object that you want to modify.
- 2** Select **Object Properties**.
- 3** Modify the SQL and connection string properties as necessary and click **OK**.

To modify the expected data in a database checkpoint:

- 1** In the Keyword View or Expert View, right-click the database checkpoint that you want to modify and select **Checkpoint Properties**. Alternatively, select the step containing the checkpoint and choose **Edit > Step Properties > Checkpoint Properties**. The Database Checkpoint Properties dialog box opens.
- 2** Modify the settings as described “Understanding the Database Checkpoint Properties Dialog Box” on page 303.

14

Checking XML

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

This chapter describes:	On page:
About Checking XML	314
Creating XML Checkpoints	316
Updating the XML Hierarchy for XML Test Object Operation Checkpoints (for WebService Test Objects Only)	334
Modifying XML Checkpoints	342
Reviewing XML Checkpoint Results	342
Using XML Objects and Methods to Enhance Your Test	343

About Checking XML

XML (Extensible Markup Language) is a meta-markup language for text documents that is endorsed as a standard by the World Wide Web Consortium (W3C). XML makes the complex data structures portable between different computer environments/operating systems and programming languages, facilitating the sharing of data.

XML files contain text with simple tags that describe the data within an XML document. These tags describe the data content, but not the presentation of the data. Applications that display an XML document or file use either Cascading Style Sheets (CSS) or XSL Formatting Objects (XSL-FO) to present the data.

You can verify the data content of XML files with XML checkpoints. A few common uses of XML checkpoints are described below:

- An XML file can be a static data file that is accessed to retrieve commonly used data for which a quick response time is needed—for example, country names, zip codes, or area codes. Although this data can change over time, it is normally quite static. You can use an XML file checkpoint to validate that the data has not changed from one application release to another.
- An XML file can consist of elements with attributes and values (character data). There is a parent and child relationship between the elements, and elements can have attributes associated with them. If any part of this hierarchy (including data) changes, your application's ability to process the XML file may be affected. Using an XML checkpoint, you can check the content of an element to make sure that its tags, attributes, and values have not changed.
- Web service operations often return XML values. If the Web Service Add-in is installed on your computer, you can send a Web service operation command to the service and use an XML checkpoint to verify that the service returns the XML in the expected structure and with the expected values.

- XML files are often an intermediary that retrieves dynamically changing data from one system. The data is then accessed by another system using Document Type Definitions (DTD), enabling the accessing system to read and display the information in the file. You can use an XML checkpoint and parameterize the captured data values if you want to check an XML document or file whose data changes in a predictable way.
- XML documents and files often need a well-defined structure to be portable across platforms and development systems. One way to accomplish this is by developing an XML schema, which describes the structure of the XML elements and data types. You can use schema validation to check that each item of content in an XML file adheres to the schema description of the element in which the content is to be placed.

Note: XML checkpoints are compatible with namespace standards and a change in namespace between expected and actual values will result in a failed checkpoint.

For more information on XML standards, refer to <http://www.w3.org/XML/>

For more information on namespace standards, refer to <http://www.w3.org/TR/1999/REC-xml-names-19990114/>

Creating XML Checkpoints

You can perform checkpoints on XML documents contained in Web pages or frames, on XML files, and on test objects that support XML. An XML checkpoint is a verification point that compares a current value for a specified XML element, attribute and/or value with its expected value. When you insert a checkpoint, QuickTest adds a checkpoint step in the Keyword View and adds a **Check CheckPoint** statement in the Expert View. When you run the test, QuickTest compares the expected results of the checkpoint to the current results. If the results do not match, the checkpoint fails.

After you run your test, you can view summary results of the XML checkpoint in the Test Results window. You can also view detailed results by opening the XML Checkpoint Results window. For more information, see Chapter 24, “Analyzing Test Results.”

You can create three types of XML checkpoints:

- **XML Web Page/Frame Checkpoint.** Checks an XML document within a Web page or frame.
- **XML File Checkpoint.** Checks a specified XML file.
- **XML Test Object Checkpoint.** Checks the XML data for an object or operation.

Creating XML Checkpoints for Web Pages and Frames

You can create an XML checkpoint for any XML document contained in a Web page or frame using the **XML Checkpoint (From Application)** option. You can create an **XML Checkpoint (From Application)** only while recording.

To create an XML Checkpoint on XML contained in a Web page or frame:

1 Begin recording your test.



2 Choose **Insert > Checkpoint > XML Checkpoint (From Application)**, or click the **Insert Checkpoint or Output Value** toolbar button and select **XML Checkpoint (From Application)**.

Note: The **XML Checkpoint (From Application)** option is available only when the Web Add-in is installed and loaded. For more information on loading add-ins, see Chapter 28, “Working with QuickTest Add-Ins.”

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

Tips:

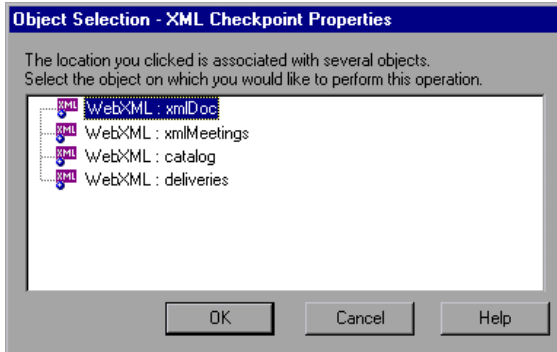
Hold the left CTRL key to change the window focus or perform operations such as right-clicking or moving the pointer over an object to display a context menu. If the window containing the object you want to select is minimized, you can display it by holding the left CTRL key, right-clicking the application in the Windows task bar, and choosing **Restore** from the context menu.

You can also insert a Web page or frame checkpoint using the **XML (From Resource)** option by selecting an existing WebXML test object as long as the actual XML object is currently available (open in the browser). For more information, see “Creating XML Test Object Checkpoints” on page 324.

- 3 Click a Web page or frame that contains XML documents.

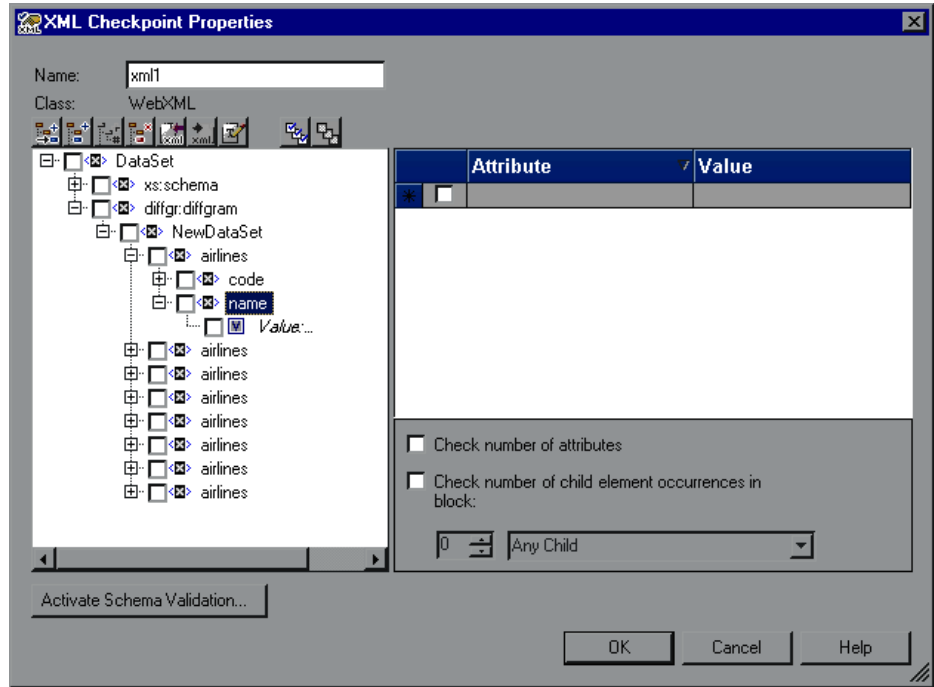
If only one XML file is associated with the Web page or frame, the XML Checkpoint Properties dialog box opens. In this case, proceed to step 5.

If more than one XML document is associated with the selected location, the Object Selection - XML Checkpoint Properties dialog box opens.



- 4 Select the XML document you want to check, and click **OK**.

The XML Checkpoint Properties dialog box opens.



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

Note: If the XML source on which you base your checkpoint is in a valid XML format, but not to W3 standards, an error message informs you that the XML tree in the dialog box will be displayed in read-only format and that you must fix the XML source using the Edit XML as Text dialog box. For more information on this dialog box, see “Understanding the Edit XML as Text Dialog Box” on page 333.

- 5** In the **Name** box, either accept the name that QuickTest assigns to the checkpoint or specify another name for it. By default, the checkpoint name is the name of the test object on which the checkpoint is being performed.

If you rename the checkpoint, make sure that the name is unique, does not begin or end with a space, and does not contain the following character/combination of characters:

" := @@

- 6** Select the items to check for the checkpoint. For more information, see “Understanding the XML Checkpoint Properties Dialog Box” on page 328.
- 7** When you finish setting your checkpoint preferences, click **OK** to add the XML checkpoint. A checkpoint similar to the following is added to the Keyword View.

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")
```

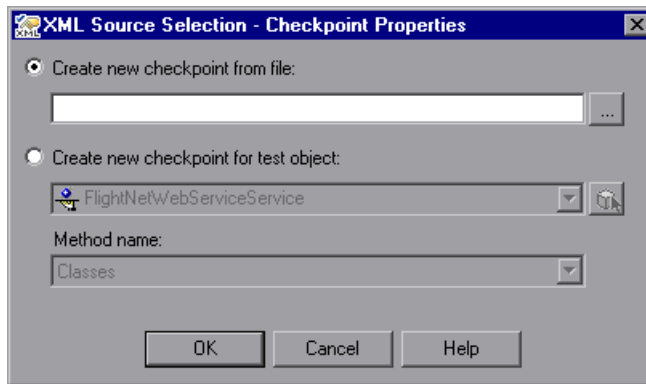
Creating XML File Checkpoints

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

To create an XML file checkpoint:



- 1 Choose **Insert > Checkpoint > XML Checkpoint (From Resource)** or click the **Insert Checkpoint or Output Value** toolbar button, and select **XML Checkpoint (From Resource)**. The XML Source Selection - Checkpoint Properties dialog box opens.



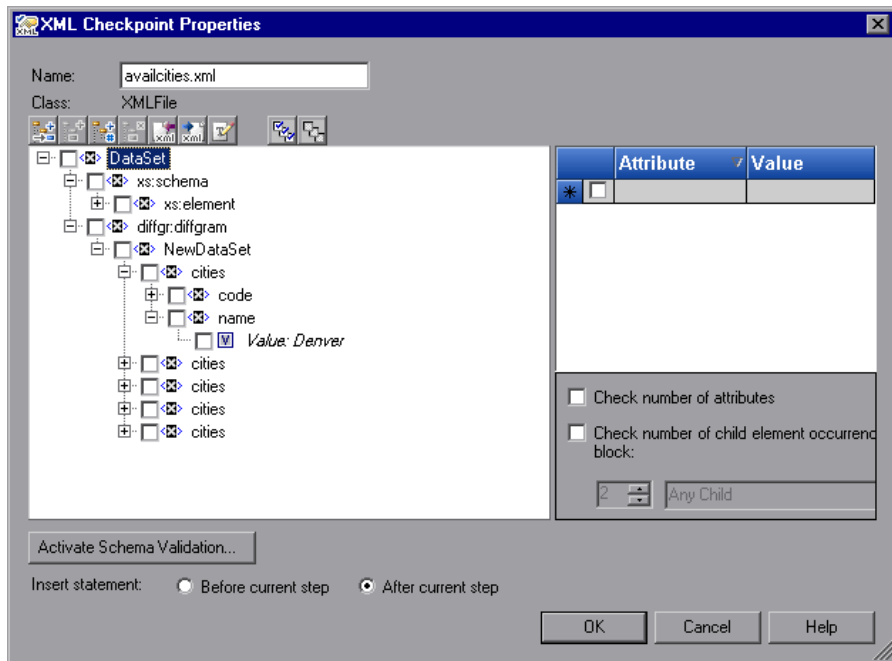
Tip: You can also insert an XML File checkpoint by selecting an existing XMLFile test object as long as the source file for the test object exists in the location stored with the test object. For more information, see “Creating XML Test Object Checkpoints” on page 324.

- 2 Select **Create new checkpoint from file**. Enter the file path or Internet address of the XML file.

Alternatively, click the browse button to open the Open XML File dialog box, and then navigate to the XML file for which you want to create a checkpoint. You can specify an XML file either from your file system or from Quality Center. Select the file and click **Open**. The file path and name are entered in the box.

Note: 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 “Setting Folder Testing Options” on page 712.

- 3 Click **OK** in the XML Source Selection - Checkpoint Properties dialog box. 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: If the XML source on which you base your checkpoint is in a valid XML format, but not to W3 standards, an error message informs you that the XML tree in the dialog box will be displayed in read-only format and that you must fix the XML source manually using the Edit XML as Text dialog box. For more information on this dialog box, see “Understanding the Edit XML as Text Dialog Box” on page 333.

- 4 In the **Name** box, either accept the name that QuickTest assigns to the checkpoint or specify another name for it. By default, the checkpoint name is the name of the test object on which the checkpoint is being performed.

If you rename the checkpoint, make sure that the name is unique, does not begin or end with a space, and does not contain the following character/combination of characters:

" := @@

- 5 Select the items to check for the checkpoint. For more information, see “Understanding the XML Checkpoint Properties Dialog Box” on page 328.
- 6 When you finish setting your checkpoint preferences, click **OK** to add the XML checkpoint. A checkpoint similar to the following is added to the Keyword View.

Item	Operation	Value	Documentation
▼ Action1			
FlightNetWebServiceService	AvailableCities		<No documentation summary is av.
availcities.xml	Check	CheckPoint("availcities.xml")	Check whether the content of the :

QuickTest inserts this step as follows in the Expert View:

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

Creating XML Test Object Checkpoints

You can create an XML test object checkpoint to check the elements, attributes and/or values of XML associated with the selected test object. For example, you can check the XML that is returned from an operation performed on a Web service. You can create an XML test object checkpoint while you are recording or editing your test.

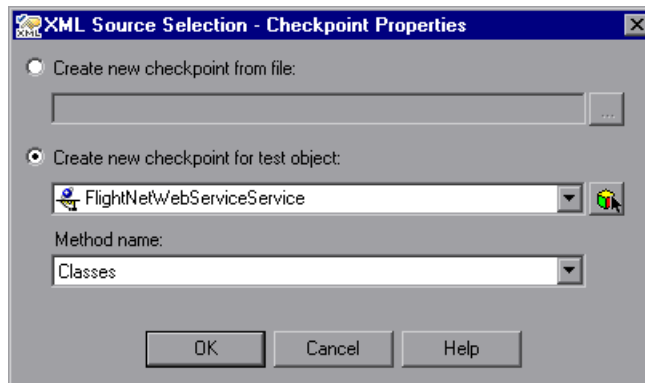
Note: You cannot insert an XML checkpoint from the Active Screen.

To create an XML test object checkpoint:



- 1 Choose **Insert > Checkpoint > XML Checkpoint (From Resource)**, or click the **Insert Checkpoint or Output Value** toolbar button and select **XML Checkpoint (From Resource)**.

The XML Source Selection - Checkpoint Properties dialog box opens.



- 2 Select **Create new checkpoint for test object** and select the test object you want to check.



To select an object that is not displayed in the list, click **Object from Repository**. Then select an XML test object from the object repository on which to create a new checkpoint. The selected object must support XML. For information on selecting objects, see “Selecting an Object from the Repository or Application” on page 552.

You can select an existing WebXML or XMLFile test object type as long as the actual XML object is currently available (in an open browser or in the file system, as relevant) or, if you are working with the QuickTest Web Services Add-in you can select a WebService test object.

Note: Selecting a WebXML or XMLFile test object is identical to using the **XML Checkpoint (From Application)** or **Create new checkpoint from file** options, but may be faster than browsing to these objects and can be inserted while recording or editing. However, to use this option, the XML source must be available when you select the test object (the Web page must be open or the file must exist in the same location as when the test object was defined).

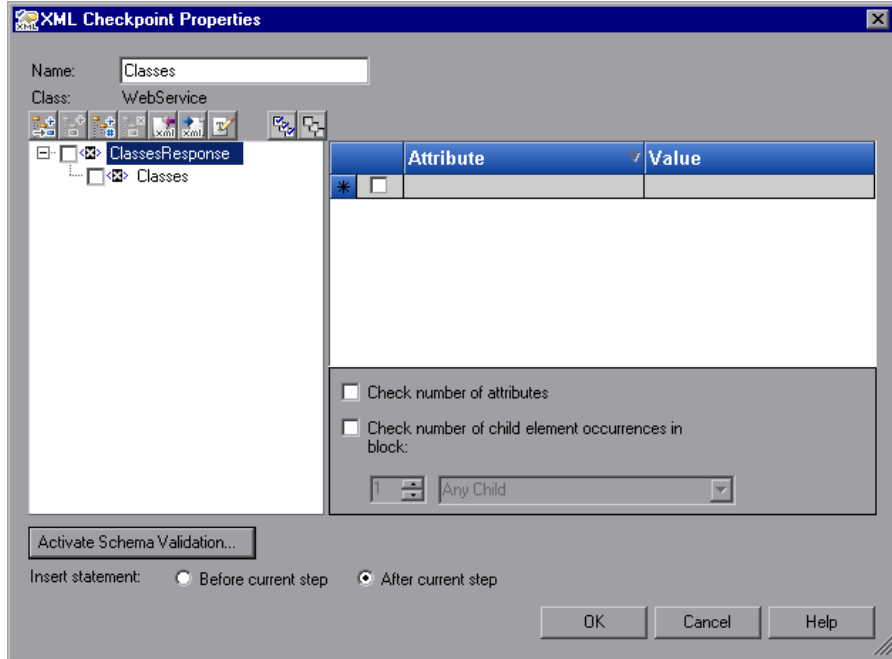
- 3** If you select a WebService test object, then the **Method name** box is enabled. Select the Web service operation whose return values you want to check.

Notes:

The **Method name** box is available only if the Web Services Add-in is installed and loaded. The **Method name** box is enabled only if you select a WebService test object.

XML Checkpoints on Web service operations compare the expected values of the checkpoint to the actual values returned from the last native Web service operation performed on the test object. If a different Web service operation step is performed prior to the checkpoint, then the checkpoint will fail.

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




Note: When you create an XML checkpoint for a test object operation return value (for a WebService test object), only a generic XML tree is created and shown in the XML Checkpoint Properties dialog box. The data that is expected when each operation is called during the test is not included. You must update the XML hierarchy by populating the XML tree with the actual elements, attributes, and values you want to check. For more information, see “Updating the XML Hierarchy for XML Test Object Operation Checkpoints (for WebService Test Objects Only)” on page 334.

- 5** In the **Name** box, either accept the name that QuickTest assigns to the checkpoint or specify another name for it. By default, the checkpoint name is the name of the test object on which the checkpoint is being performed.

If you rename the checkpoint, make sure that the name is unique, does not begin or end with a space, and does not contain the following character/combination of characters:

" := @@

- 6** Select the items to check for the checkpoint. For more information, see “Understanding the XML Checkpoint Properties Dialog Box” on page 328.
- 7** When you finish setting your checkpoint preferences, click **OK** to add the XML checkpoint. A checkpoint similar to the following is added to the Keyword View.

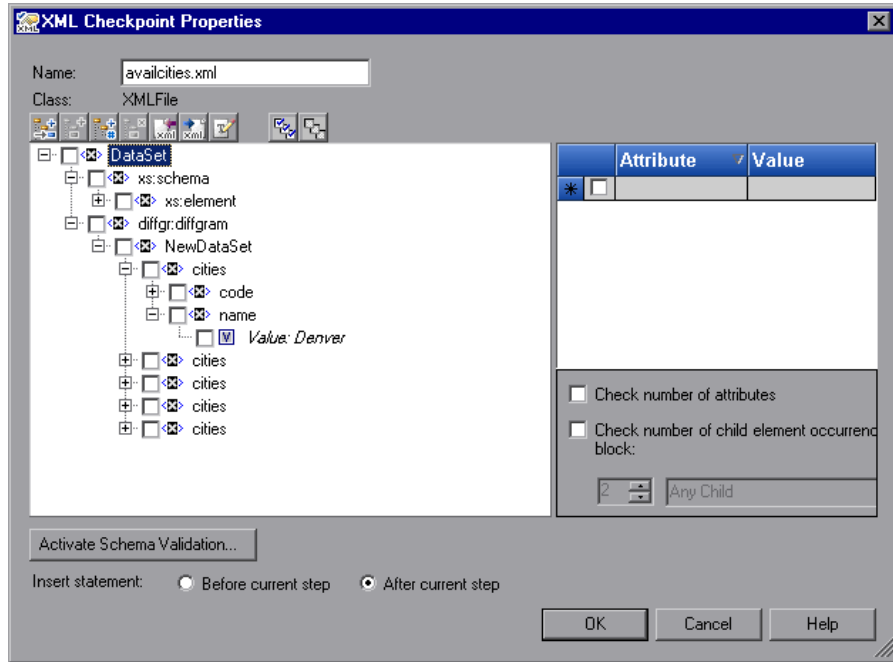
▼  Action1		
 FlightNetWebService	Airlines	
 FlightNetWebService	Check	CheckPoint("Airlines")

QuickTest records this step in the Expert View as:

```
WebService("FlightNetWebService").Check CheckPoint("Airlines")
```

Understanding the XML Checkpoint Properties Dialog Box

The XML Checkpoint Properties dialog box enables you to choose which elements, attributes, and/or values to check. You can also add, modify and delete the elements, attributes, and values in the XML tree.



In the XML tree, select the check boxes of 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.






Identifying the Object





The top part of the dialog box displays test object information on the test object for which you are creating a checkpoint:

Option	Description
Name	<p>The name that QuickTest assigns to the checkpoint. By default, the checkpoint name is the name of the test object on which the checkpoint is being performed. You can specify a different name for the checkpoint or accept the default name.</p> <p>If you rename the checkpoint, make sure that the name is unique, does not begin or end with a space, and does not contain the following character/combination of characters:</p> <p>" := @@</p>
Class	<p>The test object class on which you are creating the checkpoint. This can be: XMLFile (for files), WebXML (for Web pages or frames), or WebService (for a Web service).</p>



Modifying the XML Tree

The following buttons are available according to the node you select in the tree:

Button		Description
Add Child		Adds a child node below the selected node in the tree.
Insert Sibling		Adds a sibling node at the same level as the selected node in the tree.
Add Value		Enables you to assign a constant or parameterized value to the selected element.
Delete		Deletes the selected node. Note that you cannot delete the root node of the checkpoint.
Import XML		Enables you to browse to an existing XML file and import it. The new file replaces the selected node and its current sub-tree.

Button		Description
Export XML		Enables you to save the content of the checkpoint tree to an XML file. Enabled only when the root node of the tree is selected.
Edit XML as Text		Opens the Edit XML as Text dialog box, enabling you to modify the XML text of the selected node and its subnodes in a test editor. For more information, see “Understanding the Edit XML as Text Dialog Box” on page 333.
Select All		Selects all element and value nodes in the XML tree as well as all element attributes.
Clear All		Clears all element and value nodes in the XML tree as well as all element attributes.

XML Tree

The XML tree displays the hierarchical relationship between each element and value in the XML tree, enabling you to select the specific elements, attributes and values you want to check. Each element is displayed with a  icon. Each value is displayed with a  icon.

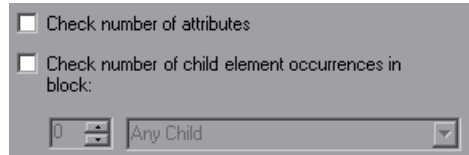
Select the check box next to an element or value node to include that item in the checkpoint. Select an element node in the XML tree to display, edit, or parameterize its expected attributes and values on the right of the XML Checkpoint Properties dialog box. Select a value node in the XML tree to display, edit, or parameterize its expected value on the right of the XML Checkpoint Properties dialog box.

Tip: The XML tree pane and the **Attribute** and **Value** columns in the right pane are resizable.

Checkpoint Options

The checkpoint options area on the bottom right of the XML Checkpoint Properties dialog box enables you to select the types of checks you want to perform on selected elements.

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



Element Checks

The following element checks are available:

Check	Description
Check number of attributes	Checks the number of attributes that are attached to the element.
Check number of child element occurrences in block	<p>Displays the number of child elements associated with the selected parent element. If you select this option, QuickTest verifies that the number of child elements in your XML tree (with the specified name, if applicable) corresponds to the number that appears in the Check number of child element occurrences in block field.</p> <p>You can specify the child element name for the Number of child element occurrences check. If you select a child element name, QuickTest verifies that the number of child elements with that name corresponds to the number that you specify in the Number of child element occurrences in block field.</p> <p>Select Any Child (default) to check the total number of child elements associated with the selected parent element.</p>

Schema Validation

You can use the **Activate Schema Validation** button to confirm that the XML in your application or file adheres to the structure defined in a specific XML schema or schemas. You can validate the structure of the XML you are checking using one or more external schema files or using schema(s) embedded within your XML document. For more information, see “Understanding the Schema Validation Dialog Box” on page 338.

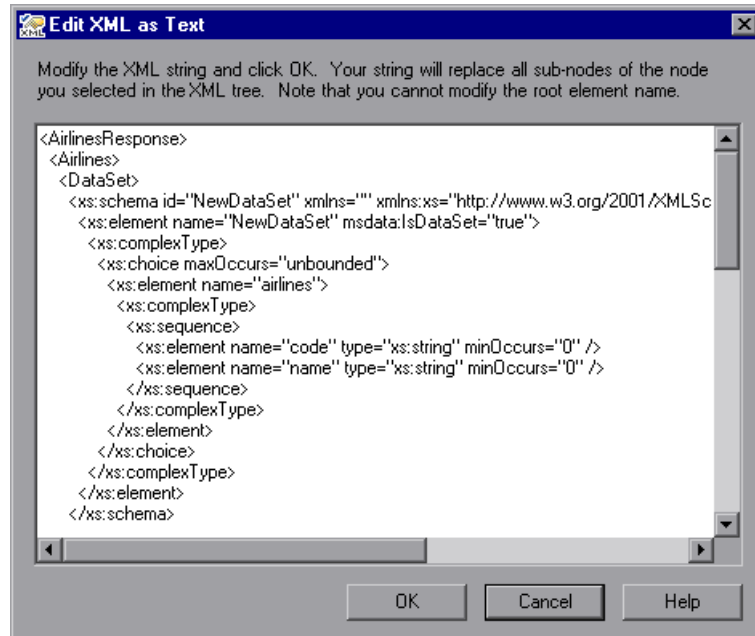
Insert Statement Options

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

Note: The **Insert statement** options are not available if you are adding an XML checkpoint while recording or if you are modifying an existing XML checkpoint. They are available only if you are adding a new XML checkpoint to an existing test.

Understanding the Edit XML as Text Dialog Box

The Edit XML as Text dialog box enables you to edit XML content from the XML tree in a text editor.



This dialog box is used mainly for constructing an entire XML segment from a string or for fixing syntax problems that prevent the dialog box from displaying the XML tree correctly. It is also useful when you want to use copy-paste functionality to edit the tree.

When you click **OK** in the Edit XML as Text dialog box, the sub-tree of the node you previously selected in the XML tree (or the entire tree if no node was selected or if the root node was selected) is completely replaced by the XML content from the Edit XML as Text dialog box.

Note: You cannot modify the name of the root element displayed in the Edit XML as Text dialog box.

Updating the XML Hierarchy for XML Test Object Operation Checkpoints (for WebService Test Objects Only)

This section is relevant only when working with XML Checkpoints on WebService test object operations (with the QuickTest Professional Web Services Add-in).

When you create an XML checkpoint for a test object operation (for a WebService test object), the expected operation return value data cannot be generated. Therefore, only a generic XML tree is created. To check the operation return values, you must first populate the XML tree with the actual elements, attributes, and values that the operation is expected to return.

You can use one of the three methods below to populate the XML tree:

- Updating the XML Tree Manually
- Importing an XML Tree from a File
- Updating the XML Tree Using Update Run Mode

Updating the XML Tree Manually

You can update the XML tree by adding elements, attributes, and values manually in the XML Checkpoint Properties dialog box.

To update the XML tree manually:

- 1** In the Keyword View, select the checkpoint whose XML tree you want to update. Click in the **Value** cell.



- 2** Click the **Checkpoint Properties** button or right-click and select **Checkpoint Properties**. The XML Checkpoint Properties dialog box opens.

- 3** Select a node in the XML tree and then click a toolbar button or choose an option from the right-click menu to:



- Add an element at the same level as the selected node
- Add an element below the selected node
- Add a value to the selected node





- Edit the XML text of the selected node
- Delete the selected node

For more information on the available tools in the XML Checkpoint Properties dialog box, see “Understanding the XML Checkpoint Properties Dialog Box” on page 328.

Importing an XML Tree from a File

You can import an XML tree from an existing file for a specific element in the XML tree hierarchy or for the whole tree.

To import an existing XML tree from a file:

- 1** In the Keyword View, select the checkpoint whose XML tree you want to update.
-  **2** Click in the **Value** cell and then click the **Checkpoint Properties** button. The XML Checkpoint Properties dialog box opens.
- 3** If you want to import an XML hierarchy for the whole XML tree, select the root node. If you want to import an XML hierarchy for a specific element, select the element in the XML tree hierarchy.
-  **4** Click the **Import XML** button. A message warns you that the imported hierarchy overwrites the selected node and its sub-tree. Click **Yes** to close the message.
- 5** In the Import XML from File dialog box, browse to the required XML file and click **Open**. The XML hierarchy is imported from the file.
- 6** If required, configure a constant or parameterized value for each of the element and value nodes in the XML tree. For more information on parameterizing values, see Chapter 16, “Parameterizing Values.”

Updating the XML Tree Using Update Run Mode

QuickTest cannot generate the expected return values of an operation when you insert an XML checkpoint on a Web service operation, but it can generate this information after it runs the operation. Therefore, you can run your Web service test in Update Run mode to automatically populate or update the elements, attributes and values in your XML tree.

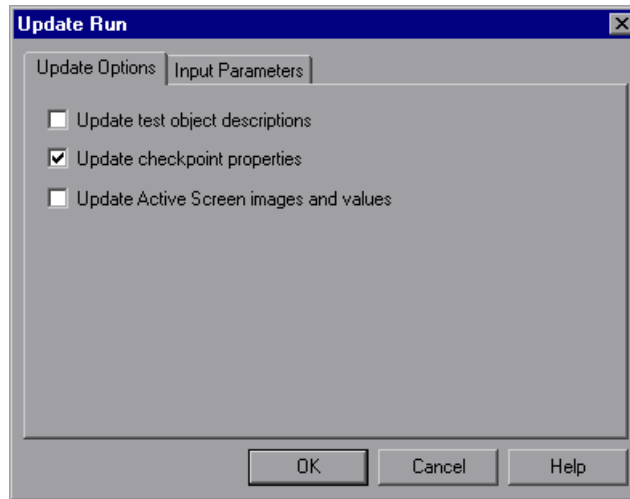
To generate a new XML tree based on the current return values of the Web service operation, ensure that none of the node, attribute, or value check boxes are selected in the XML checkpoint.

To maintain the current hierarchy in the XML tree and update only the expected values, select one or more node, attribute, or value check boxes in the dialog box.

Note: XML Checkpoints on Web service operations check the actual values returned from the last native Web service operation performed on the test object. If a different Web service operation step is performed prior to the checkpoint, then the update run will be unable to update the XML tree for that operation.

To update an XML tree using Update Run mode:

- 1** Open a test containing XML Test Object checkpoints for Web service operations.
- 2** Click **Update Run Mode** or choose **Automation > Update Run Mode**.
- 3** Click **Run** or choose **Automation > Run**. The Update Run dialog box opens.



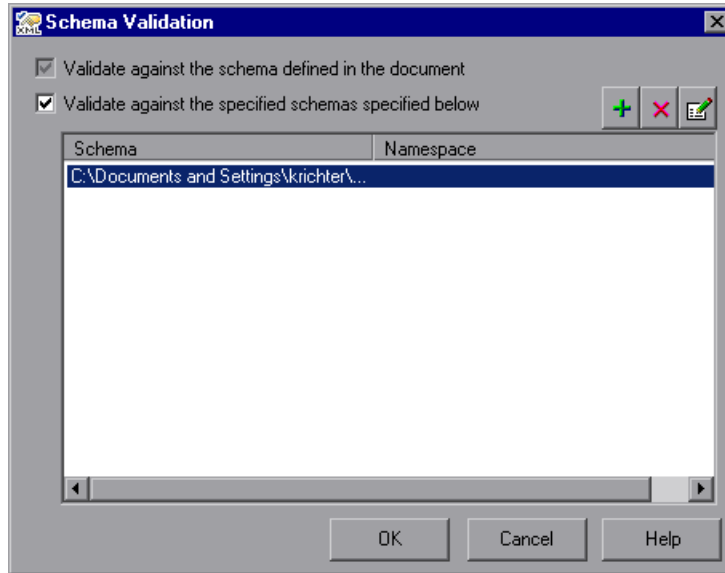
- 4** Select **Update checkpoint properties** and click **OK**. QuickTest runs the test and updates the XML hierarchy for each XML checkpoint.
- 5** If you want to confirm that QuickTest successfully updated your checkpoint, expand the tree in the Test Results window and select the XML checkpoint. Then check that **Update done** is displayed in the right-hand pane. (If the Test Results window did not open automatically at the end of the run, click the **Results** button or choose **Automation > Results**.)



Tip: When you have completed updating your XML tree using this method, it is recommended to exit the update run mode by clicking the **Update Run Mode** button again.

Understanding the Schema Validation Dialog Box

The Schema Validation dialog box enables you to specify an XML schema against which you want to validate the hierarchy of the XML in your application or file.



The Schema Validation dialog box contains the following options:




- **Validate against the schema defined in the document.** Instructs QuickTest to use the schema or schemas defined within your XML document to validate the hierarchy of the XML in your Web page/frame, XML file, or XML test object.
- **Validate against the specified schemas specified below.** Instructs QuickTest to use one or more external XML schema files to validate the hierarchy of your XML. If you select this option, any schemas defined within your XML document are also checked. (The **Validate against the schema defined in the document** is automatically selected and is disabled.)

When you select the **Validate against the specified schemas specified below** option, the **Add Schema** button is enabled. Clicking this button opens the Add Schema dialog box, in which you can specify the following:

- **Schema path or URL.** Enter the path or URL of your XML schema file. Alternatively, click the browse button to navigate to the XML schema you want to use to validate the XML in your Web page/frame, XML file, or XML test object. You can specify schema files either from your file system or from Quality Center. For each external file you add, you must specify its path or URL and namespace.
- **Schema namespace.** (If applicable.) If your schema file has a namespace, specify it. QuickTest checks that the namespace matches the schema file as part of the validation process. If the schema file has a namespace and you do not specify it, or if the namespace you specify is different than the one specified in the schema file, the validation will fail.

Click **OK** in the Add Schema dialog box to add the selected schema to the list in the Schema Validation dialog box. Click the **Add Schema** button again if you want to add another schema.

If you select **Use external schemas**, the following toolbar buttons are enabled when appropriate:

Button	Description
	Enables you to add an external schema file to the list. For more information, see “Understanding the Add Schema Dialog Box” on page 341.
	Enables you to modify the details of the selected external schema file in the list. For more information, see “Understanding the Edit Schema Dialog Box” on page 341.
	Enables you to remove the selected external schema file from the list.

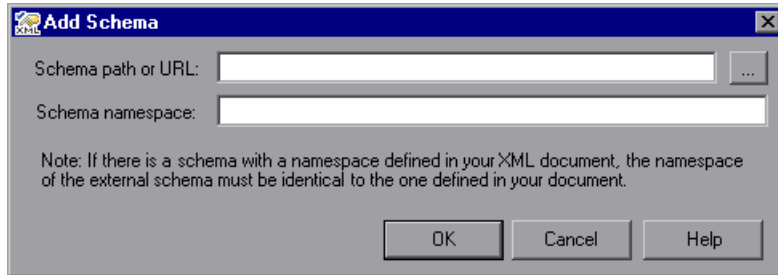
Guidelines for Schema Validation

Following are specific guidelines to consider when specifying a schema file to validate your XML.

- If you are validating an XML file using a schema defined in the XML file, the schema can be defined with an absolute or relative path. When you specify a relative path, QuickTest searches for the schema in the folders listed in the Folders tab of the Options dialog box. For more information, see “Setting Folder Testing Options” on page 712.
- If you are validating an XML document located on the Web with a schema file located on your file system, you cannot use UNC format (for example, `\\ComputerName\Path\To\Schema`) to specify the schema file location. Instead, map the schema file location to a network drive.
- If there is a schema with a namespace defined in your XML document, the namespace of the external schema must be identical to the one defined in your document. Using an external XML schema file to validate an XML document may cause an unexpected result if the XML document has an XML schema declaration, and the namespace in the external schema file and the schema defined in the document are not identical.
- When you perform a schema validation, QuickTest validates all of the elements in the XML document, even if certain XML elements are not associated with a schema file. Any XML elements that are not associated with a schema file will cause the schema validation to fail.

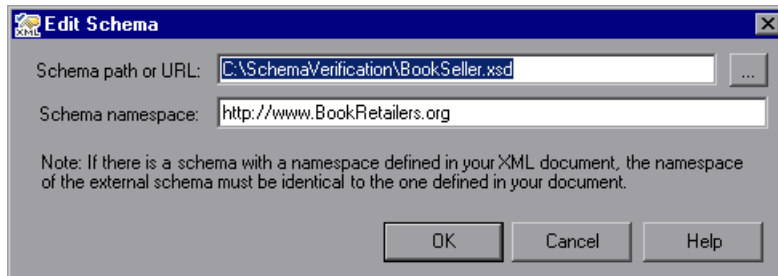
Understanding the Add Schema Dialog Box

The Add Schema dialog box enables you to specify the path or URL of an external schema file and its namespace. If there is a schema with a namespace defined in your XML document, the namespace of the external schema must be identical to the one defined in your document.



Understanding the Edit Schema Dialog Box

The Edit Schema dialog box displays the path and namespace of the schema file you selected in the list. You can modify the path or URL of the selected schema file, and its namespace.



Modifying XML Checkpoints

You can change the expected data and settings of an existing XML checkpoint.

To modify an XML checkpoint:

- 1 In the Keyword View or the Expert View, right-click the XML checkpoint that you want to modify and select **Checkpoint Properties**. Alternatively, select the step containing the XML checkpoint and choose **Edit > Step Properties > Checkpoint Properties**. The XML Checkpoint Properties dialog box opens.
- 2 Modify the settings as described in the previous sections.

Reviewing XML Checkpoint Results

By adding XML checkpoints to your tests, you can verify that the data and structure in your XML documents, XML files, or XML test objects have not changed unexpectedly. When you run your test, QuickTest compares the expected results of the checkpoint to the actual results of the run session. If the results do not match, the checkpoint fails.

You can view summary results of the XML checkpoint in the Test Results window. You can view detailed results by opening the XML Checkpoint Results window. For more information on XML checkpoint results, see “Analyzing XML Checkpoint Results” on page 661.

Note: XML Checkpoints on Web service operations compare the expected values of the checkpoint to the actual values returned from the last native Web service operation performed on the test object. If a different Web service operation step is performed prior to the checkpoint, then the checkpoint will fail.

Using XML Objects and Methods to Enhance Your Test

QuickTest provides several scripting methods that you can use with XML data. You can use these scripting methods to retrieve data and return new XML objects from existing XML data. You do this by using the XMLUtil, or WebXML objects to return XML data and then using the supported XMLData objects and methods to manipulate the returned data.

Tip: All XMLData objects and methods are compatible with namespace and XPath standards.

For more information on XML standards, refer to <http://www.w3.org/XML/>

For more information on namespace standards, refer to <http://www.w3.org/TR/1999/REC-xml-names-19990114/>

For more information on XPath standards, refer to <http://www.w3.org/TR/1999/REC-xpath-19991116>

For more information on programming in the Expert View, see Chapter 34, “Working in the Expert View and Function Library Windows.” For more information on XML objects and methods, refer to the Supplemental section of the *QuickTest Professional Object Model Reference*.

15

Configuring Values

QuickTest enables you to configure the values for properties and other items by defining a value as a constant or a parameter. You can also use regular expressions in values to increase the flexibility and adaptability of your tests.

This chapter describes:	On page:
About Configuring Values	345
Configuring Constant and Parameter Values	346
Understanding and Using Regular Expressions	352
Defining Regular Expressions	355

About Configuring Values

Some dialog boxes, such as the Checkpoint Properties dialog boxes, include a **Configure value** area, in which you can define the value for a selected item as a constant or a parameter. In other contexts, such as the Keyword View, Step Generator, and Object Repository window, you can select a value directly and parameterize it or define it as a constant.

- **Constant.** A value that is defined directly in the step and remains unchanged for the duration of the test.
- **Parameter.** A value that is defined or generated separately from the step and is retrieved when the specific step runs. For example, a parameter value may be defined in an external file or generated by QuickTest.


When you define a value as a parameter, you can also specify other settings according to the parameter type. For more information on using parameters in your tests, see Chapter 16, “Parameterizing Values.”

You can edit a constant value in the **Configure value** area. In certain contexts, you can define a constant value using a regular expression.

A regular expression is a string that specifies a complex search phrase. Regular expressions are used to identify objects and text strings with varying values. For example, if the name of a window’s title bar changes according to a file name, you can use a regular expression to identify a window whose title bar has the specified product name, followed by a hyphen, and then any other text.

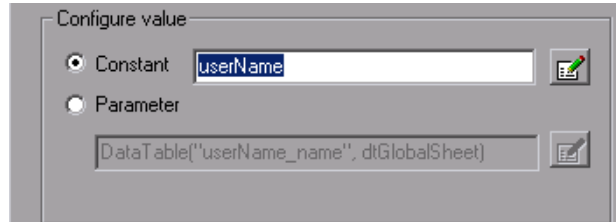
Configuring Constant and Parameter Values

You can define a value as a constant or a parameter:

- in the Value Configuration Options dialog box, by clicking the parameterization button  for a selected value, for example, in the Keyword View, Step Generator, or Object Repository window. For more information, see “Configuring a Selected Value” on page 350.
- in the **Configure value** area of a dialog box, by selecting a property or argument, for example, in the Checkpoint Properties dialog box.

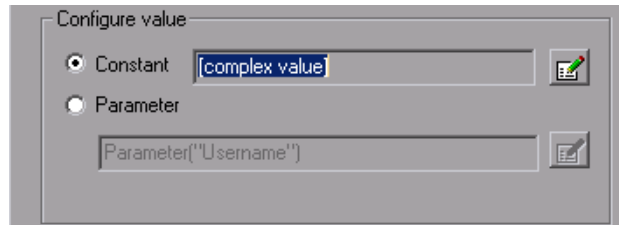
Setting Values in the Configure Value Area

When you select an item in a dialog box containing a **Configure value** area, such as the 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 352.

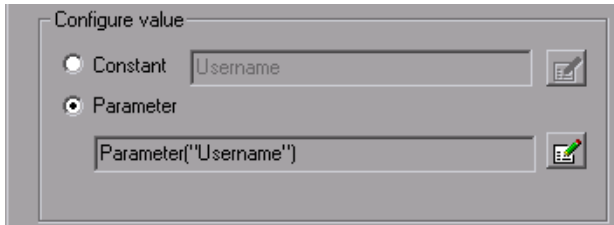
If the entire value cannot be displayed in the **Constant** box, it is shown as **[complex value]**. For example, the value of a list’s **all items** property is a multiline value, where each line contains the value of an item in the list.



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 349.

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 373.



You can click the **Parameter Options** button to select a different parameter type or modify the parameter settings for the value.

The Parameter Options dialog box opens for the displayed parameter type. For more information on defining values for specific parameter types, see:

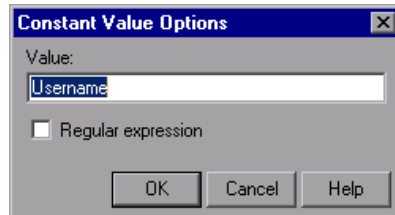
- “Setting Test and Action Parameter Options” on page 376
- “Setting Data Table Parameter Options” on page 381
- “Setting Environment Variable Parameter Options” on page 393
- “Using Random Number Parameters” on page 396

For more information on using parameters in your tests, see Chapter 16, “Parameterizing Values.”

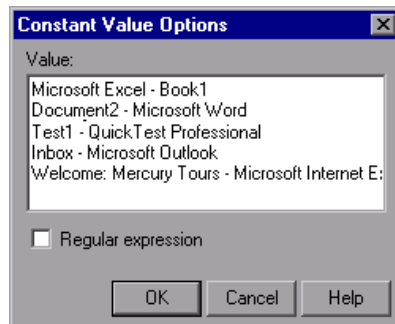
Setting Constant Value Options



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.

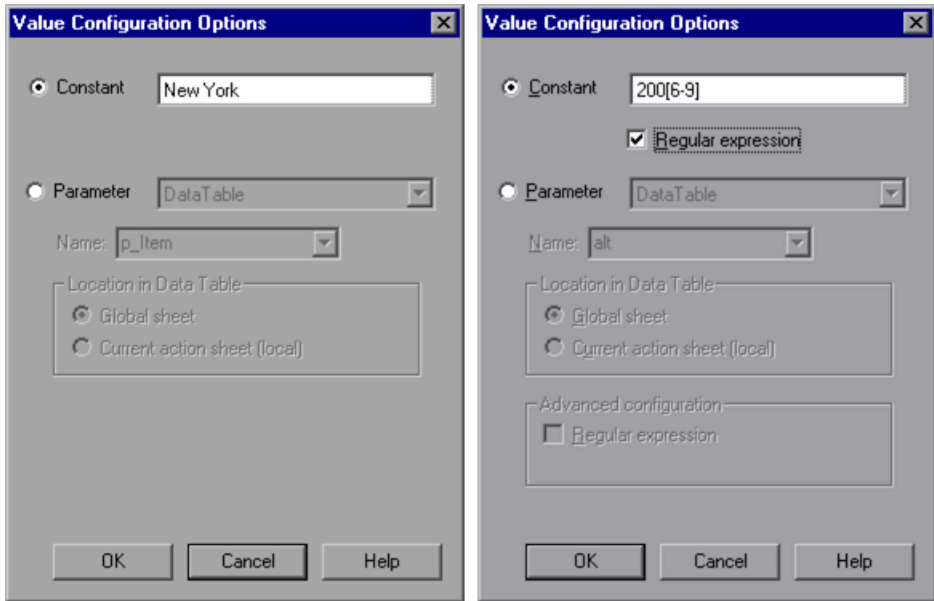


You can update the following options to edit the value of the constant:

- **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 352.
 - For information on defining a regular expression, see “Defining Regular Expressions” on page 355.

Configuring a Selected Value

When you click the parameterization button  for a selected value, the Value Configuration Options dialog box opens. In some situations, you can also define the constant or parameter using a regular expression. (The following examples illustrate the Value Configuration Options dialog box with and without the **Regular expression** check box.)



Note: The parameter options shown in this dialog box change according to the parameter type selected in the **Parameter** box.

You can select one of the following options:

- **Constant.** Defines a value that remains unchanged for the duration of the test. You can edit the value directly in the **Constant** box.

In some situations, for example, when parameterizing an object identification property value, you can also specify a constant value using a regular expression (by using a regular expression in the **Constant** box and selecting the **Regular expression** check box). For information on regular expressions, see “Understanding and Using Regular Expressions” on page 352.

- **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 373.

You can change the default definition by selecting a different parameter type or modifying the parameter settings for the value. The options in the **Parameter** section change according to the parameter type you select.

Note: If you are using an environment variable to parameterize an argument that receives a predefined constant or number, only the environment variable parameters whose value is of type **integer** are shown in the **Name** list.

The **Parameter** section of the Value Configuration Options dialog box is very similar to the Parameter Options dialog box. For more information on configuring values for specific parameter types, see:

- “Defining the Settings for a Test or Action Parameter” on page 377
- “Defining the Settings for a Data Table Parameter” on page 382
- “Defining the Settings for an Environment Variable Parameter” on page 393
- “Defining Settings for a Random Number Parameter” on page 396

For more information on using parameters in your tests, see Chapter 16, “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,” below
- “Using Regular Expressions in Checkpoints” on page 354

For information on defining regular expressions, including regular expression syntax, see “Defining Regular Expressions” on page 355.

Using Regular Expressions for Property Values

If you expect the value of a property to change in a predictable way during each run session, you can use regular expressions when defining or parameterizing property values, for example, in the Object Repository window, or in programmatic descriptions. For more information on programmatic descriptions, see “Using Programmatic Descriptions” on page 1005.

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 346.

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 346.

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 1005.

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 (\) can serve two purposes. It can be used in conjunction with a special character to indicate that the next character be treated as a literal character. For example, \. would be treated as period (.) instead of a wildcard. Alternatively, if the backslash (\) is used in conjunction with some characters that would otherwise be treated as literal characters, such as the letters n, t, w, or d, the combination indicates a special character. For example, \n stands for the newline character.

For example:

- w matches the character w
- \w is a special character that matches any word character including underscore
- \\ matches the literal character \
- \(matches the literal character (

For example, if you were looking for a Web site called:

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 360. For more information on the | regular expression character, see “Matching One of Several Regular Expressions” on page 361.

Matching Any Single Character in a List

Square brackets instruct QuickTest to search for any single character within a list of characters. For example, to search for the date 1967, 1968, or 1969, enter:

196[789]

Matching Any Single Character Not in a List

When a caret (^) is the first character inside square brackets, it instructs QuickTest to match any character in the list except for the ones specified in the string. For example:

```
[^ab]
```

matches any character except a or b.

Note: The caret has this special meaning only when it is displayed first within the brackets.

Matching Any Single Character within a Range

To match a single character within a range, you can use square brackets ([]) with the hyphen (-) character. For instance, to match any year in the 1960s, enter:

```
196[0-9]
```

A hyphen does not signify a range if it is displayed as the first or last character within brackets, or after a caret (^).

For example, [-a-z] matches a hyphen or any lowercase letter.

Note: Within brackets, the characters ".", "*", "[", and "\" are literal. For example, [.*] matches . or *. If the right bracket is the first character in the range, it is also literal.

Matching Zero or More Specific Characters

An asterisk (*) instructs QuickTest to match zero or more occurrences of the preceding character. For example:

`ca*r`

matches `car`, `caaaaaar`, and `cr`.

Matching One or More Specific Characters

A plus sign (+) instructs QuickTest to match one or more occurrences of the preceding character. For example:

`ca+r`

matches `car` and `caaaaaar`, but not `cr`.

Matching Zero or One Specific Character

A question mark (?) instructs QuickTest to match zero or one occurrences of the preceding character. For example:

`ca?r`

matches `car` and `cr`, but nothing else.

Grouping Regular Expressions

Parentheses (()) instruct QuickTest to treat the contained sequence as a unit, just as in mathematics and programming languages.

Using groups is especially useful for delimiting the argument(s) to an alternation operator (|) or a repetition operator (*, +, ?, { }).

Matching One of Several Regular Expressions

A vertical line (|) instructs QuickTest to match one of a choice of expressions. For example:

`foo|bar`

causes QuickTest to match either foo or bar.

`fo(o|b)ar`

causes QuickTest to match either fooar or fobar.

Matching the Beginning of a Line

A caret (^) instructs QuickTest to match the expression only at the start of a line, or after a newline character.

For example:

`book`

matches book within the lines—book, my book, and book list, while

`^book`

matches book only in the lines—book and book list.

Matching the End of a Line

A dollar sign (\$) instructs QuickTest to match the expression only at the end of a line, or before a newline character. For example:

`book`

matches book within the lines—my book, and book list, while a string that is followed by (\$), matches only lines ending in that string. For example:

`book$`

matches book only in the line—my book.

Matching Any AlphaNumeric Character Including the Underscore

`\w` instructs QuickTest to match any alphanumeric character and the underscore (A-Z, a-z, 0-9, `_`).

For example:

`\w*` causes QuickTest to match zero or more occurrences of the alphanumeric characters—A-Z, a-z, 0-9, and the underscore (`_`). It matches `Ab`, `r9Cj`, or `12_uYLgeu_435`.

For example:

`\w{3}` causes QuickTest to match 3 occurrences of the alphanumeric characters A-Z, a-z, 0-9, and the underscore (`_`). It matches `Ab4`, `r9_`, or `z_M`.

Matching Any Non-AlphaNumeric Character

`\W` instructs QuickTest to match any character other than alphanumeric characters and underscores.

For example:

`\W` matches `&`, `*`, `^`, `%`, `$`, and `#`.

Combining Regular Expression Operators

You can combine regular expression operators in a single expression to achieve the exact search criteria you need.

For example, you can combine the `'.'` and `'*'` characters to find zero or more occurrences of any character (except `\n`).

For example,

`start.*`

matches `start`, `started`, `starting`, `starter`, and so forth.

You can use a combination of brackets and an asterisk to limit the search to a combination of non-numeric characters. For example:

`[a-zA-Z]*`

To match any number between 0 and 1200, you need to match numbers with 1 digit, 2 digits, 3 digits, or 4 digits between 1000-1200.

The regular expression below matches any number between 0 and 1200.

`([0-9]?[0-9]?[0-9]|1[01][0-9][0-9]|1200)`

16

Parameterizing Values

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

This chapter describes:	On page:
About Parameterizing Values	365
Parameterizing Values in Steps and Checkpoints	367
Using Test and Action Input Parameters	374
Using Data Table Parameters	378
Using Environment Variable Parameters	385
Using Random Number Parameters	396
Example of a Parameterized Test	398
Using the Data Driver to Parameterize Your Test	403

About Parameterizing Values

You can use the parameter feature in QuickTest to enhance your test by parameterizing the values that it uses. A **parameter** is a variable that is assigned a value from an external data source or generator.

You can parameterize values in steps and checkpoints in your test. You can also parameterize the values of action parameters.

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

There are four types of parameters:

- **Test/action parameters.** Test parameters enable you to use values passed from your test. Action parameters enable you to pass values from other actions in your test.

To use a value within a specific action, you must pass the value down through the action hierarchy of your test to the required action. You can then use that parameter value to parameterize a step in your test. For example, suppose that 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 nested action of Action1), and then parameterize the required step using this action input parameter value (that was passed through from the external application).

Alternatively, you can pass an output action parameter value from an action step to a later sibling action at the same hierarchical level. For example, suppose that Action2, Action3, and Action4 are sibling actions at the same hierarchical level, and that these are all nested actions of Action1. You can parameterize a call to Action4 based on an output value retrieved from Action2 or Action3. You can then use these parameters in your action step.

For more information, see “Guidelines for Working with Action Parameters” on page 871.

- **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 using a list of values so that during each iteration of the run session, QuickTest can identify the different picture buttons.

- **Environment variable parameters.** Enable you to use variable values from other sources during the run session. These may be values you supply, or values that QuickTest generates for you based on conditions and options you choose.

For example, you can have QuickTest read all the values for filling in a Web form from an external file, or you can use one of QuickTest's built-in environment variables to insert current information about the computer running the test.

- **Random number parameters.** Enable you to insert random numbers as values in your test. For example, to check how your application handles small and large ticket orders, you can have QuickTest generate a random number and insert it in a **number of tickets** edit box.

Parameterizing Values in Steps and Checkpoints

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

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 box 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 for a local object, you are modifying the test object description in the local object repository. Therefore, all occurrences of the specified object within the action are parameterized. For more information on the local object repository, see Chapter 6, "Working with 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 with a separate checkpoint for each destination, you can add a Data Table parameter for the destination information. This enables you to create a list of different destinations. QuickTest will then check the flight information for a different destination for each iteration of the test.

For more information on using checkpoints, see Chapter 8, “Understanding Checkpoints.”

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

For more information on using specific parameter types, see:

- “Using Test and Action Input Parameters” on page 374
- “Using Data Table Parameters” on page 378
- “Using Environment Variable Parameters” on page 385
- “Using Random Number Parameters” on page 396

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





- “Parameterizing Values for Operations” on page 369
- “Parameterizing Property Values for Objects and Checkpoints” on page 371


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 541.

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 runs.

Book a Flight: Mercury				
	passFirst0	Set	"Sandra"	Enter "Sandra" in the "passFirst0" edit box
	passLast0	Set	"Herber"	Enter "Herber" in the "passLast0" edit box
	creditnumber	Set	 <RandomNumber(0, 100)>	Enter <the value of a generated random number>

You can parameterize operation values using the parameterization icon  in the **Value** column of the Keyword View.

To parameterize a value for an operation using the parameterization icon:

- 1 In the Keyword View, click in the **Value** column of the required step.
- 2 Click the parameterization icon  for the value that you want to parameterize. The Value Configuration Options dialog box opens, showing the currently defined value.



Note: The parameter options shown in this dialog box change according to the parameter type selected in the **Parameter** box.

- 3 Select **Parameter**. If the value is already parameterized, the **Parameter** section displays the current parameter definition for the value. If the value is not yet parameterized, the **Parameter** section displays the default parameter definition for the value. For more information, see “Understanding Default Parameter Values” on page 373.

4 Accept or change the parameter definition:

- Click **OK** to accept the displayed parameter statement and close the dialog box.
- Modify the value settings for the selected parameter type and click **OK**.
- Change the parameter type. The options in the **Parameter** section change according to the parameter type you select.

For more information on configuring values for specific parameter types, see:

- “Defining the Settings for a Test or Action Parameter” on page 377
- “Defining the Settings for a Data Table Parameter” on page 382
- “Defining the Settings for an Environment Variable Parameter” on page 393
- “Defining Settings for a Random Number Parameter” on page 396

Parameterizing Property Values for Objects and Checkpoints


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

Note: For information on parameterizing a property value for an object in a shared object repository, see Chapter 38, “Managing Object Repositories.”

To parameterize local object values:

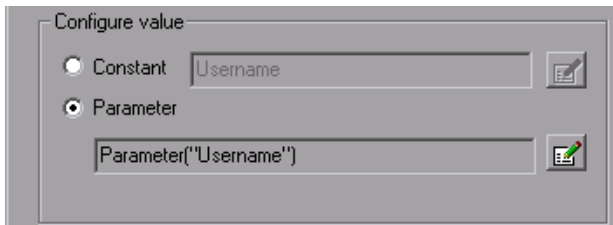
- 1** Open the dialog box for the object properties in one of the following ways:
 - Select a step and choose **Edit > Step Properties > Object Properties**, or right-click a step and choose **Object Properties**. The Object Properties dialog box opens.
 - Choose **Resources > Object Repository**, click the **Object Repository** toolbar button, or right-click the action containing the object and choose **Object Repository**. The Object Repository dialog box opens.



- 2 Click in the **Value** cell for the property that you want to parameterize, and click the parameterization icon . The Value Configuration Options dialog box opens.
- 3 Select **Parameter**. If the value is already parameterized, the **Parameter** box displays the current parameter definition for the value. If the value is not yet parameterized, the **Parameter** box displays the default parameter definition for the value. For more information, see “Configuring a Selected Value” on page 350.
- 4 Click **OK** to accept the displayed parameter statement or change the displayed parameter definition, and then click **OK**.
- 5 To accept the displayed parameter statement and parameterize another of the displayed values, select another property and follow the previous steps.

To parameterize checkpoint property values:

- 1 Open the dialog box for the checkpoint properties by choosing **Edit > Step Properties > Checkpoint Properties**, or right-click the checkpoint and choose **Checkpoint Properties**.
- 2 In the **Configure value** area of the dialog box, select **Parameter**.



If the value is already parameterized, the **Parameter** box displays the current parameter definition for the value. If the value is not yet parameterized, the **Parameter** box displays the default parameter definition for the value. For more information, see “Understanding Default Parameter Values” on page 373.

3 Accept or change the displayed parameter definition:

- To accept the displayed parameter statement and close the dialog box, click **OK**.
- To change the parameter type or modify the value settings for the selected property, click the **Parameter Options** button. The Parameter Options dialog box opens for the displayed parameter type.

**4** To accept the displayed parameter statement and parameterize another of the displayed values, select another property and follow the previous steps.

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

- “Setting Test and Action Parameter Options” on page 376
- “Setting Data Table Parameter Options” on page 381
- “Choosing Global or Action Data Table Parameters” on page 384
- “Using Random Number Parameters” on page 396

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

When parameterizing	Condition	Default parameter type	Default parameter name
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

If the relevant condition described above is not true, the default parameter type is Data Table. If you accept the default parameter details, QuickTest creates a new Data Table parameter with a name based on the selected value. Data Table parameters are created in the Global sheet.

For more information on Data Table sheets, see Chapter 20, “Working with Data Tables.”

Using Test and Action Input Parameters

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

You can parameterize a value using a test or action parameter only if the parameter has been defined for the test or action. For more information on defining parameters, see “Setting Action Parameters” on page 864 and “Setting Action Call Parameter Values” on page 875.

You can parameterize steps by selecting input parameters in the Parameter Options or Value Configuration Options dialog box. The parameter options that are available in these dialog boxes depend on where you are currently located in your test, and whether test or action parameters are defined. For more information, see “Using Action Parameters” on page 868 and “Defining Parameters for Your Test” on page 771.

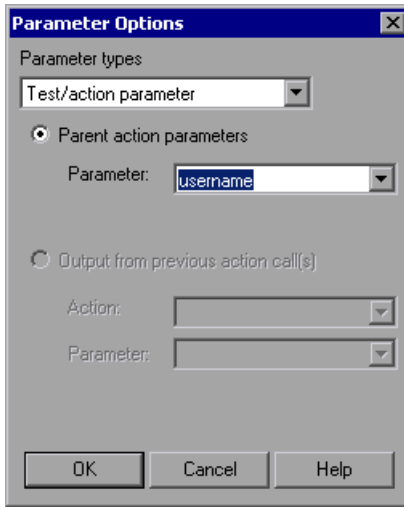
Alternatively, you can enter the parameter name in the Expert View using the **Parameter** utility object, in the format: `Parameter("ParameterName")` for the current action, or `Parameter("ActionName", "ParameterName")` to use the output parameter from a previous action as an input parameter in the current action. For more information, see “Using Action Parameters in Steps in the Expert View” on page 377.

Tip: You can also create test or action parameter output values that retrieve values during the run session and store them for use at another point in the run session. You can then use these output values to parameterize a step in your test. For more information, see “Outputting a Value to an Action Parameter” on page 426.

Setting Test and Action Parameter Options



When **Test/action parameter** is selected as the parameter type, you can select the required parameter in the Parameter Options dialog box. You open the Parameter Options dialog box by clicking the **Parameter Options** button in any Checkpoint Properties 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/action parameter**. For more information on default parameter type settings, see “Understanding Default Parameter Values” on page 373.

Defining the Settings for a Test or Action Parameter

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

- **Test parameters** or **Parent action parameters**. Parameter defined in the test or parent action. (If no output parameters are defined in the test or parent action, this area is disabled.) **Test parameters** are available only for top-level actions. They are defined in the Parameters tab of the Test Settings dialog box. **Parent action parameters** are available for subsequent steps and for nested actions. They are defined in the action containing the steps or in the action that calls the nested action.
- **Parameter**. Specifies the name of the input parameter. The read-only list of available parameters contains the names and full descriptions of the currently defined input parameters for the action. You can resize the display, as needed, and, if the list of parameters is long, you can scroll through the list.
- **Output from previous action call(s)**. Any previous action in the same hierarchical level for which output parameters are defined. (If no output parameters are defined in previous actions, this area is disabled.)
 - **Action**. Specifies the previous action from which you can choose an output parameter. You can choose any action in the list.
 - **Parameter**. Specifies the name of the output parameter. The read-only list of available parameters contains the names and full descriptions of the currently defined output parameters from the previous action(s). You can resize the display, as needed, and, if the list of parameters is long, you can scroll through the list.

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

Using Action Parameters in Steps in the Expert View

Instead of selecting input (or output) parameters from the appropriate dialog boxes while parameterizing steps or inserting output value steps, you can enter input and output parameters as values in the Expert View using the **Parameter** utility object in the format: `Parameter("ParameterName")`.

Suppose you have test steps that enter information in a form to display a list of purchase orders in a table, and then return the total value of the orders displayed in the table.

You can define input parameters, called **SoldToCode** and **MaterialCode**, for the codes entered in the **Sold to** and **Materials** edit boxes of the form so that the Orders table that is opened is controlled by the input parameter values passed when the test is called.

You can define an output parameter, called **TotalValue**, to store the returned value. The output value (**TotalValue**) could then be returned to the application that called the test.

The example described above might look something like this (parameters are in bold font):

```
Browser("Mercury").Page("List Of Sales").WebEdit("Sold to").
    Set Parameter("SoldToCode")
Browser("Mercury").Page("List Of Sales").WebEdit("Materials").
    Set Parameter("MaterialCode")
Browser("Mercury").Page("List Of Sales").WebButton("Enter").Click
NumTableRows = Browser("Mercury").Page("List Of Sales").
    WebTable("Orders").RowCount
Parameter("TotalValue") = Browser("Mercury").Page("List Of Sales").
    WebTable("Orders").GetCellData(NumTableRows,"Total")
```

Using Data Table Parameters

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

For example, consider the Mercury Tours sample Web site, which enables you to book flight requests. To book a flight, you supply the flight itinerary and click the **Continue** button. The site returns the available flights for the requested itinerary.

You could conduct the test by accessing the Web site and recording the submission of numerous queries. This is a slow, laborious, and inefficient solution. By using Data Table parameters, you can run the test for multiple queries in succession.

When you parameterize your test, you first record steps that access the Web site and check 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 to the relevant sheet of the Data Table, one for each itinerary.

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

When you create a new Data Table parameter, a new column is added in the Data Table and the current value you parameterized is placed in the first row. If you parameterize a value and select an existing Data Table parameter, then the values in the column for the selected parameter are retained, and are not overwritten by the current value of the parameter.

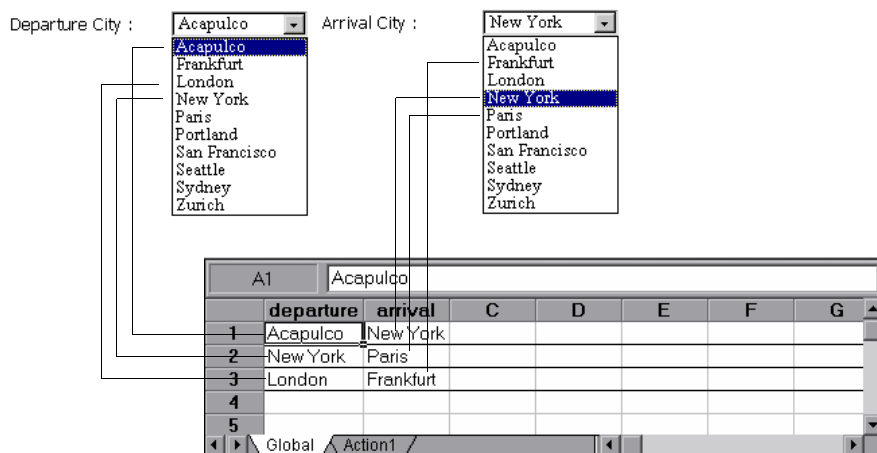
Each column in the table represents the list of values for a single Data Table parameter. The column header is the parameter name.

Each row in the table represents a set of values that QuickTest submits for all the parameters during a single iteration of the test. When you run your test, QuickTest runs one iteration of the test for each row of data in the table. For example, a test with ten rows in the Global sheet of the Data Table will run ten times.

For more information on entering values in the Data Table, see Chapter 20, “Working with Data Tables.”

Tip: You can also create Data Table output values, which retrieve values during the run session and insert them into a column in the Data Table. You can then use these columns as Data Table parameters in your test. For more information, see Chapter 17, “Outputting Values.”

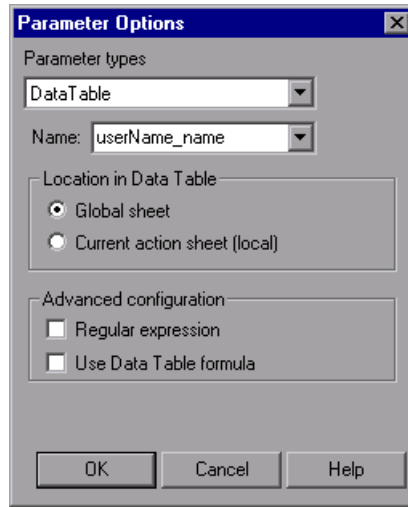
In the previous example, QuickTest submits a separate query for each itinerary when you run the test.



Setting Data Table Parameter Options



When **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. You open the Parameter Options dialog box by clicking the **Parameter Options** button in any Checkpoint Properties 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, **Data Table** may be set as the default parameter type. For more information on default parameter type settings, see “Understanding Default Parameter Values” on page 373.

Defining the Settings for a Data Table Parameter

The following options are available for configuring Data Table parameters:

Name. Specifies the name of the parameter in the Data Table. You can create a new parameter by using the default parameter name or entering a new, descriptive name. Alternatively, you can select an existing Data Table parameter from the list.

Note: The parameter name must be unique in the sheet. It can contain letters, numbers, periods, and underscores. The first character of the parameter name must be a letter or an underscore. If you specify an invalid name, QuickTest displays a warning message when you click **OK**. You can choose to edit the name manually or to instruct QuickTest to fix the name automatically (by adding an underscore at the beginning of the name).

Location in Data Table. Specifies whether to store the parameter in the global or current action sheet in the Data Table.

For more information on global and action Data Table parameters, see “Choosing Global or Action Data Table Parameters” on page 384. For more information on actions, see Chapter 18, “Working with Actions” and Chapter 30, “Working with Advanced Action Features.”

Advanced configuration (if applicable):

- **Regular expression.** Sets the value of the parameter as a regular expression. For more information, see “Understanding and Using Regular Expressions” on page 352. 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 534.

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

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 17, “Outputting Values.”

For more information on setting global iteration preferences, see “Defining Run Settings for Your Test” on page 763.

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 Data Table 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 on setting action iteration preferences, see “Inserting a Call to an Existing Action” on page 861.

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 678.

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 20, “Working with Data Tables.”

There are several types of environment variables:

- **User-Defined Internal.** Variables that you define within the test. These variables are saved with the test and are accessible only within the test in which they were defined.

You can create or modify internal, user-defined environment variables for your test in the Environment 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” on page 774.

For information on creating or modifying environment variables in the Parameter Options dialog box, see “Setting Environment Variable Parameter Options” on page 393.

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 17, “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 387.
- **Built-in.** Variables that represent information about the test and the computer on which the test is run, such as **Test path** and **Operating system**. These variables are accessible from all tests, and are designated as read-only. For more information, see “Using Built-in Environment Variables” on page 390.

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 Record and Run Details for Your Test” on page 802.

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 dialog box and use the **Export** button to create the file with the correct structure. For more information on exporting environment variables, see Chapter 26, “Setting Options for Individual Tests.”

Notes:

You can also store environment variable files in Quality Center. For more information, see “Using Environment Variable Files with Quality Center” on page 389.

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\QTEEnvironment.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 **File > Settings** to open the Test Settings dialog box. For more information on the Test Settings dialog box, see Chapter 26, “Setting Options for Individual Tests.”
- 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 393.

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 Quality Center. Thus, once you save the file to the project, changes made to the Quality Center environment variable file will not affect the file system file and vice versa.

To use an environment variable file with Quality Center:

- 1** To add a new environment variable file, create a new **.xml** file in your file system, as described in “Using User-Defined External Environment Variables” on page 387.
- 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 1261.
- 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 45, “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 393.

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	<p>The path of the folder in which the current test results are located.</p> <p>Note: You cannot use the ResultDir environment variable when running a test from Business Availability Center, LoadRunner, or the Silent Test Runner in QuickTest.</p>
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.

Name	Description
UpdatingActiveScreen	Indicates whether the Active Screen images and values are being updated during the update run process. For more information, see “Updating a Test” on page 616.
UpdatingCheckpoints	Indicates whether checkpoints are being updated during the update run process. For more information, see “Updating a Test” on page 616.
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” on page 616.
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. You open the Parameter Options dialog box by clicking the **Parameter Options** button in any Checkpoint Properties dialog box. 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.

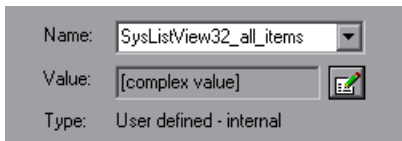
Notes:

If you edit the name displayed in the **Name** box for an existing parameter, you create a new internal user-defined environment variable parameter. The original environment variable parameter is not modified.

If you are parameterizing an argument that receives a predefined constant or number, only the environment variable parameters whose value is of type **integer** are shown in the **Name** list.

- **Value.** Specifies the value of the parameter. You can enter the value for a new user-defined internal parameter, or modify the value for an existing user-defined internal parameter. External and built-in environment variable parameter values cannot be modified in this dialog box.

If the entire value of a selected environment variable parameter cannot be displayed in the **Value** box, it is shown as **[complex value]**. For example, the value of a list's **all items** property is a multi-line value, where each line contains the value of an item in the list.



You can view or edit a complex value by clicking the **View/Edit Complex Value** button. For more information, see “Viewing and Editing Complex Parameter Values” on page 395.

- **Type.** Specifies the type of environment variable parameter (read-only):
 - **internal user-defined**
 - **external user-defined**
 - **built-in**

Tip: The value of an environment variable remains the same throughout the test run, regardless of the number of iterations, unless you change the value of the variable programmatically in your script.

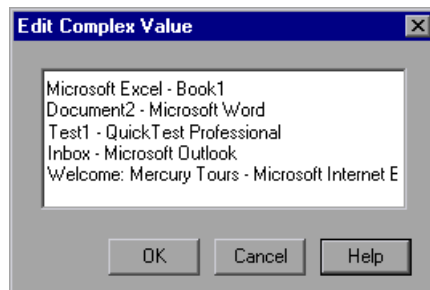
- **Regular expression.** Sets the value of the parameter as a regular expression. This option is available only when parameterizing a checkpoint or object property text string value, and the selected environment variable parameter type is **internal user-defined**. For more information on regular expressions, see “Understanding and Using Regular Expressions” on page 352.

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

Viewing and Editing Complex Parameter Values



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



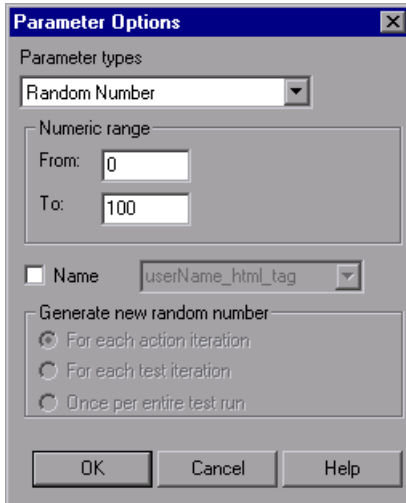
You can edit the value for an internal user-defined environment variable parameter.

For an external or built-in environment variable parameter, you can view the value but you cannot modify it in this dialog box.

Using Random Number Parameters



When you select **Random Number** as the parameter type, the Parameter Options dialog box enables you to configure your parameter to use random numbers. You open the Parameter Options dialog box by clicking the **Parameter Options** button in any Checkpoint Properties dialog box. 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.

The following options are available for configuring random number parameters:

- **Numeric range.** Specifies the range from which the random number is generated. By default, the random number range is between 0 and 100. You can modify the range by entering different values in the **From** and **To** boxes. The range must be between 0 and 2147483647 (inclusive).
- **Name.** Assigns a name to your parameter. Assigning a name to a random parameter enables you to use the same parameter several times in your test. You can select an existing named parameter or create a new named parameter by entering a new, descriptive name.
- **Generate new random number.** Defines the generation timing for a named random parameter. This box is enabled when you select the **Name** check box. You can select one of the following options:
 - **For each action iteration.** Generates a new number at the end of each action iteration.
 - **For each test iteration.** Generates a new number at the end of each global iteration.
 - **Once per entire test run.** Generates a new number the first time the parameter is used. The same number is used for the parameter throughout the test run.

Notes:

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 *Mercury QuickTest Professional Object Model Reference*.

Example of a Parameterized Test

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

When you test your application 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.

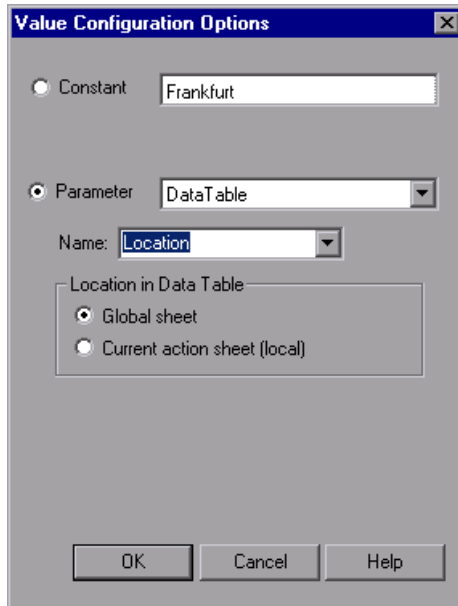
Item	Operation	Value	Documentation
▼ Action1			
Welcome: Mercury Tours			
Welcome: Mercury Tours			
userName	Set	"mercury"	Enter "mercury" in the "userName" edit box.
password	SetSecure	"435b8eb45e4fd...	Enter the encrypted string "435b8eb45e4fd8dd653c4d65fc1aa90e5c...
Sign-In	Click	26,8	Click the "Sign-In" image.
Find a Flight: Mercury			
fromPort	Select	"Frankfurt"	Select the "Frankfurt" item from the "fromPort" list.
fromMonth	Select	"Dec"	Select the "Dec" item from the "fromMonth" list.
fromDay	Select	"29"	Select the "29" item from the "fromDay" list.
toPort	Select	"Acapulco"	Select the "Acapulco" item from the "toPort" list.
toMonth	Select	"Dec"	Select the "Dec" item from the "toMonth" list.
toDay	Select	"31"	Select the "31" item from the "toDay" list.
servClass	Select	"Business"	Select the "Business" radio button in the "servClass" radio button group.
findFlights	Click	37,5	Click the "findFlights" image.
Select a Flight: Mercury			
reserveFlights	Click	63,12	Click the "reserveFlights" image.
Book a Flight: Mercury			
passFirst0	Set	"Tom"	Enter "Tom" in the "passFirst0" edit box.
passLast0	Set	"Smith"	Enter "Smith" in the "passLast0" edit box.
creditnumber	Set	"5456194"	Enter "5456194" in the "creditnumber" edit box.

Parameterize a Step

Parameterize the method argument of the **fromPort** step:

 fromPort Select "Frankfurt" Select the "Frankfurt" item in the "fromPort" list.

In the Keyword View, click in the **Value** cell of the step and then click the parameterization icon . In the Value Configuration Options dialog box, select the **Parameter** radio button. In the **Name** box, rename p_item to **Location**.



Click **OK**. The **Location** column is added to the Data Table.

For more information on parameterizing a step, see “Parameterizing Values in Steps and Checkpoints” on page 367.

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:

SELECT FLIGHT

✈

Select your departure and return flight from the selections below. Your total price will be higher than quoted if you elect to fly on a different airline for both legs of your travel.

DEPART

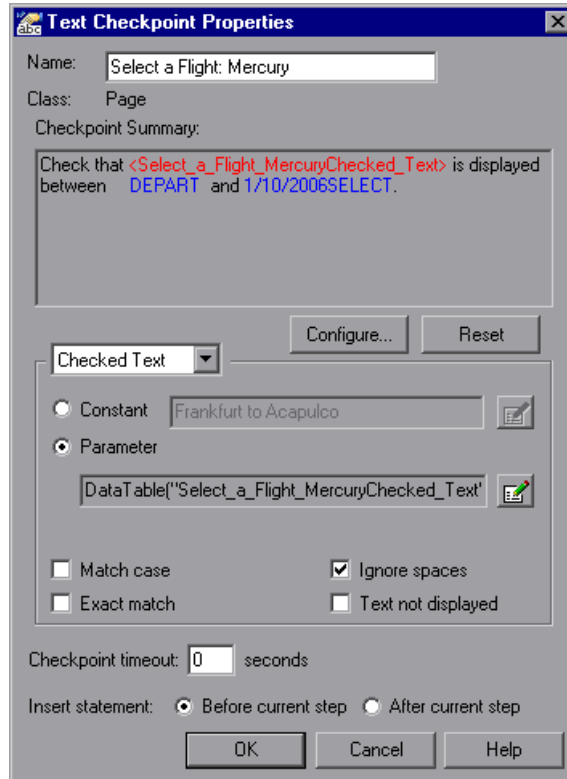
Frankfurt to Acapulco
03/09/2004

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 Data Table parameter to **Check_Locations_Text**. Click **OK** in the Parameter Options dialog box and in the Text Checkpoint Properties dialog box. A **Check_Locations_Text** column is added to the Data Table.



For more information on parameterizing a checkpoint, see “Parameterizing Values in Steps and Checkpoints” on page 367.

Enter Data in the Data Table

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

	Location	Check_Locations_Text	C	D
1	Frankfurt	Frankfurt to Acapulco		
2	Acapulco	Acapulco to Frankfurt		
3				
4				
5				

For more information on Data Tables, see Chapter 20, “Working with Data Tables.”

Modified Test

The following example shows the test after parameterizing the step and creating a parameterized text checkpoint.

Welcome: Mercury T...			
userName	Set	"mercury"	Enter "mercury" in the "userName" edit box.
password	SetS...	"404ee5e998b25bdc6f116b031452..."	Enter the encrypted string "404ee5e998b25bdc6f116b031452..."
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 text in the "Frankfurt to Acapulco" checkpoint.
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:

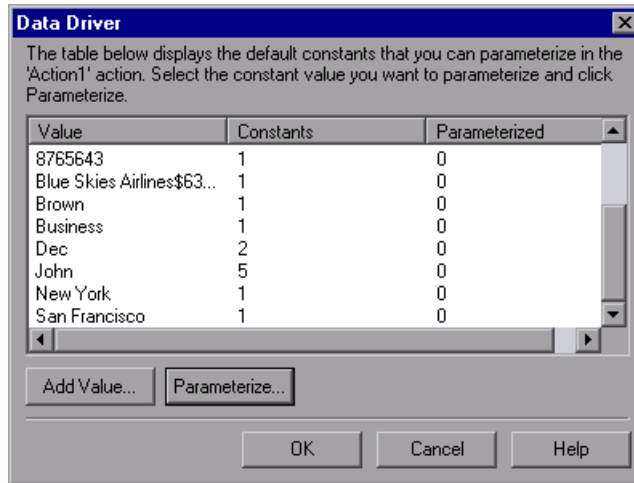
When finding multiple occurrences of a selected value, QuickTest conducts a search that is case sensitive and searches only for exact matches. (It does not find values that include the selected value as part of a longer string.)

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.

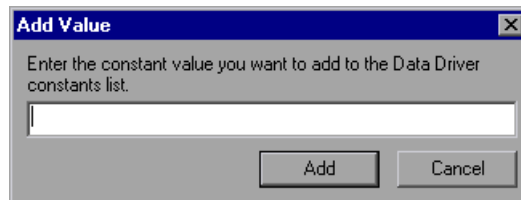
The Data Driver displays the Constants list for the action. For each constant value, it displays the number of times the constant value appears in the action.

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

For more information on how to work with testing methods, see Chapter 34, “Working in the Expert View and Function Library Windows.” For syntax and method information, refer to the *Mercury QuickTest Professional Object Model Reference*.

Note: If you chose not to wait for the constants to load, the Data Driver opens with an empty Constants table. You can add the constant values that you want to parameterize to the Data Driver, as described below.

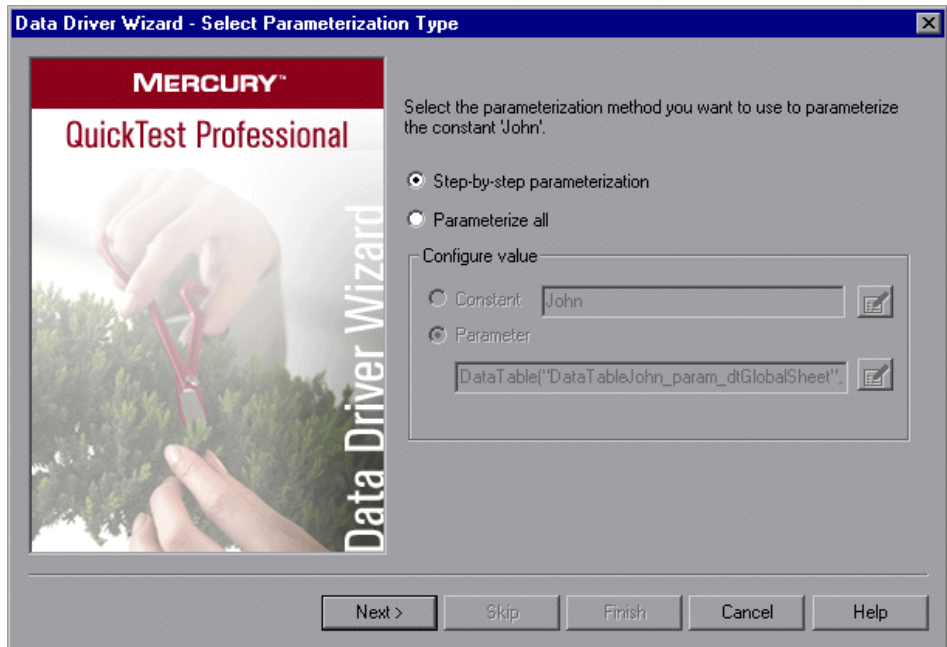
- 3 If you want to parameterize a value that is not currently displayed in the list (such as an object property value), click **Add Value**. The Add Value dialog box opens.



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

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

- 4 Select the value you want to parameterize from the Constants list and click **Parameterize**. The Data Driver Wizard opens.

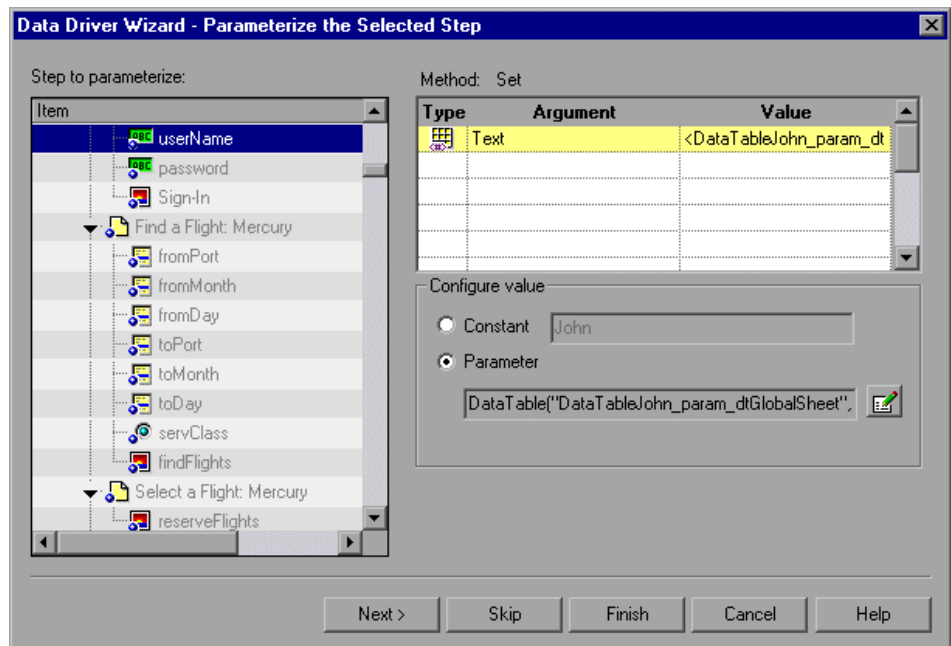


- 5 Select the type of parameterization you want to perform:
 - **Step-by-step parameterization.** Enables you to view the current values of each step containing the selected value. For each step, you can choose whether or not to parameterize the value and if so, which parameterization options you want to use.
 - **Parameterize all.** Enables you to parameterize all occurrences of the selected value throughout the action. You set your parameterization preferences one time and the same options are applied to all occurrences of the value.

- 6 If you selected **Step-by-step parameterization**, click **Next**. The Parameterize the Selected Step screen opens.

If you selected **Parameterize all**, the **Parameter** option is enabled in the **Configure value** area. Select your parameterization preferences the same way that you would for an individual step. For more information, see “Parameterizing Values in Steps and Checkpoints” on page 367. 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 373.



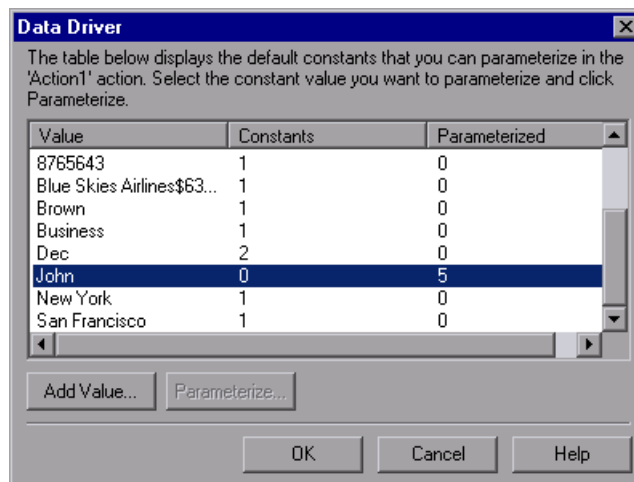
Accept the default parameterization settings or click the **Parameter Options** button to set the parameterization options you want to apply to this step. For more information, see “Parameterizing Values in Steps and Checkpoints” on page 367.

- 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.

17

Outputting Values

QuickTest enables you to retrieve values in your test and 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:	On page:
About Outputting Values	411
Creating Output Values	412
Outputting Property Values	419
Specifying the Output Type and Settings	425
Outputting Text Values	430
Outputting Table Values	436
Outputting Database Values	450
Outputting XML Values	454
Updating the XML Hierarchy for XML Test Object Operation Output Value Steps (For WebService Test Objects Only)	466

About Outputting Values

An **output value** is a step in which one or more values are captured at a specific point in your test and stored for the duration of the run session. The values can later be used as input at a different point in the run session.

You can output the property values of any object. You can also output values from text strings, table cells, databases, and XML documents.

When you create output value steps, you can determine where the values are stored during the run session and how they can be used. During the run session, QuickTest retrieves each value at the specified point and stores it in the specified location. When the value is needed later in the run session, QuickTest retrieves it from this location and uses it as required.

Output values are stored only for the duration of the run session. When the run session is repeated, the output values are reset.

Note: After the run session, you can view the output values retrieved during the session as part of the session results. For more information, see “Viewing Parameterized Values and Output Value Results” on page 675.

Creating Output Values

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

You can create the following categories of output values:

- Standard output values
- Text output values
- Table output values
- Database output values
- XML output values

Standard Output Values

You can use standard output values to output the property values of most objects. For example, in a Web-based application, the number of links on a Web page may vary based on the selections a user makes on a form on the previous page. You could create an output value in your test to store the number of links on the page.

Note: You can also use standard output values to output the contents of table cells. For more information, see “Table Output Values” on page 414.

Tip: You can use standard output values to output text strings by specifying the **text** property of the object as an output value.

For more information on standard output values, see “Outputting Property Values” on page 419.

Text Output Values

You can use text output values to output text strings displayed on a Web page or application. When creating a text output value, you can output a part of the object’s text. You can also specify the text before and after the output text.

For example, suppose that you want to store the text of any error message that appears after a specific step in the Web application you are testing. Inside the If statement, you check whether a window exists with a known title bar value, for example Error. If it exists, you output the text in this window (assuming that the window size is the same for all possible error messages).

For more information on text output values, see “Outputting Text Values” on page 430.

Note: QuickTest no longer supports the creation of text output values for checking non-Web-based applications. QuickTest also no longer supports the creation of text area output values. However, if you used an earlier version of QuickTest to create text area output values or text output values, you can still use and edit these text output values (as long as the relevant QuickTest add-in is loaded).

Table Output Values

Table output values are a subset of standard output values, described above. You can use table output values to output the contents of table cells. For some types of tables, you can specify a row range from which to choose the table cells. During the run session, QuickTest retrieves the current data from the specified table cells according to the settings that you specified and outputs the values to the Data Table.

For more information, see “Outputting Table Values” on page 436.

Database Output Values

You can use database output values to output the value of the contents of database cells, based on the results of a query (result set) that you define on a database. You can create output values from the entire contents of the result set, or from a part of it. During the run session, QuickTest retrieves the current data from the database and outputs the values according to the settings that you specified.

For more information, see “Outputting Database Values” on page 450.

XML Output Values

You can use XML output values to output the values of XML elements and attributes in XML documents.

After the run session has finished, you can view summary results of the XML output values in the Test Results window. You can also view detailed results by opening the XML Output Value Results window. For more information, see Chapter 24, “Analyzing Test Results.”

For example, suppose that an XML document in a Web page contains a price list for new cars. You can output the price of a particular car by selecting the appropriate XML element value to output.

For more information on XML output values, see “Outputting XML Values” on page 454.

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 Environments
Standard	S	S	S	S	NA
Page (Standard)	S	NA	NA	NA	NA
Table (Standard)	S	NA	NA	S	NA
Text	S	NS	NS	NS	NS
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 or action parameter
- the run-time Data Table
- an environment variable

Note: Output values are stored only for the duration of the test, and are not saved with the test. If you select to output a value to an existing parameter, Data Table column, or environment variable, the existing value is overwritten when the output value step runs. When the run session ends, the original value is restored.

Storing Values in Test and Action Parameters

You can output a value to an action parameter, so that values from one part of a run session can be used later in the run session, or be passed back to the application that ran (called) the test.

For example, suppose you are testing a shopping application that calculates your purchases and automatically debits your account with the amount that you purchased. You want to test that the application correctly debits the purchase amount from the account each time that the action is run with a different list of items to purchase. You could output the total amount spent to an action parameter value, and then use that value later in your run session in the action that debits the account.

For more information on action parameters in general, see “Using Action Parameters” on page 868.

Storing Values in the Run-time Data Table

The option to output a value to the run-time Data Table is especially useful with a **data-driven** test (or action) that runs several times. In each repetition, or **iteration**, QuickTest retrieves the current value and stores it in the appropriate row in the run-time Data Table.

For example, suppose you are testing a flight reservation application and you design a test to create a new reservation and then view the reservation details. Every time you run the test, the application generates a unique order number for the new reservation. To view the reservation, the application requires the user to input the same order number. You do not know the order number before you run the test.

To solve this problem, you output a value to the Data Table for the unique order number generated when creating a new reservation. Then, in the View Reservation screen, you use the column containing the stored value to insert the output value into the order number input field.

When you run the test, QuickTest retrieves the unique order number generated by the site for the new reservation and enters this output value in the run-time Data Table. When the test reaches the order number input field required to view the reservation, QuickTest inserts the unique order number stored in the run-time Data Table into the order number field.

Storing Values in Environment Variables

When you output a value to an internal user-defined environment variable, you can use the environment variable input parameter at a later stage in the run session.

Note: You can output values only to internal user-defined environment variables and not to external or built-in environment variables, which are read-only.

For example, suppose you are testing an application that prompts the user to input an account number on a Welcome page and then displays the user's name. You can use a text output value to capture the value of the displayed name and store it in an environment variable.

You can then retrieve the value in the environment variable to enter the user's name in other places in the application. For example, in an Order Checkbook Web page, which for security reasons requires users to enter the name to appear on the checks, you could use the value to insert the user's name into the **Name** edit box.

Viewing and Editing Output Values

When you insert an output value step in your test, the Keyword View shows the step with **Output** displayed in the **Operation** column and **CheckPoint** displayed in the **Value** column, followed by the name assigned to the output value.

The output value statement is displayed in the Expert View with the following syntax:

Object.Output CheckPoint(Name)



You can view or edit the output value or its details in the relevant Output Value Properties dialog box, by right-clicking the step and choosing **Output Value Properties**. Alternatively, you can click the step in the **Value** column in the Keyword View and then click the **Output Properties** button.

For more information on the options available in the different Output Value Properties dialog boxes, see:

- “Defining Standard Output Values” on page 421
- “Defining Text Output Values” on page 431
- “Outputting Table Content” on page 440
- “Outputting Table Properties” on page 445
- “Defining Database Output Values” on page 451
- “Understanding the XML Output Properties Dialog Box” on page 462

Outputting Property Values

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

You can create standard output values while recording or editing your test.

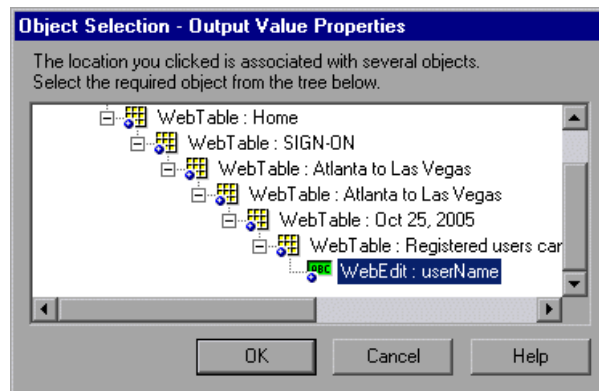
To create standard output values while recording:



- 1** Choose **Insert > Output Value**. Alternatively, you can click the arrow beside the **Insert Checkpoint or Output Value** button on the toolbar.
- 2** Select **Standard Output Value**. The pointer turns into a pointing hand.

Tip: Hold the left CTRL key to change the window focus or perform operations such as right-clicking or moving the pointer over an object to display a context menu. If the window containing the object you want to select is minimized, you can display it by holding the left CTRL key, right-clicking the application in the Windows task bar, and choosing **Restore** from the context menu.

- 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 421. If you selected a **Table** item, see “Outputting Table Content” on page 440 and “Outputting Table Properties” on page 445.
- 6 When you finish defining the output value details, click **OK**. QuickTest inserts an output value step in your test.

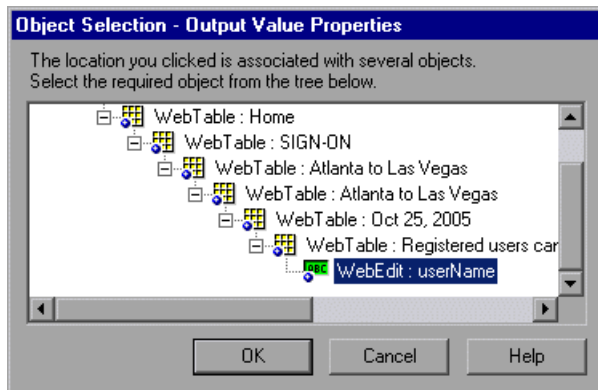
To create standard output values while editing your test:



- 1 Make sure the **Active Screen** button is selected.
- 2 Click a step whose Active Screen contains the object for which you want to specify an output value. The Active Screen displays the captured bitmap or HTML source corresponding to the highlighted step.

For Windows-based applications, make sure that the Active Screen contains property data for the object for which you want to specify an output value. For more information, see “Setting Active Screen Options” on page 715.

- 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.

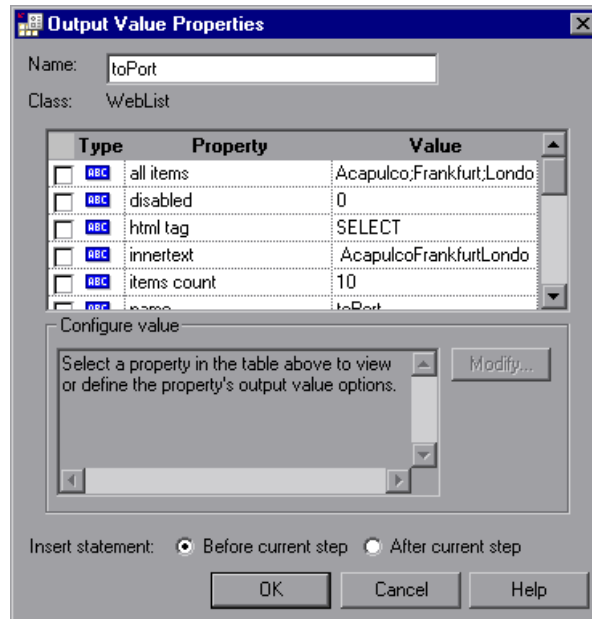




- 5 Select the object for which you want to specify an output value, and click **OK**. The Output Value Properties dialog box opens for the selected object. If you select a **Table** item, the Table Output Value Properties dialog box opens.
- 6 Specify the property values to output and their settings. For more information, see “Defining Standard Output Values” on page 421. If you selected a **Table** item, see “Outputting Table Content” on page 440 and “Outputting Table Properties” on page 445.
- 7 When you finish defining the output value details, click **OK**. QuickTest inserts an output value step in your test.

Defining Standard Output Values

The Output Value Properties dialog box enables you to choose which property values to output and to define the settings for each value that you select.



Note: If you insert an output value on a Web page, the Page Output Value Properties dialog box opens. This dialog box is identical to the Output Value Properties dialog box, except that it contains two additional option areas, **HTML verification** and **All objects in page**. These options are relevant only for checkpoints and are disabled when defining output values.

You can select a number of properties for the same object and define the output settings for each property value before closing the dialog box. When the output value step is reached during the run session, QuickTest retrieves all of the specified property values.





Identifying the Output Value

The top part of the dialog box displays information on the output value:

Item	Description
Name	<p>The name that QuickTest assigns to the output value. By default, the output value name is the name of the test object for which you are performing the output value step. You can specify a different name for the output value or accept the default name.</p> <p>If you rename the output value, make sure that the name is unique, does not begin or end with a space, and does not contain the following character/combination of characters:</p> <p>" := @@</p>
Class	<p>The type of test object. In this example, the WebList class indicates it is a list object in a Web application.</p>

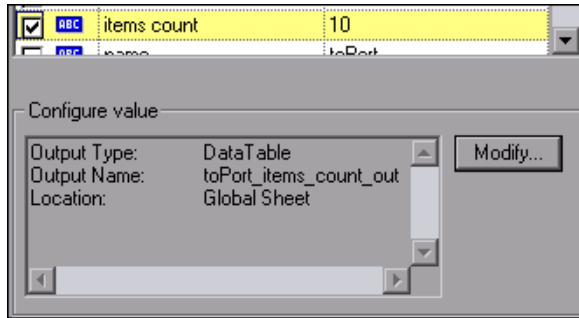
Selecting the Property Values to Output

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	<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 stored in a test or action parameter.</p> <p>The  icon indicates that the value of the property is currently stored in the run-time Data Table.</p> <p>The  icon indicates that the value of the property is currently stored in an environment variable.</p>
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 425.

When you select a property value to output, you can:

- change the output type and/or settings for the selected value by clicking the **Modify** button. The Output Options dialog box opens and displays the current output type and settings for the value. For more information, see “Specifying the Output Type and Settings” on page 425.
- accept the displayed output definition by selecting another property value or by clicking **OK**.

Specifying the Location for the Output Value Step

If the **Insert statement** area is displayed at the bottom of the dialog box, you can specify where the new output value step should be inserted in your test. For more information, see “Selecting the Location for the Output Value Step” on page 429.

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 in the Output Value Properties dialog box.

Understanding Default Output Definitions

When you initially select a value for output, QuickTest generates a default output definition for the value.

When you output a value for a step in an action (for a test):

- If at least one output parameter is defined in the action, the default output type is **Test/action parameter** and the default output name is the first output parameter displayed in the Action Properties dialog box.
- If no output parameters are defined in the action, the default output type is Data Table and QuickTest creates a new Data Table output name based on the selected value.

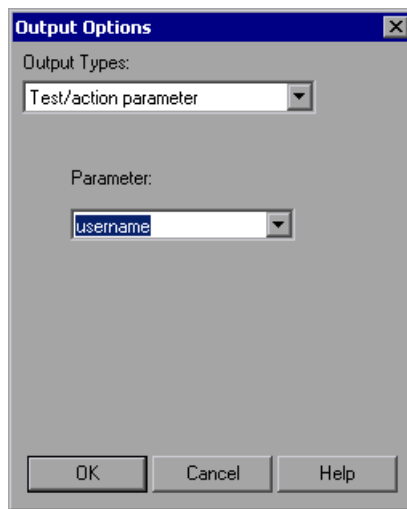
The output value is created in the Global sheet of the Data Table. For more information on creating output parameters for actions, see “Removing Actions from a Test” on page 499.

For more information on Data Table sheets, see Chapter 20, “Working with Data Tables.”

Outputting a Value to an Action Parameter

You can output a value to an action parameter, so that the values can be used later in the run session, or the values can be passed back to the external application that ran (called) the test. You can only output a value to an action parameter if the parameter has been defined as an output parameter for the calling action. You open the Output Options dialog box by clicking the **Modify** button in any Output Value Properties dialog box.

When **Test/action parameter** is selected as the output type, the Output Options dialog box enables you to select the parameter in which to store the selected value for the duration of the run session.

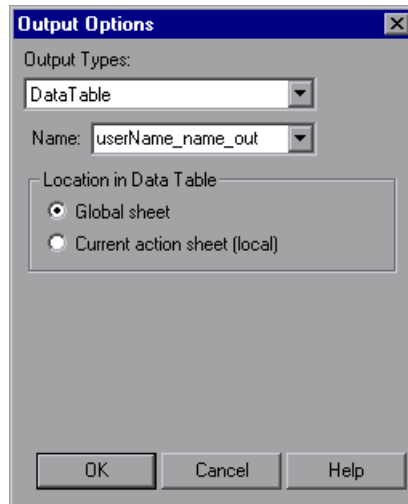


Tip: When you open the Output Options dialog box, QuickTest may display **Test/action parameter** as the default output type. This occurs if at least one output parameter is defined in the action.

The **Parameter** box specifies the name of the parameter in which to store the output value. The read-only list of available parameters contains the names and full descriptions of the currently defined output parameters for the action. You can resize the display, as needed, and, if the list of parameters is long, you can scroll through the list.

Outputting a Value to the Data Table

When **Data Table** is selected as the output type, the Output Options dialog box enables you to specify where to store the selected value within the run-time Data Table. You open the Output Options dialog box by clicking the **Modify** button in any Output Value Properties dialog box.



Tip: When you open the Output Options dialog box, QuickTest may display **Data Table** as the default output type. For more information, see “Understanding Default Output Definitions” on page 425.

The following options are available when outputting a value to the Data Table:

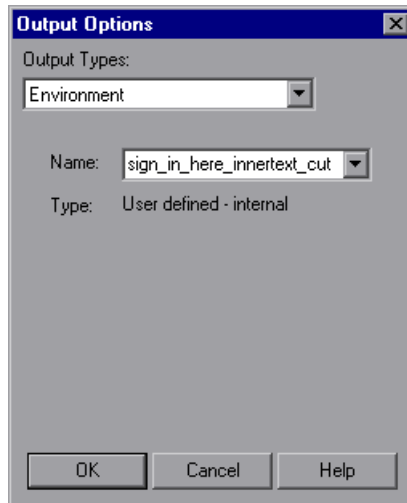
- **Name.** Specifies the name of the column in the Data Table in which to store the value. QuickTest suggests a default name for the output. You can select an existing output name from the list, or create a new output name by using the default output name or entering a valid descriptive name.

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 475. For more information on actions, see Chapter 18, “Working with Actions” and Chapter 30, “Working with Advanced Action Features.”

Outputting a Value to an Environment Variable

When you select **Environment** as the output type, the Output Options dialog box enables you to specify the internal user-defined environment variable in which to store the selected value for the duration of the run session. You open the Output Options dialog box by clicking the **Modify** button in any Output Value Properties dialog box.



The following options are available when outputting a value to an Environment variable:

- **Name.** Specifies the name of the internal user-defined environment variable in which to store the value. The list contains all currently defined internal user-defined environment variables with the corresponding type. You can select an existing variable from the list, or you can create a new internal environment variable by modifying the displayed name or by entering a new, descriptive name.

Note: If you edit the name displayed in the **Name** box for an existing variable, you create a new internal user-defined environment variable. The original environment variable is not modified.

Alternatively, you can output the value to an existing environment variable. If you select an existing variable from the list, QuickTest prompts you to choose whether to overwrite its current value with the new value when the output value step runs.

If you choose not to overwrite the current value of the selected variable, a new environment variable is created with the original variable name and an identifying suffix.

- **Type.** Displays the environment variable type. Since it is not possible to output values to external or built-in environment variables, the type is always **User-defined - internal**.

For more information on environment variables, see “Using Environment Variable Parameters” on page 385.

Selecting the Location for the Output Value Step

When you create output values while editing a test, the **Insert statement** area is displayed at the bottom of the dialog box.

By default, QuickTest inserts the new output value step before the current step (the step you selected when you chose the **Output Value** option). You can instruct QuickTest to insert the new output value step after the current step, by selecting the **After current step** option.

Note: This option is not available while recording. QuickTest automatically inserts the new output value step after the previously recorded step. It is also not available when modifying an existing output value step.

Outputting Text Values

You can create a text output value from a text string displayed on a Web page or application. You can define the output value as part of the displayed text, and specify the text before and/or after the output text.

You can create a text output value while recording or editing your test.

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 pointer turns into a pointing hand.

Tip: Hold the left CTRL key to change the window focus or perform operations such as right-clicking or moving the pointer over an object to display a context menu. If the window containing the object you want to select is minimized, you can display it by holding the left CTRL key, right-clicking the application in the Windows task bar, and choosing **Restore** from the context menu.

- 3 In your application, click the text string for which you want to specify a text output value. The Text Output Value Properties dialog box opens.
- 4 Specify the settings for the output value. For more information, see “Defining Text Output Values” on page 431.
- 5 When you finish defining the text output value details, click **OK**. QuickTest inserts an output value step in your test.

To create a text output value when editing your test:



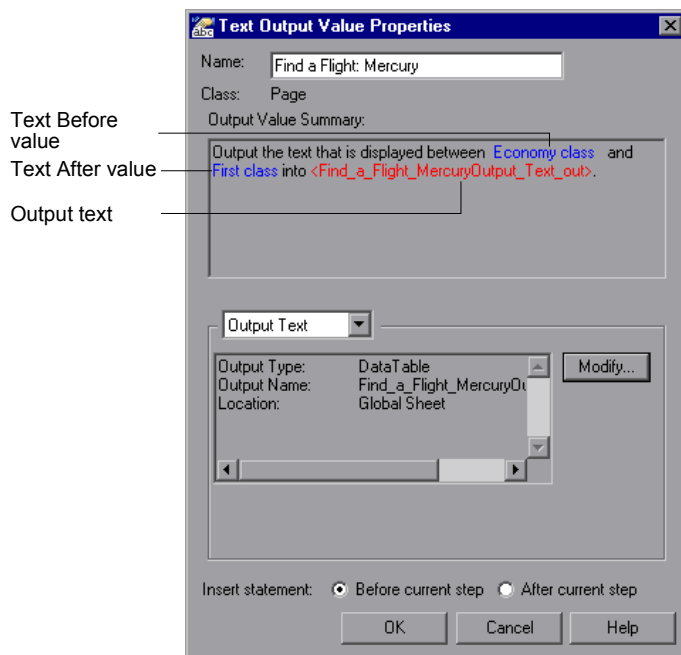
- 1 Make sure the **Active Screen** button is selected.
- 2 Click a step in your test where you want to create an output value. The Active Screen displays the screen corresponding to the highlighted step.
- 3 In the Active Screen, highlight or display the text string you want to specify as an output 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 Output Values” on page 431.
- 6 When you finish defining the output value details, click **OK**. QuickTest inserts an output value step in your test.

Defining Text 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 dialog box enables you to define the output value settings for the selected text string, and to define the options for the text displayed before and after the output value.



The top of the Text Output Value Properties dialog box displays the name of the output value and the class of the test object on which the output value check is being performed. You can modify the output value name, if required. For more information, see “Identifying the Output Value” on page 422.

The **Output Value Summary** pane at the top of the dialog box describes the text string for the output value. The text string is the string displayed between the **Text Before** value and the **Text After** value. This pane also shows the output name assigned to the text string. QuickTest automatically displays the text output in red, and the text before and after the text output in blue. For example, in the dialog box displayed above, the output value is the text displayed between **Economy class** (the **Text Before** value) and **First class** (the **Text After** value).

When you create a text output value, you can specify the captured text as an output value. You can also specify options for **Text Before** and **Text After** values. For example, you can define these values as parameters. If the specified text is displayed more than once in the selected object or area, you can specify the exact occurrence that relates to the output value. If you are editing your test, you can also specify the location for the output value step.

Identifying the Output Value

The top part of the Table Output Value Properties dialog box contains the following options:



The screenshot shows a dialog box with two fields. The 'Name' field contains the text 'Find a Flight: Mercury' and the 'Class' field contains the text 'Page'.

Name	<p>The name that QuickTest assigns to the output value. By default, the output value name is the name of the test object for which you are performing the output value step. You can specify a different name for the output value or accept the default name.</p> <p>If you rename the output value, make sure that the name is unique, does not begin or end with a space, and does not contain the following character/combination of characters:</p> <p>" := @@</p>
Class	<p>Specifies the type of object (read-only). This may be a table-type object or a list view-type object.</p>

Specifying the Captured Text as an Output Value

By default, **Output Text** is selected in the list box in the middle of the dialog box. The area below the list box displays the current output value settings for the selected text.

When you create a new output value, the default output definition is displayed for the value. For more information, see “Understanding Default Output Definitions” on page 425.

You can accept the displayed output definition, or you can click **Modify** to specify the output settings for the selected text. For more information, see “Specifying the Output Type and Settings” on page 425.

Specifying Options for the Text Before/Text After Values

When you choose **Text Before** or **Text After** from the list box, you can define the options for the text displayed before or after the output value string.

Option	Description
Use the text before / Use the text after	<p>When selected, the current Text Before / Text After value is displayed in the Constant box.</p> <p>When cleared, QuickTest retrieves the value of the first occurrence of the defined output string, regardless of the text displayed before it (if you chose Text Before) or after it (if you chose Text After).</p> <p>Note: When this check box is cleared, the options below it are not available.</p>

Option	Description
<p>Text to capture is displayed before occurrence /</p> <p>Text to capture is displayed after occurrence</p>	<p>Specifies the exact occurrence of the value specified in the Constant or Parameter box, if the value is displayed more than once in the object or area.</p> <p>If you accept the default text that QuickTest recommends, the number in this box is correct. For example, if the selected output string is displayed before the first occurrence of the string First (as shown in the dialog box above). When Text After is selected, the number 1 is displayed in the Text to capture is displayed before occurrence box.</p> <p>If you modify the recommended value, you must confirm that the occurrence number is accurate. If you choose text that is not unique in the defined object or area, change the occurrence number appropriately. For example, if you want to output the text displayed after the third occurrence of the string Mercury Tours, select Text Before and enter 3 in the Text to capture is displayed after occurrence box.</p> <p>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.</p>

Option	Description
Constant	<p>Sets the Text Before or Text After value as a constant. A constant is a value that is defined directly within the test. It remains set for the duration of the test.</p> <p>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.</p> <p>Tip: It is recommended to specify a text string that is unique within the object or area whenever possible, to ensure that the occurrence number is 1.</p>
Parameter	<p>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 348.</p>

Specifying the Location for the Output Value Step

If the **Insert statement** area is displayed at the bottom of the dialog box, you can specify where the new output value step should be inserted in your test. For more information, see “Selecting the Location for the Output Value Step” on page 429.

Outputting Table Values

You can output the values of table cells and table properties while recording or editing your test. You specify the values to output using the Table Output Value Properties dialog box.

To output table values while recording:

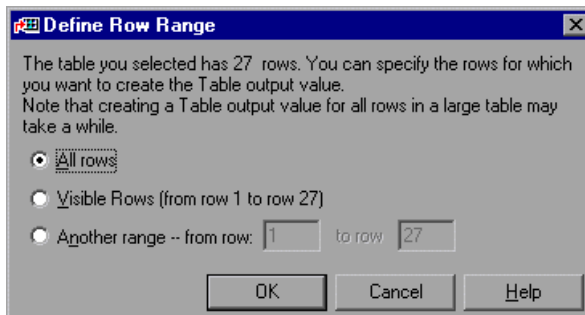


- 1 Choose **Insert > Output Value > Standard Output Value** or choose **Standard Output Value** from the **Insert Checkpoint or Output Value** button. The QuickTest window is minimized, and the pointer turns into a pointing hand.

Tip: You can hold the left CTRL key to change the pointing hand into a standard pointer, and then change the window focus or perform operations such as right-clicking or moving the pointer over an object to display a context menu. If the window containing the object you want to select is minimized, you can display it by holding the left CTRL key, right-clicking the application in the Windows task bar, and choosing **Restore** from the context menu.

- 2 Click the table from which you want to output cell values. The Object Selection - Output Value Properties dialog box opens.
- 3 Select a table item from the displayed object tree and click **OK**. If the Table Output Value Properties dialog box opens, skip to step 4.

Otherwise, for certain objects in certain environments, such as WinList View objects, the Define Row Range dialog box opens.



Select the range of rows you want to include in your output value. You can include:

- **All rows.** Includes all of the rows in the table. Note that capturing all of the data for large table or list view objects may take some time.
- **Visible Rows (from row X to row Y).** Includes only the rows visible on the screen. Note that this option may not be available for some environments or object types.
- **Another range -- from row X to row Y.** You can specify any row range between 1 and the number of rows listed in the table.

Click **OK**. The Define Row Range dialog box closes, and the Table Output Value Properties dialog box displays the rows you specified (above the grid area).

- 4 In the Table Output Value Properties dialog box, specify the settings for the output value. For information on specifying the table content to output, see “Outputting Table Content” on page 440. For information on specifying the object properties to output, see “Outputting Table Properties” on page 445.

Note: For some environments, the Table Output Value Properties dialog box contains two tabs: Table Content and Properties. For other environments, the Table Output Value Properties dialog box displays only the options available in the Table Content tab, but does not contain any tabs.

- 5 Click **OK** to close the dialog box. An output value statement is added for the selected object in the Keyword View and Expert View.

To add a table output value while editing:

- 1 Depending on whether the object from which you want to output a value is already in a step, do one of the following:
 - If you have already recorded a step on the object from which you want to output values, right-click the step and choose **Insert Output Value**. Alternatively, select the step and choose **Insert > Output Value > Standard Output Value**.

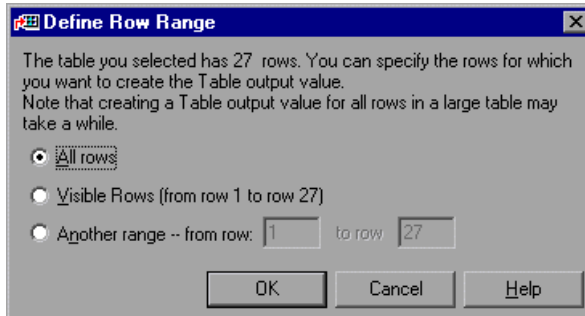


- If you have not recorded a step on the object from which you want to output values, make sure the **Active Screen** button is selected and the Active Screen is visible. Click a step in your test where you want to add an output value. The Active Screen displays the Web page or application screen corresponding to the highlighted step. Right-click the table in the Active Screen and choose **Insert Output Value**. The Object Selection - Output Value Properties dialog box opens. Select a table item from the displayed object tree and click **OK**.

Note: In some environments, you must have the table open in your application to output a value from it.

- 2 If the Table Output Value Properties dialog box opens, skip to step 3.

Otherwise, for certain objects in certain environments, the Define Row Range dialog box opens.



Select the range of rows you want to include in your output value. You can include:

- **All rows.** Includes all of the rows in the table. Note that capturing all of the data for large table or list view objects may take some time.
- **Visible Rows (from row X to row Y).** Includes only the rows visible on the screen. Note that this option may not be available for some environments or object types.
- **Another range -- from row X to row Y.** You can specify any row range between 1 and the number of rows listed in the table.

Click **OK**. The Define Row Range dialog box closes, and the Table Output Value Properties dialog box displays the rows you specified (above the grid area).

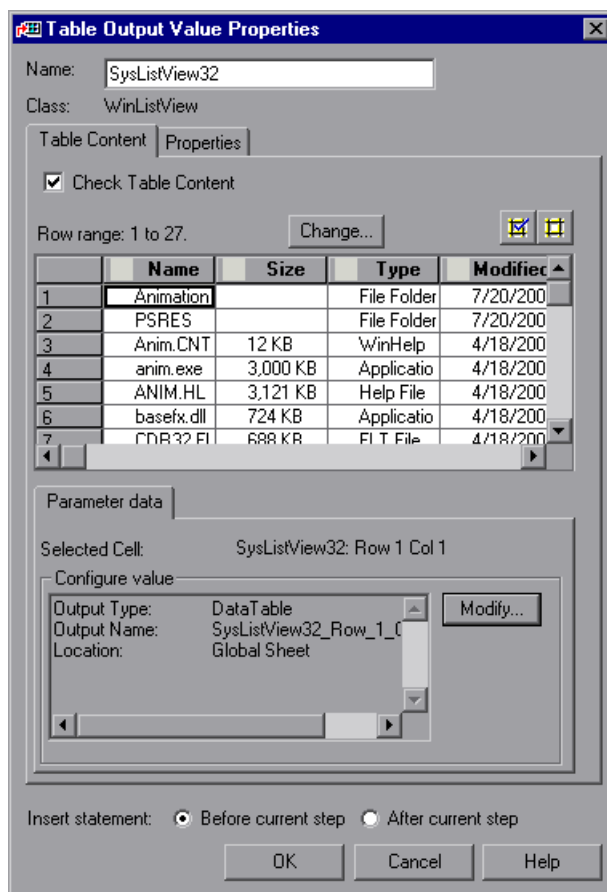
- 3** In the Table Output Value Properties dialog box, specify the settings for the output value. For information on specifying the table content to output, see “Outputting Table Content” on page 440. For information on specifying the object properties to output, see “Outputting Table Properties” on page 445.

Note: For some environments, the Table Output Value Properties dialog box contains two tabs: Table Content and Properties. For other environments, the Table Output Value Properties dialog box displays only the options available in the Table Content tab, but does not contain any tabs.

- 4** Click **OK** to close the dialog box. An output value statement is added for the selected object.

Outputting Table Content

You can specify the table cells whose content you want to output. Depending on the environment, you do this either in the Table Content tab of the Table Output Value Properties dialog box—or directly in the Table Output Value Properties dialog box, if the dialog box does not contain any tabs.



Notes:

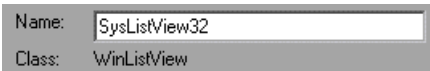
Some of the options shown in this example are available only in certain environments and only for certain objects.

For some environments, you can also specify object properties (using the Properties tab).

This section describes the general settings and options displayed in the Table Output Value Properties dialog box. Most of the options described in this section are available regardless of whether the Table Output Value Properties dialog box contains tabs.

Identifying the Output Value

The top part of the Table Output Value Properties dialog box contains the following options:



A screenshot of the 'Table Output Value Properties' dialog box. It has two fields: 'Name' with the value 'SysListView32' and 'Class' with the value 'WinListView'.

Name	<p>The name that QuickTest assigns to the output value. By default, the output value name is the name of the test object for which you are performing the output value step. You can specify a different name for the output value or accept the default name.</p> <p>If you rename the output value, make sure that the name is unique, does not begin or end with a space, and does not contain the following character/combination of characters:</p> <p>" := @@</p>
Class	<p>Specifies the type of object (read-only). This may be a table-type object or a list view-type object.</p>

Tabs (If Available)

If the Table Output Value Properties dialog box contains tabs, each tab displays a check box. You can select one or both of these check boxes to specify the type of data to output.

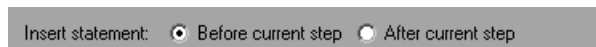


Check Table Content check box	(Table Content tab) Selecting the Check Table Content check box instructs QuickTest to output the values of the selected cells in the table object. (Selected by default.)
Check Properties check box	(Properties tab) Selecting the Check Properties check box instructs QuickTest to output the property values of the selected cells in the table object. (Cleared by default.)

Note: These check boxes are displayed only if the Table Output Value Properties dialog box contains tabs. If the Table Output Value Properties dialog box does not contain tabs, QuickTest automatically outputs the values of the selected cells as defined in the dialog box.

Statement Location

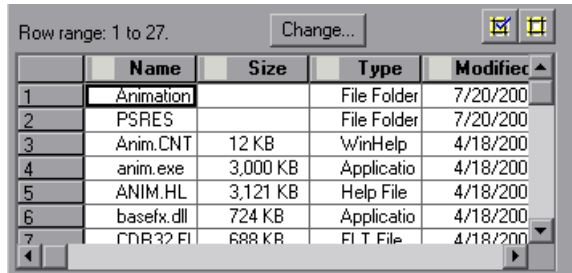
The bottom part of the Table Output Value Properties dialog box contains the following options:



Insert statement	<p>Specifies when to output the cell values in the test. Choose Before current step if you want to output the table cell values before the highlighted step is performed. Choose After current step if you want to output the table cell values after the highlighted step is performed.</p> <p>Note: The Insert statement option is available only when adding a new output value while editing an existing test. (This option is not available during recording.)</p>
-------------------------	-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

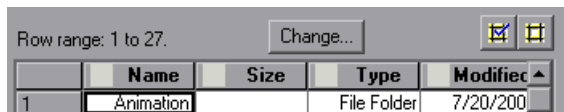
Choosing Cells for Output Values

The top part of the dialog box displays a grid representing the cells in the captured table. The column header names are captured from the table you selected for your output value step. You can output the values for one or more cells in the grid.



Tip: You can change the column widths and row heights of the grid by dragging the column and row header dividers.

Some environments and objects support selecting a row range. This enables you to specify which rows are displayed in the grid area. If row range selection is supported, the row range you specify when creating the output value is displayed above the grid:



Clicking the **Change** button enables you to modify the row range. (Depending on the environment, your application may need to be open and the relevant table displayed to modify the row range.) For more information, see “Modifying a Table Output Value” on page 447.

To choose a cell for a value to output:



Double-click the cell or select it and click the **Add Output Value** button (located above the grid, on the right). An output value icon is added to the cell.

To remove a cell from an output value:



Double-click the cell again or select it and click the **Remove Output Value** button (located above the grid, on the right). The output value icon is removed from the cell.

Specifying the Settings for the Output Value

When a value in a table cell is first selected for output, the default output definition for the value is displayed in the **Configure value** area. For more information on default output definitions, see “Understanding Default Output Definitions” on page 425.

When you select a value in a table cell, you can:

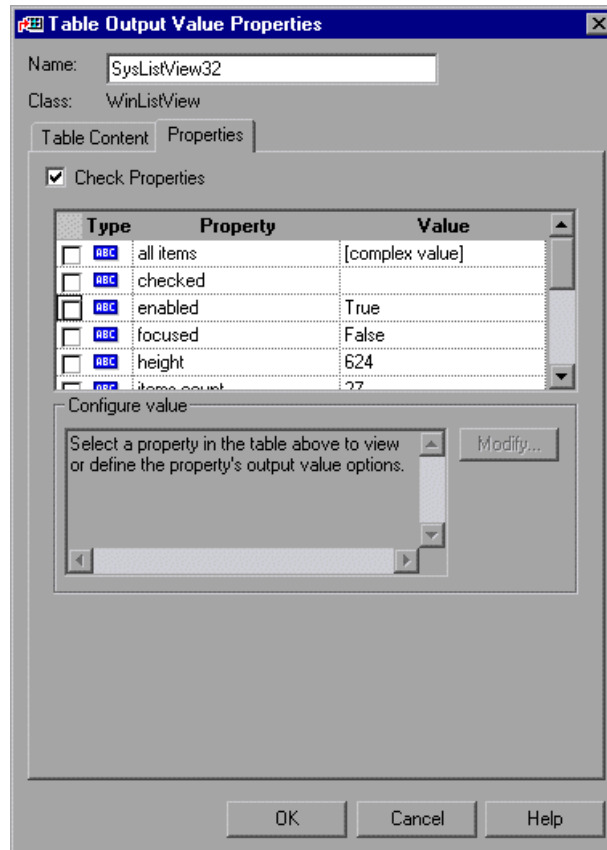
- accept the displayed output definition by selecting another cell or by clicking **OK**.
- change the output type and/or settings for the selected value by clicking the **Modify** button. The Output Options dialog box opens and displays the current output type and settings for the value. For more information, see “Specifying the Output Type and Settings” on page 425.

Specifying the Location for the Output Value Step

If the **Insert statement** area is displayed at the bottom of the dialog box, you can specify where the new output value step should be inserted in your test. For more information, see “Selecting the Location for the Output Value Step” on page 429.

Outputting Table Properties

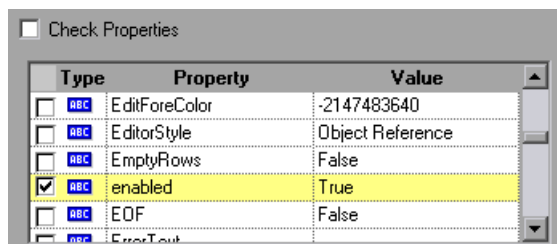
For certain environments, you can specify which object property values you want to output. By default, when you create a table output value on an object, QuickTest captures all the object's properties, but does not select any properties to output.



Note: For information on general table output value options, such as **Name** and **Class**, and on the options available in the Table Content tab, see “Outputting Table Content” on page 440.






Selecting Properties to Output

When you create a table output value, the Properties pane displays the object's default properties, including the properties, their values, and their types.



You instruct QuickTest to output specific properties by selecting the **Check Properties** check box. (This check box is cleared by default.)

The Properties pane for the object contains the following:

Check box	<p>To output a property, select the corresponding check box.</p> <p>To remove a property from the output, 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 or action 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 347.

Modifying a Table Output Value

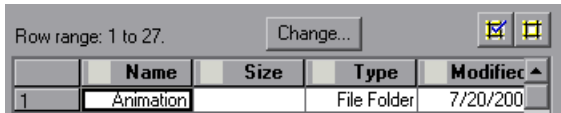
You can modify the table output value options, which specify where an output value is stored and how it can be used during the run session. You can also modify the number of rows for which QuickTest can output the values of specific table cells.

To modify the table output value options:

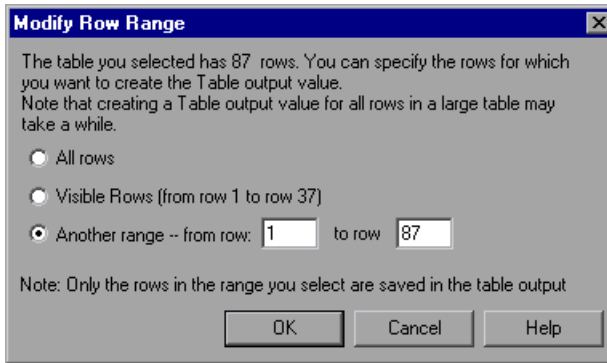
- 1** In the Keyword View or Expert View, right-click the Output CheckPoint step for the table whose output options you want to modify and select **Output Value Properties**. Alternatively, select the step containing the Output CheckPoint and choose **Edit > Step Properties > Output Value Properties**. The Table Output Value Properties dialog box opens.
- 2** Perform one of the following:
 - If the Table Output Value Properties dialog box does not contain tabs, click the **Modify** button. The Output Options dialog box opens.
 - If the Table Output Value Properties dialog box contains tabs:
 - To modify the output options for the table content, make sure the Table Content tab is displayed and click the **Modify** button.
 - To modify the output options for the object properties, select the Properties tab and click the **Modify** button.The Output Options dialog box opens.
- 3** Modify the output value, as needed. For more information, see “Specifying the Output Type and Settings” on page 425.
- 4** You can also rename the output value, if needed. For more information, see “Identifying the Output Value” on page 422.

To modify the range or number of rows in an existing table output value:

- 1 Open the application containing the table or list view object from which you want to output a value and display the object in the application.
- 2 In the Keyword View or Expert View, right-click the Output CheckPoint step for the table whose row range you want to modify and select **Output Value Properties**. Alternatively, select the step containing the Output CheckPoint and choose **Edit > Step Properties > Output Value Properties**. The Table Output Value Properties dialog box opens, displaying the currently selected row range.



- 3 In the Table Content tab, click the **Change** button at the top of the dialog box (above the grid area). The Modify Row Range dialog box opens.



- 4 Select the range of rows you want to include in your checkpoint. You can include all the rows, only the visible rows, or another range that you specify.

Note: The **Visible Rows** option may not be available for some environments or object types.

- 5** Click **OK**. The Modify Row Range dialog box closes, and the Table Checkpoint Properties dialog box displays the rows you specified in the Modify Row Range dialog box.
- If your modified row range includes new rows, you can select the cells from which you want to output values from the newly selected rows. The cells whose values you select will be outputted during the run session.
 - If your modified row range includes some or all of the rows that you specified previously, the cells whose values you selected to output will be outputted during the run session.
 - If your modified row range excludes some or all of the rows that were selected previously, the values of any previously selected cells in those rows will not be outputted during the run session.

Note: You can output values only from cells that are included in the specified row range.

Outputting Database Values

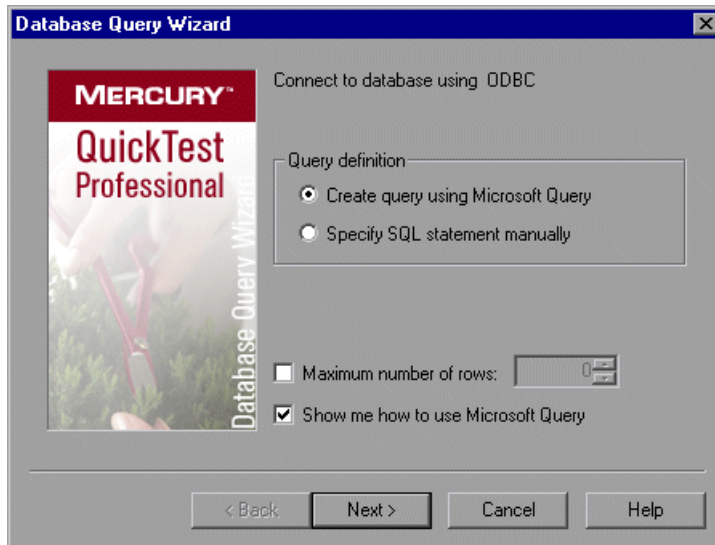
You can create database output values by defining a query to retrieve data from the database and selecting the values you want to output from the query result set. You can then specify the output settings for the selected values. During the run session, QuickTest captures the current data from the database and outputs the values according to the specified settings.

You can create database output values while recording or editing your test.

To create database output values:



- 1 Choose **Insert > Output Value > Database Output Value**. The Database Query Wizard opens.



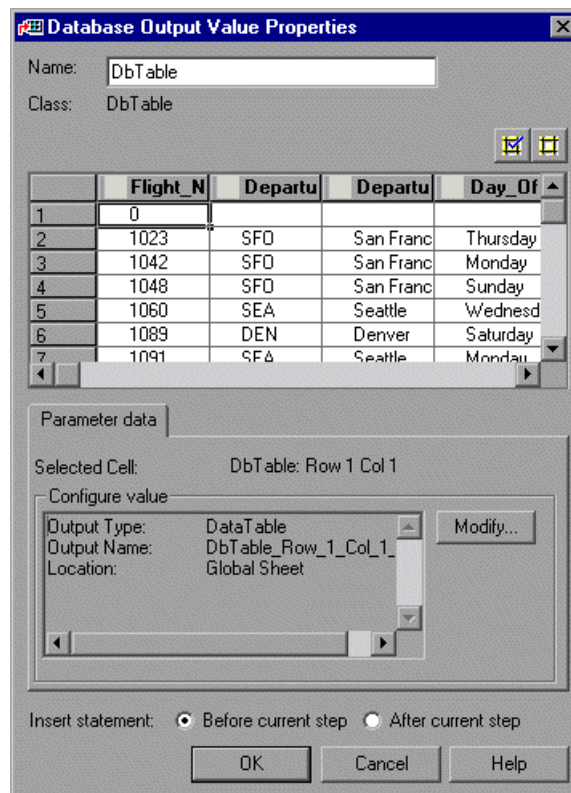
- 2 Use the wizard to define a query to retrieve the data that you want to output. Follow the instructions for creating a database checkpoint in “Creating a Check on a Database” on page 298.

After you finish defining your query, the Database Output Value Properties dialog box opens.

- 3 Specify the values to output and their settings. For more information, see “Defining Database Output Values” on page 451.
- 4 When you finish defining the output value details, click **OK**. QuickTest inserts an output value step in your test.

Defining Database Output Values

The Database Output Value Properties dialog box enables you to select the database cells for the values you want to output. You can define the output settings for each value that you select.



Identifying the Database Output Value

The **Name** box displays the name that QuickTest assigns to the output value. By default, the output value name is the name of the test object for which you are performing the output value step. You can specify a different name for the output value or accept the default name.

If you rename the output value, make sure that the name is unique, does not begin or end with a space, and does not contain the following character/combination of characters:

" := @@

The **Class** area displays the type of test object (read-only) for which you are performing the output value step.

Choosing Cells for Output Values

The top part of the dialog box displays a grid representing the cells in the captured database query results set. You can output the values for one or more cells in the grid.

Tip: You can change the width of the columns and the height of the rows in the grid by dragging the boundaries of the column and row headers.

To choose a cell for a value to output:



Double-click the cell or select it and click the **Add Output Value** button (located above the grid, on the right). An output value icon is added to the cell.

To remove a cell from an output value:



Double-click the cell again or select it and click the **Remove Output Value** button (located above the grid, on the right). The output value icon is removed from the cell.

Specifying the Settings for the Output Value

When a value in a database cell is first selected for output, the default output definition for the value is displayed in the **Configure value** area. For more information on default output definitions, see “Understanding Default Output Definitions” on page 425.

When you select a value in a database cell, you can:

- accept the displayed output definition by selecting another cell or by clicking **OK**.
- change the output type and/or settings for the selected value by clicking the **Modify** button. The Output Options dialog box opens and displays the current output type and settings for the value. For more information, see “Specifying the Output Type and Settings” on page 425.

Specifying the Location for the Output Value Step

If the **Insert statement** area is displayed at the bottom of the dialog box, you can specify where the new output value step should be inserted in your test. For more information, see “Selecting the Location for the Output Value Step” on page 429.

Outputting XML Values

You can create XML output value steps from any XML document contained in an XML Web page or frame, directly from an XML file, or from test objects that support XML. You can output element and/or attribute values in an XML output value step.

You can insert XML Web page or frame output value steps only while you are recording. You can create XML output value steps from an XML file or from a test object while recording or editing your test.

Note: XML Output Values are compatible with namespace standards and a change in namespace between nodes stored in the Output Properties dialog box XML tree and the actual values will result in a failed output value step.

For more information on XML standards, refer to <http://www.w3.org/XML/>

For more information on namespace standards, refer to <http://www.w3.org/TR/1999/REC-xml-names-19990114/>

To create XML output values from an XML Web page or frame:



- 1 While recording, choose **Insert > Output Value > XML Output Value (From Application)**, or click the **Insert Checkpoint or Output Value** button and select **XML Output Value (From Application)**. The pointer turns into a pointing hand.

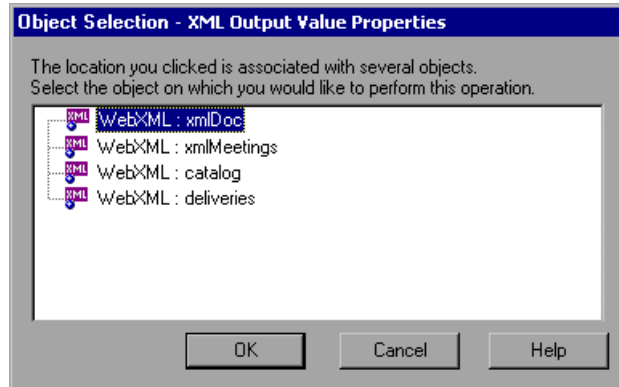
Note: The **XML Checkpoint (From Application)** option is available only when the Web Add-in is installed and loaded. For more information on loading add-ins, see Chapter 28, “Working with QuickTest Add-Ins.”

Tips:

Hold the left CTRL key to change the window focus or perform operations such as right-clicking or moving the pointer over an object to display a context menu. If the window containing the object you want to select is minimized, you can display it by holding the left CTRL key, right-clicking the application in the Windows task bar, and choosing **Restore** from the context menu.

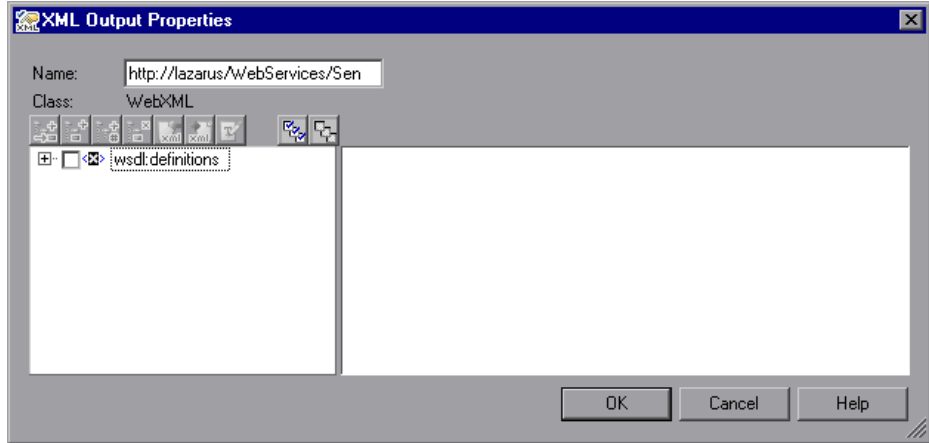
You can also insert a Web page or frame output value step using the **XML (From Resource)** option by selecting an existing WebXML test object. For more information, see creating XML output values from a test object that supports XML on page 459.

- 2 Click the XML object for which you want to specify an output value. If the location you clicked is associated with more than one object, the Object Selection - XML Output Value Properties dialog box opens.



- 3 Select the XML item you want to specify for the output value step.

- 4 Click **OK**. The XML Output Properties dialog box opens.



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

In the **Name** box, either accept the name that QuickTest assigns to the output value step or specify another name for it. By default, the output value name is the name of the test object on which the output value step is being performed.

If you rename it, make sure that the name is unique, does not begin or end with a space, and does not contain the following character/combination of characters:

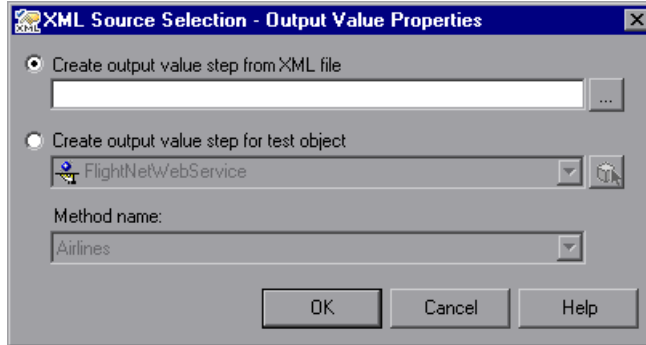
" := @@

- 5 Select the items to output. For more information, see “Understanding the XML Output Properties Dialog Box” on page 462.
- 6 When you finish defining the output value details, click **OK**. QuickTest inserts an output value step in your test.

To create an XML output value step from an XML file:



- 1 Choose **Insert > Output Value > XML Output Value (From Resource)**, or click the **Insert Checkpoint or Output Value** button and select **XML Checkpoint**. The XML Source Selection - Output Value Properties dialog box opens.



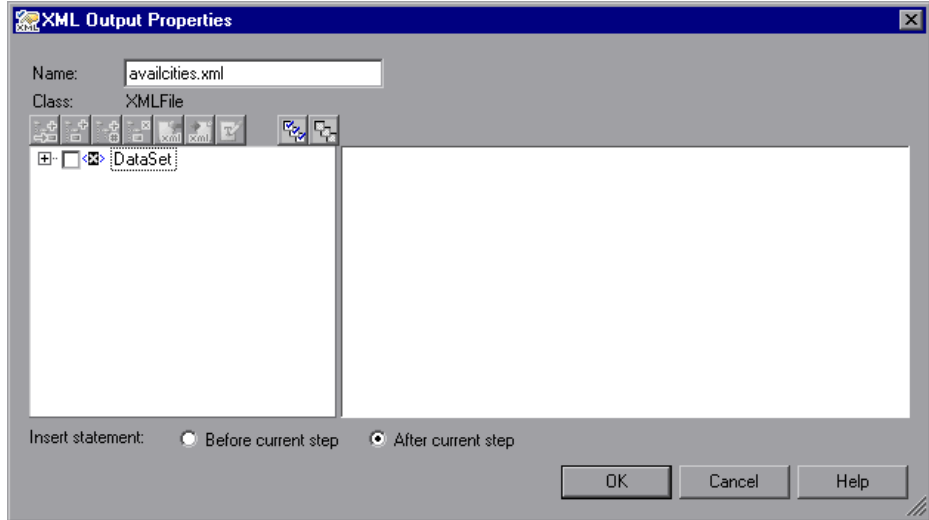
Tip: You can also insert an XML File output value step by selecting an existing XMLFile test object. For more information, see creating XML output values from a test object that supports XML on page 459.

- 2 Select **Create new checkpoint from file**. Enter the Internet address or file path of the XML file.

Alternatively, click the browse button to open the Open XML File dialog box, and then navigate to the XML file for which you want to create an output value. You can specify an XML file either from your file system or from Quality Center. Select the file and click **Open**. The file path and name are entered in the box.

Note: You can enter a relative path and QuickTest will search for the XML file in the folders listed in the Folders 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 “Setting Folder Testing Options” on page 712.

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



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

In the **Name** box, either accept the name that QuickTest assigns to the output value step or specify another name for it. By default, the output value name is the name of the test object on which the output value step is being performed.

If you rename it, make sure that the name is unique, does not begin or end with a space, and does not contain the following character/combination of characters:

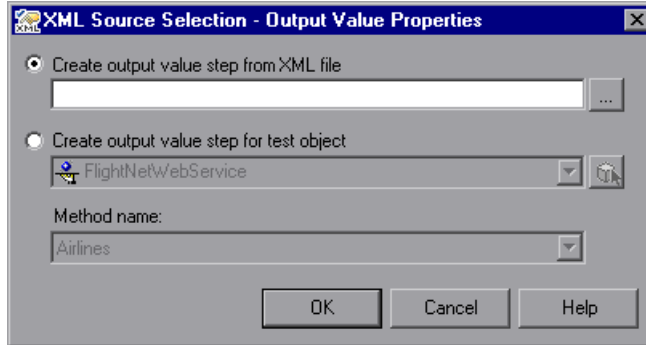
" := @@

- 4 Select the items to output. For more information, see "Understanding the XML Output Properties Dialog Box" on page 462.
- 5 When you finish defining the output value details, click **OK**. QuickTest inserts an output value step in your test.

To create an XML output value step from a test object that supports XML:



- 1 Choose **Insert > Output Value > XML Output Value (From Resource)**, or click the **Insert Checkpoint or Output Value** button and select **XML Checkpoint**. The XML Source Selection - Output Value Properties dialog box opens.



- 2 Select **Create new checkpoint for test object** and select the test object from which you want to output values.



To select an object that is not displayed in the list, click **Object from Repository**. Then select an XML test object from the object repository on which to create a new output value step. The selected object must support XML.

You can select an existing WebXML or XMLFile test object type or you can select a WebService test object.

Note: Selecting a WebXML or XMLFile test object is identical to using the **XML Output Value (From Application)** or **Create new checkpoint from file** options, but may be faster than browsing to these objects and can be inserted while recording or editing. However, to use this option, the XML source must be available when you select the test object (the Web page must be open or the file must exist in the same location as when the test object was defined).

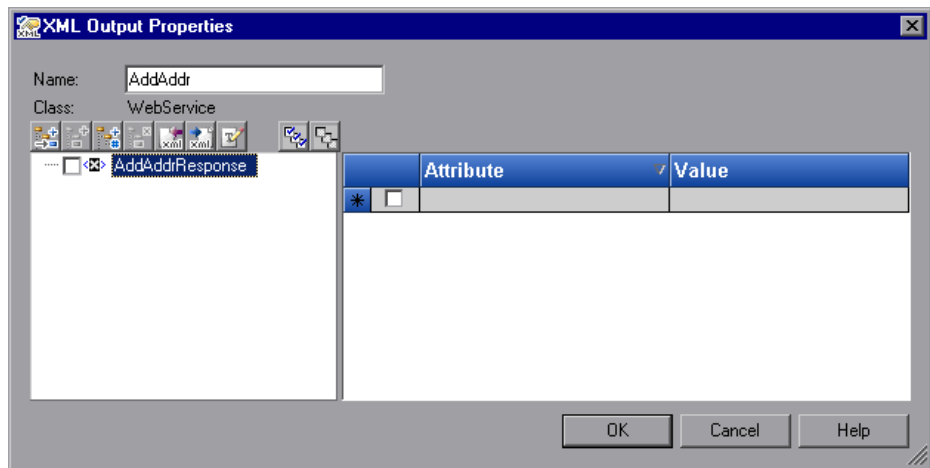
- 3 If you select a WebService test object, then the **Method name** box is enabled. Select the Web service operation whose return values you want to check.

Notes:

The **Method name** box is available only if the Web Services Add-in is installed and loaded. The **Method name** box is enabled only if you select a WebService test object.

XML output value steps on Web service operations retrieve the values returned from the last Web service operation performed on the test object. If a different Web service operation step is performed prior to the output value step, then the output value step will fail.

- 4** Click **OK**. The XML Output Properties dialog box opens.



The XML Output Properties dialog box displays the element hierarchy in an XML tree, and the attributes and values (if any) of the selected XML output.

When you create an XML output value for an operation return value, only a generic XML tree is created and shown in the XML Output Properties dialog box. Before you can select which element or attribute values you want to output, you must populate the XML tree with the actual elements, attributes, and values. For more information, see “Updating the XML Hierarchy for XML Test Object Operation Output Value Steps (For WebService Test Objects Only)” on page 466.

- 5 In the **Name** box, either accept the name that QuickTest assigns to the checkpoint or specify another name for it. By default, the output value name is the name of the test object on which the checkpoint is being performed.

If you rename it, make sure that the name is unique, does not begin or end with a space, and does not contain the following character/combination of characters:

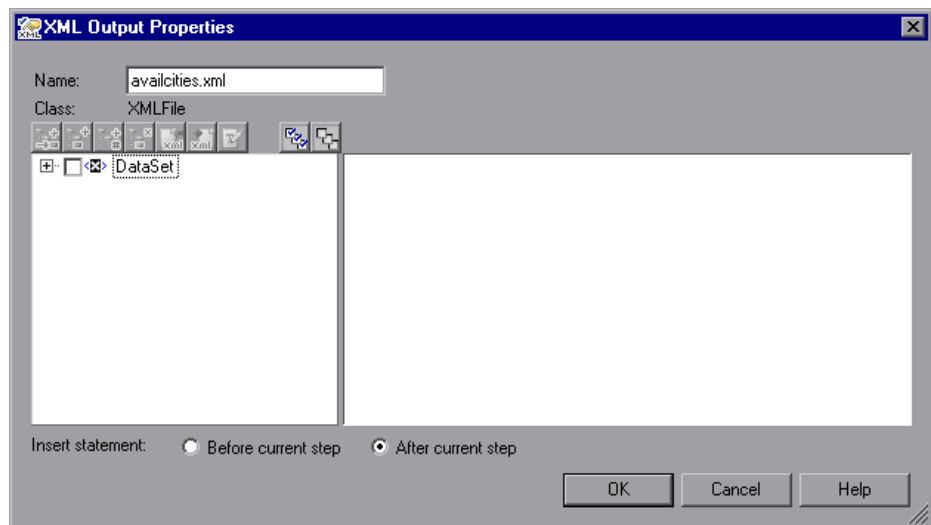
" := @@

Select the items to output. For more information, see “Understanding the XML Output Properties Dialog Box” on page 462.

- 6 When you finish defining the output value details, click **OK**. QuickTest inserts an output value step in your test.

Understanding the XML Output Properties Dialog Box

The XML Output Properties dialog box enables you to choose which element and/or attribute values to output and to define the output settings for each value that you select.








Identifying the Object





The top part of the XML Output Properties dialog box displays information on the test object for which you are creating an output value step:

Item	Description
Name	<p>The name that QuickTest assigns to the output value step. By default, the output value name is the name of the test object for which you are performing the output value step. You can specify a different name for the output value or accept the default name.</p> <p>If you rename it, make sure that the name is unique, does not begin or end with a space, and does not contain the following character/combination of characters:</p> <p>" := @@</p>
Class	<p>The test object class on which you are creating the output value step. This can be: XMLFile (for files), WebXML (for Web pages or frames) or WebService (for a Web service).</p>



Modifying the XML Tree

The following buttons are available according to the node you select in the tree:

Button Name	Icon	Description
Add Child		Adds a child node below the selected node in the tree.
Insert Sibling		Adds a sibling node at the same level as the selected node in the tree.
Add Value		Enables you to assign a constant or parameterized value to the selected element.
Delete		Deletes the selected node. Note that you cannot delete the root node of the output value step.
Import XML		Enables you to browse to and select a file structure from an existing XML file. The new file overrides the selected node's current sub-tree.

Button Name	Icon	Description
Export XML		Enables you to save the file structure of the selected node to an XML file.
Edit XML as Text		Opens the Edit XML as Text dialog box, enabling you to modify the XML text of the selected node and its subnodes in a test editor. For more information, see “Understanding the Edit XML as Text Dialog Box” on page 333.
Select All		Selects all elements and values in the XML tree.
Clear All		Clears all elements and values in the XML tree.

XML Tree


The XML tree displays the hierarchical relationship between each element and value in the XML tree, enabling you to select the element and/or attribute values that you want to output. Each element node is displayed with a  icon. Each value node is displayed with a  icon.

Note: When you create an XML output value for an operation return value, only a generic XML tree is created and shown in the XML Output Properties dialog box. Before you can select which element or attribute values you want to output, you must populate the XML tree with the actual elements, attributes, and values. For more information, see “Updating the XML Hierarchy for XML Test Object Operation Output Value Steps (For WebService Test Objects Only)” on page 466.

Select an element node in the XML tree to display or set output options for its attributes and values on the right of the XML Output Properties dialog box. Select a value node in the XML tree to display or set output options for its value on the right of the XML Output Properties dialog box.

Tip: The XML tree pane and the **Attribute** and **Value** columns in the right pane are resizable.

To set output XML options:

- 1** Select the check box for an element or value node in the XML tree to indicate that you want to output a value for that node.
- 2** Select the element or value node to display or set output options for its attributes and/or values.
- 3** If you are outputting an element attribute, select the check box of the attributes for which you want output values.
- 4** Click in the **Value** column of an attribute, or click in the cell of an element value, and then click the **Output Options** button  to display the Value Configuration Options dialog box, which enables you to select or define the parameter in which you want to store the retrieved value.
- 5** In the Value Configuration Options dialog box, select the parameter type. Additional options are available for the output parameter type that you select. For more information on the options available for each parameter type, see:
 - **Data Table.** “Using Data Table Parameters” on page 378.
 - **Environment.** “Using Environment Variable Parameters” on page 385.
 - **Random Number.** “Using Random Number Parameters” on page 396.

Insert Statement Options

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

Note: The **Insert statement** options are not available if you are adding a new output value step while recording or if you are modifying an existing output value step. They are available only if you are adding a new output value step while editing steps.

Updating the XML Hierarchy for XML Test Object Operation Output Value Steps (For WebService Test Objects Only)

This section is relevant only when working with XML output value steps on WebService test object operations (with the QuickTest Professional Web Services Add-in).

When you create an XML output value step for a test object operation (for a WebService test object), the XML tree of the operation return value data cannot be generated. Therefore, only a generic XML tree is created. To select the elements and attributes to output, you must first populate the XML tree with the actual elements, attributes, and values that the operation is expected to return.

You can use one of the three methods below to populate the XML tree:

- Updating the XML Tree Manually
- Importing an XML Tree from a File
- Updating the XML Tree Using Update Run Mode

Updating the XML Tree Manually

You can update the XML tree by adding elements, attributes, and values manually in the XML Output Properties dialog box.

To update the XML tree manually:

- 1 In the Keyword View, select the output value step whose XML tree you want to update. Click in the **Value** cell.



- 2 Click the **Output Properties** button or right-click and select **Output Value Properties**. The XML Output Properties dialog box opens.
- 3 Select a node in the XML tree and then click a toolbar button or choose an option from the right-click menu to:



- Add an element at the same level as the selected node
- Add an element below the selected node
- Add a value to the selected node
- Edit the XML text of the selected node
- Delete the selected node

For more information on the available tools in the XML Output Properties dialog box, see “Understanding the XML Output Properties Dialog Box” on page 462.

Importing an XML Tree from a File

You can import an XML tree from an existing file for a specific element in the XML tree hierarchy or for the whole tree.

To import an existing XML tree from a file:

- 1 In the Keyword View, select the checkpoint whose XML tree you want to update.



- 2 Click in the **Value** cell and then click the **Output Properties** button. The XML Checkpoint Properties dialog box opens.
- 3 If you want to import an XML hierarchy for the whole XML tree, select the root node. If you want to import an XML hierarchy for a specific element, select the element in the XML tree hierarchy.



- 4** Click the **Import XML** button. A message warns you that the imported hierarchy overwrites the selected node and its sub-tree. Click **Yes** to close the message.
- 5** In the Import XML from File dialog box, browse to the required XML file and click **Open**. The XML hierarchy is imported from the file.

Updating the XML Tree Using Update Run Mode

QuickTest cannot generate the return values of an operation when you insert an XML output value step on a Web service operation, but it can generate this information after it runs the operation. Therefore, you can run your Web service test in Update Run mode to automatically populate or update the elements, attributes and values in your XML tree.

To generate a new XML tree based on the current return values of the Web service operation, ensure that none of the node, attribute, or value check boxes are selected in the XML tree of the Output Value Properties dialog box.

Note: XML Output Value steps on Web service operations retrieve the values returned from the last Web service operation performed on the test object. If a different Web service operation step is performed prior to the output value step, then the output value step will fail.

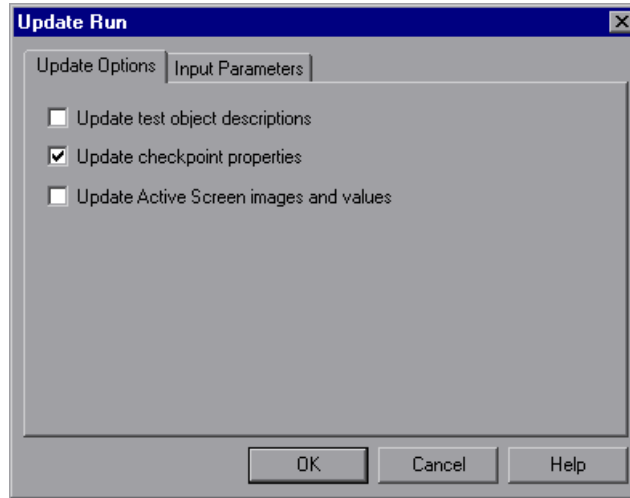
To update an XML tree using Update Run mode:

- 1** Open a test containing an XML test object output value step for a Web service operation.
- 2** Click **Update Run Mode** or choose **Automation > Update Run Mode**.





- 3 Click **Run** or choose **Automation > Run**. The Update Run dialog box opens.



- 4 Select **Update checkpoint properties** and click **OK**. QuickTest runs the test and updates the XML hierarchy and values for each blank XML checkpoint and XML output value step in the test. It updates values only for XML checkpoints or output value steps that have one or more nodes selected.



- 5 If you want to confirm that QuickTest successfully updated your output value step, expand the tree in the Test Results window and select the XML output value step. Then check that **Update done** is displayed in the pane on the right. (If the Test Results window did not open automatically at the end of the run, click the **Results** button or choose **Automation > Results**.)



Tip: When you have completed updating your XML tree using this method, it is recommended to exit the update run mode by clicking the **Update Run Mode** button again.

18

Working with Actions

You can divide your test into actions to streamline the process of testing your application or Web site. This chapter covers the basic use of actions in your test. Using advanced action-related features is described in Chapter 30, “Working with Advanced Action Features”.

This chapter describes:	On page:
About Working with Actions	472
Using Global and Action Data Sheets	475
Using the Action Toolbar in the Keyword View	477
Creating New Actions	478
Guidelines for Working with Actions	480
Setting Action Properties	482
Nesting Actions	493
Splitting Actions	495
Renaming Actions	497
Removing Actions from a Test	499
Creating an Action Template	503

About Working with Actions

Actions help divide your test into logical units, such as the main sections of a Web site, or specific activities that you perform in your application.

A test comprises calls to actions. When you create a new test, it contains a call to a single action. By creating tests that call multiple actions, you can design tests that are more modular and efficient.

An action consists of its own test script, including all of the steps recorded in that action, and any objects in its local object repository.

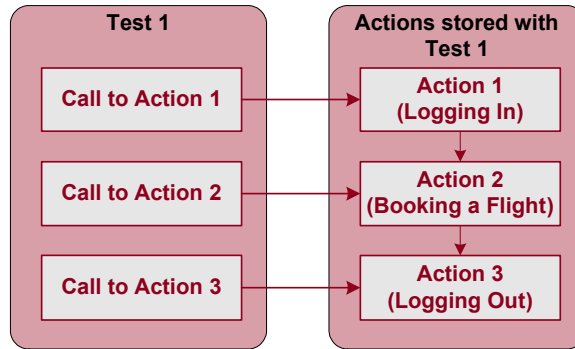
Each action is stored together with the test in which you created it. You can insert a call to an action that is stored with the test and, depending on the properties of the action, you may also be able to call an action stored with another test.

When you open a test, you can choose to view the test flow (calls to actions) or you can view and edit the individual actions stored with your test.

If you work with tests that include many steps or lines of script, it is recommended that you use actions to divide your test steps. Actions should ideally contain no more than a few dozen test steps.

For example, suppose you want to test several features of a flight reservation system. You plan several tests to test various business processes, but each one requires the same login and logout steps. You can create one action that contains the steps required for the login process, another for the logout steps, and other actions for the main steps in your test. 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 16, “Parameterizing Values,” and Chapter 17, “Outputting Values.”

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 on creating and calling new actions, see “Creating New Actions” on page 478. For more information on inserting calls to existing actions, see “Nesting Actions” on page 493.

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 493). 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 24, “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.

When there are parameters or output value steps in the current action's sheet, you can set QuickTest to run one or more iterations on that action before continuing with the current global iteration of the test. When you set your action call properties to run iterations on all rows, QuickTest inserts the next value from or to the corresponding action parameter or output value during each action iteration, while the values of the global parameters stay constant.

Note: If you create Data Table parameters or output value steps in your action and select to use the **Current action sheet (local)** option, be sure that the run settings for your action are set correctly in the Run tab of the Action Call Properties dialog box. You can set your action to run without iterations, to run iterations on all rows in the action's data sheet, or to run iterations only on the rows you specify. For more information on setting action iteration preferences, see "Inserting a Call to an Existing Action" on page 861.

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 20, “Working with Data Tables.” For more information on parameterization, see Chapter 16, “Parameterizing Values.” For more information on output values, see Chapter 17, “Outputting Values.”

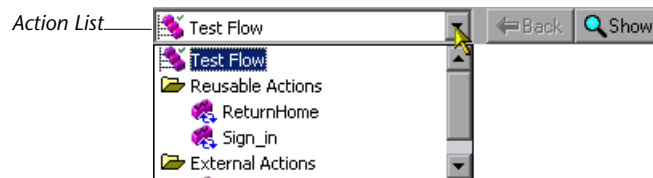
Using the Action Toolbar in the Keyword View

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.



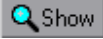
Tip: You can display or hide the Action toolbar in the Keyword View by choosing **View > Toolbars > Action**. For more information, see Chapter 2, “QuickTest at a Glance.”

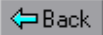
The *Action List* enables you to view either the entire test flow (the calls to the actions in the test) or you can view the steps for a selected reusable or external action. The test flow displays the overall flow of your test with all the action calls in your test. The test flow also enables you to 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 on reusable actions, see “Setting General Action Properties” on page 485.

There are several 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.



Tip: You can return to the Test Flow by clicking the **Back** button.

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 34, “Working in the Expert View and Function Library Windows.”

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 493.

You can also split an existing action into two actions. For more information on splitting actions, see “Splitting Actions” on page 495.

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 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 In the **Name** box, type a new action name or accept the default name. If you rename the action, make sure that the action name is unique (within the test), does not exceed 1023 characters, does not begin or end with a space, and does not contain the following characters:
`\ / : * ? " < > | % ' ! { }`
- 4 In the **Description** box, add a description of the action. You can also add an action description at a later time using the Action Properties dialog box.

Tip: Descriptions of actions are displayed in the Select Action dialog box. The description makes it easier for you to select the action you want to call. For more information, see “Setting General Action Properties” on page 485.

- 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 on reusable actions, see “Using Multiple Actions in a Test” on page 473. For more information on the Action Properties dialog box, see “Setting Action Properties” on page 482.

- 6 Decide where to insert the call to the action by selecting **At the end of the test** or **After the current step**. Choosing **At the end of the test** creates a call from the test flow to a top-level action. Choosing **After the current step** inserts the call to the action from within the current action (nests the action).

Note: If the currently selected step is a reusable action from another test, the new action is added automatically to the end of the test (the location options are disabled).

For more information on inserting action calls within actions, see “Nesting Actions” on page 493.

- 7 Click **OK**. A new action is stored with your test and the call to it is displayed at the bottom of the test or after the current step. You can move your action call to another location at a parallel (sibling) level within your test by dragging it to the desired location. For more information on moving actions, see “Managing Action Steps” on page 137.
- 8 If you inserted the call to the new action while editing your test, make sure your new action is selected before adding steps to it.

Guidelines for Working with Actions

Consider the following guidelines when working with actions:

- If your action runs more than one iteration, the action must end at the same point in your application as it started, so that it can run another iteration without interruption. For example, suppose you are testing a sample flight reservation site. If the action starts with a blank flight reservation form, it should conclude with a blank flight reservation form.

- A single test may include both global Data Table parameters and action (local) Data Table parameters. For example, you can create a test in which a travel agent logs into the flight reservation system, books three flights, and logs out; the next travel agent logs into the flight reservation system, books three flights, logs out, and so forth.

To parameterize the ‘book a flight’ action, you 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 > Action > Rename Action**. (Make sure you follow the naming conventions for actions. For more information, see “Creating New Actions” on page 478.)
- 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 change the required steps, if necessary, after the application is modified.
- If you decide to remove an action, consider how that might affect your test or another test that contains a call to that action. For example, will it prevent a later action in the same test from running correctly? Will it cause the test containing a call to that action to fail?

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. For an external action, you can set the Data Table definitions.

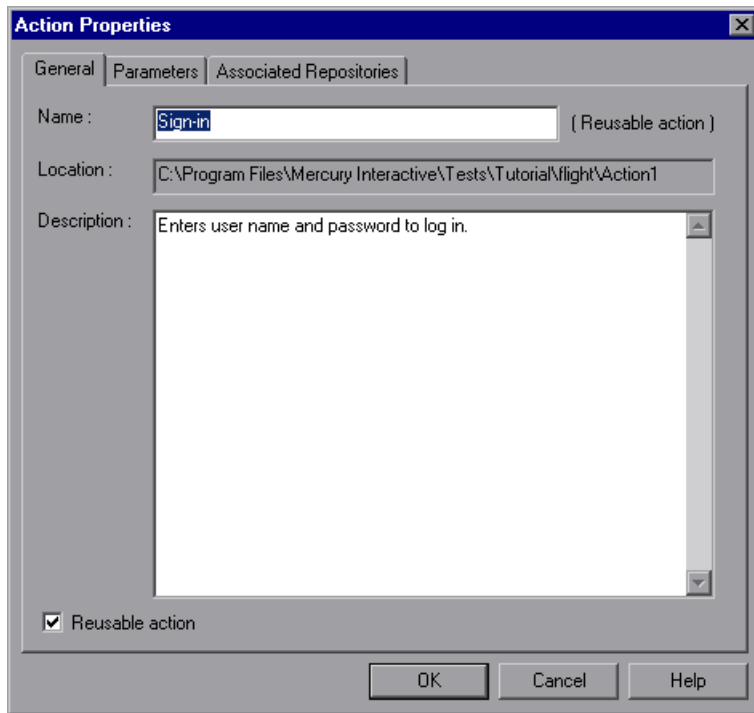
The Action Properties dialog box also enables you to define input and output parameters to be used by the action, and specify the object repositories that are associated with the action. For more information, see “Setting Action Parameters” on page 864 and “Associating Object Repositories with Actions” on page 488.

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 880.

You can open the Action Properties dialog box while recording or editing your test by:

- Choosing **Edit > Action > 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**.

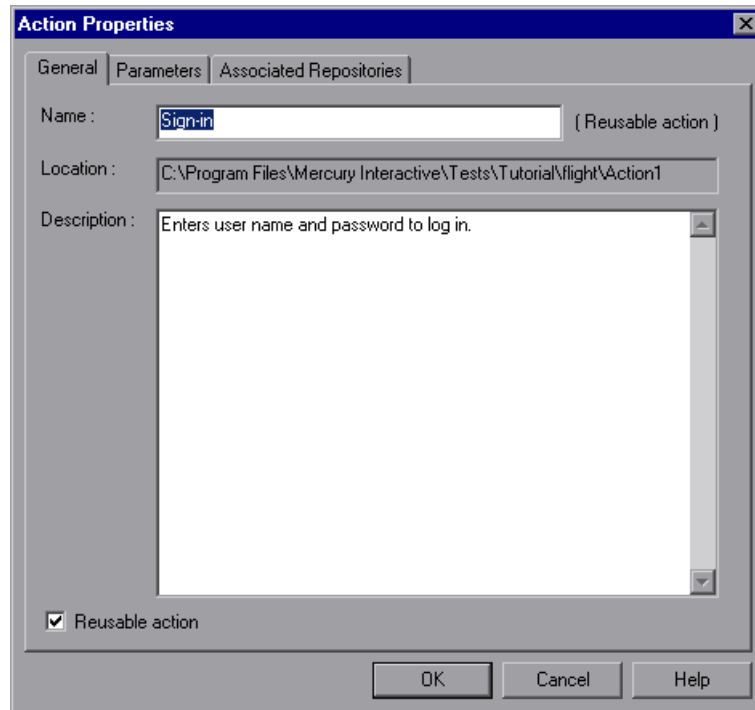
The Action Properties dialog box always contains the General tab, the Parameters tab (described in “Setting Action Parameters” on page 864), and the Associated Repositories tab as shown below:





Note: In addition to the tabs shown above, the Action Properties dialog box for an external action also contains an External Action tab, and the other tabs are read-only. For more information, see “Setting Properties for an External Action” on page 492.

Setting General Action Properties

You can use the General tab of the Action Properties dialog box (**Edit > Action > Action Properties**) 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. Note that the action name must be unique (within the test), cannot begin or end with a space, cannot exceed 1023 characters, and cannot contain the following characters: \ / : * ? " < > % ' ! { }
Location	The folder or Quality Center path where the action is stored.
Description	<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 on inserting copies and calls to actions, see “Nesting Actions” on page 493.</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 478.</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>

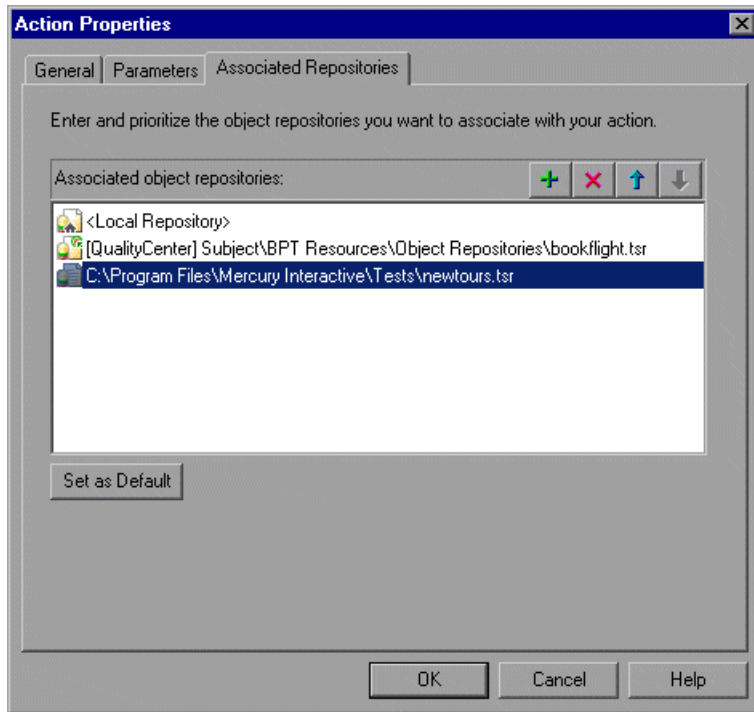
Notes:

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 on the test flow and action views, see “Using the Action Toolbar in the Keyword View” on page 477.

Associating Object Repositories with Actions

You can use the Associated Repositories tab of the Action Properties dialog box (**Edit > Action > Action Properties**) to associate one or more object repositories with the current action.




Note: If an associated shared object repository is stored in the file system on the network, its path must always be specified as a mapped drive letter (for example, C:), and not as a UNC path (for example, \\myserver\...).

Tip: You can associate shared object repositories with multiple actions simultaneously, using the Associate Repositories dialog box. For more information, see “Managing Shared Object Repository Associations” on page 214.

QuickTest searches these files to locate test object descriptions when identifying objects in your application. You can associate object repositories that are saved in your file system or in a Quality Center project.

Note: QuickTest uses associated object repositories from Quality Center project folders only when you are connected to the corresponding Quality Center project. If you are not connected to the relevant Quality Center project, all associated object repositories that are stored in your Quality Center project are listed as missing in the Missing Resources pane. (QuickTest always lists any associated object repository that cannot be found in the Missing Resources pane.)

In addition, if an object repository cannot be found, QuickTest displays a warning message when you click the Associated Repositories tab in the Action Properties dialog box. QuickTest also adds a question mark to the missing object repository icon  to the left of the missing object repository in the **Associated object repositories** list.

For more information on missing resources, see Chapter 19, “Handling Missing Resources.”





You can associate as many object repositories as needed with an action, and the same object repository can be associated with different actions as needed. You can also set the default object repositories to be associated with all new actions in all tests.

The order of the object repositories in the list determines the order in which QuickTest searches for a test object description. If there are test objects in different object repositories with the same name, object class, and parent hierarchy, QuickTest uses the first one it finds based on the priority order defined in the Associated Repositories tab. The local object repository is always listed first and cannot be moved down the priority list or deleted.

You can enter an associated object repository as a relative path. During the run session, QuickTest searches for the file in the folders listed in the Folders tab of the Options dialog box, in the order in which the folders are listed. For more information, see “Setting Folder Testing Options” on page 712.

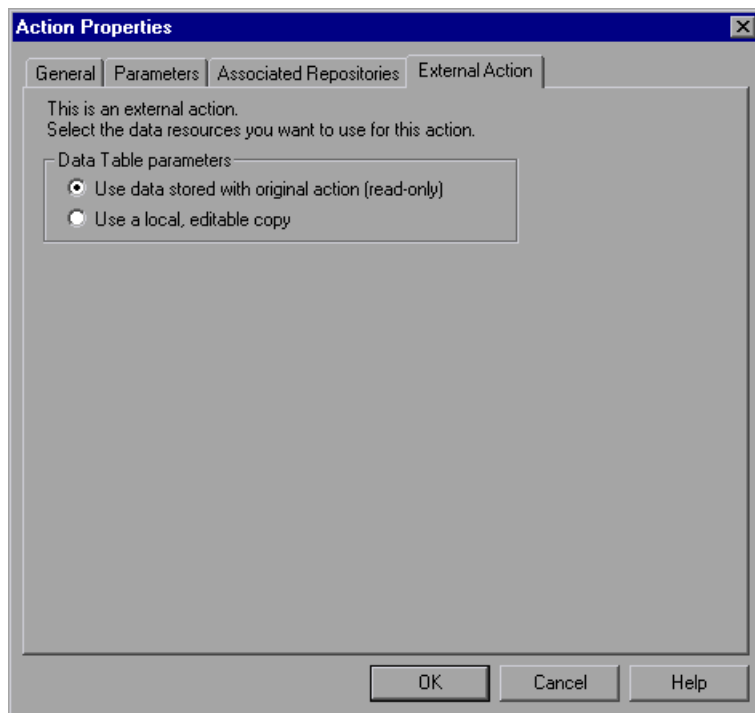
Note: If your object repositories are stored in the file system and you want other users or Mercury products to be able to run this action on other computers, you should set the file path as a relative path (click the path once to highlight it, and then click it again to enter edit mode). Any users who want to run this action should then specify the drive letter and folder in which QuickTest should search for the relative path in the Folders tab of the Options dialog box (**Tools > Options**). For more information, see “Setting Folder Testing Options” on page 712.

You can add, delete and prioritize the object repositories associated with the action using the following buttons:

Option	Description
	<p>Associates an object repository with the action. You can enter the absolute or relative path and filename of the object repository, or use the browse button to locate the required file. You can associate object repositories that are saved in your file system or in a Quality Center project.</p> <p>Note: If you are associating a shared object repository file located within the network, you must map the drive and use the mapped drive in the defined path.</p> <p>Tip: To add a Quality Center path when connected to Quality Center, click this button. QuickTest adds [Quality Center], and displays a browse button so that you can locate the Quality Center path.</p> <p>When not connected to Quality Center, hold the SHIFT key and click this button. QuickTest adds [Quality Center], and you enter the path. You can also type the entire Quality Center path manually. If you do, you must add a space after [Quality Center]. For example: [Quality Center] Subject\ObjectRepositories\flight.tsr.</p>
	<p>Removes an associated object repository from the list.</p>
	<p>Assigns a higher priority to the selected object repository.</p>
	<p>Assigns a lower priority to the selected object repository.</p>
<p>Set as Default</p>	<p>Sets the current list of object repositories as the default list to be associated with all new actions.</p> <p>Note: The Set as Default option is enabled when the setting for this action is different than the default for all actions.</p>

Setting Properties for an External Action

When you insert a call to an external action you can choose where you want QuickTest to store the Data Table data. You can specify this in the External Action tab of the Action Properties dialog box (**Edit > Action > Action Properties**).



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.

Nesting Actions

Sometimes you may want to call an action from within an action. This is called *nesting*. By nesting actions, you can:

- Maintain the modularity of your test.
- Run one or more actions based on the results of a conditional statement.

For example, suppose you have parameterized a step where a user selects one of three membership types as part of a registration process. When the user selects a membership type, the page that opens depends on the membership type selected in the previous page. You can create one action for each type of membership. Then you can use **If** statements to determine which membership type was selected in a particular iteration of the test and run the appropriate action for that selection.

In the Keyword View, your test might look something like this:

Item	Operation	Value	Documentation
Demographics Info			Call the Demographics Info action.
Membership Preferences			Call the Membership Preferences action.
Membership Preference			
Membership Preference			
MemType	Select	DataTable("memty...	Select radio button <the value of the specified Data Table c...
MemType	GetROProperty	selected	Retrieve the current value of the selected property for the "...
IF	Statement	Mem_Type = "paid"	Check whether (Mem_Type = "paid") is true. If so:
Paid_Mem			Call the Paid_Mem action.
ELSE	Statement	Mem_Type = "free"	Otherwise, Check whether (Mem_Type = "free") is true. If so:
Free_Mem			Call the Free_Mem action.
ELSE	Statement		
Preferred			Call the Preferred action.

In the Expert View, your test might look something like this:

```

Browser("Membership Preference").Page("Membership Preference").
    WebRadioGroup("MemType").Select DataTable("memtype", dtGlobalSheet)
Mem_Type=Browser("Membership Preference").
    Page("Membership Preference").WebRadioGroup("MemType").
        GetROProperty ("value")
If Mem_Type="paid" Then
    RunAction "Paid_Mem", oneliteration
Elseif Mem_Type = "free" Then
    RunAction "Free_Mem", oneliteration
Else
    RunAction "Preferred", oneliteration
End If

```

For more information on inserting conditional statements, see “Using Conditional Statements” on page 560.

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 478, 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 856.

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:

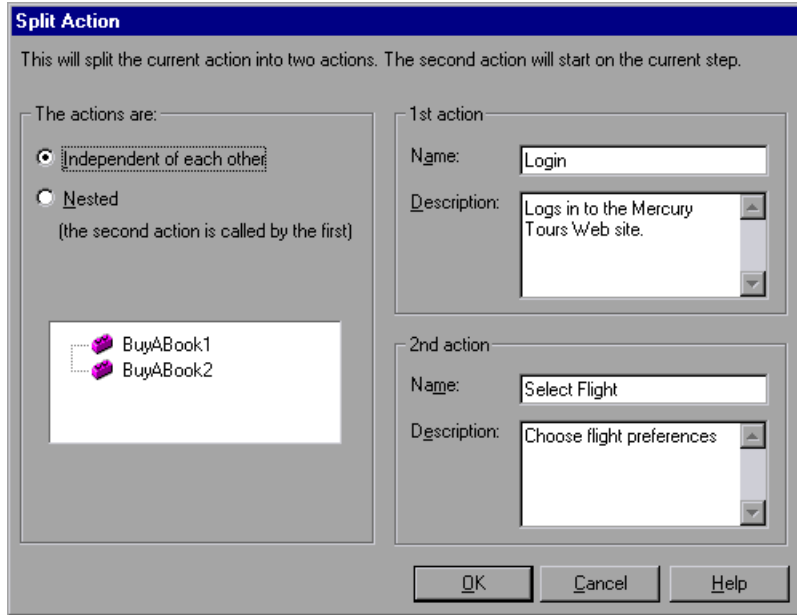
- an external action is selected
- the first step of an action is selected
- recording a test
- running a test
- you are working with a read-only test

When you split an action in your test that uses a local object repository:

- QuickTest makes a copy of the local object repository.
- The two actions have identical local object repositories containing all of the objects that were in the original local object repository.
- If you add objects to one of the split actions, the new objects are added only to the corresponding local object repository.

To split an action:

- 1 Select the step before which you want the new (second) action to begin.
- 2 Choose **Edit > Action > 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.

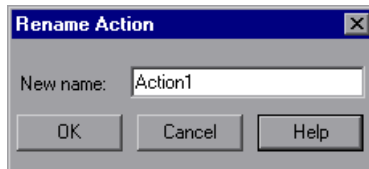
Renaming Actions

You can rename an action from the Keyword View or Expert View using the Action Properties dialog box or the Rename Action dialog box. When you rename an action, consider how it will affect your test and any tests that call this action. For example, if you rename an action that is used by another test, future run sessions may fail because the test cannot locate the specified action.

Note: 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 > Action > Rename Action**. In the Expert View display the action that you want to rename and choose **Edit > Action > Rename Action**. The Rename Action dialog box opens.

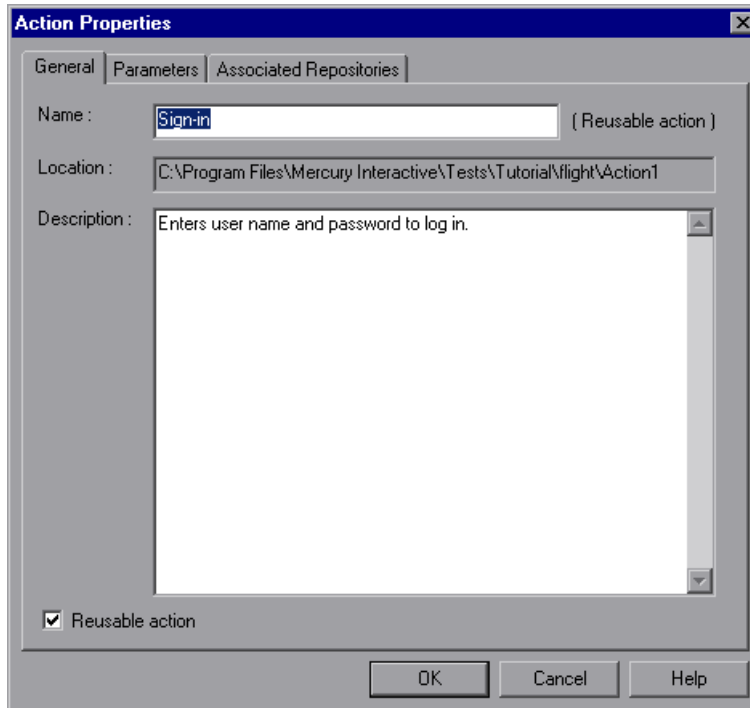


- 2 Enter a new name for the action in the **New name** box. Make sure that action name is unique within the test, does not begin or end with a space, does not exceed 1023 characters, and does not contain the following characters:
\\ : * ? " < > | % ' ! { }
- 3 Click **OK** to save the change.

Tip: You can also press SHIFT + F2 to open the Rename Action dialog box.

To rename an action in the Action Properties dialog box:

- 1 In the Keyword View or the Expert View, select an action and choose **Edit > Action > Action Properties**. Alternatively, in the Keyword View, right-click the action and choose **Action Properties**. The Action Properties dialog box opens.



- 2 Enter a new action name in the **Name** box of the General tab. Each action name within a test must be unique. Make sure that action name is unique within the test, does not begin or end with a space, does not exceed 1023 characters, and does not contain the following characters:
`\ / : * ? " < > | % ' ! { }`
- 3 Click **OK** to save the change.

Removing Actions from a Test

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. For more information, see "Removing a Non-Reusable Action" on page 500.
- 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. For more information, see "Removing a Call to a Reusable or External Action from the Test Flow" on page 500.
- 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. For more information, see "Removing a Reusable or External Action from a Test" on page 501.

Note: If you open a test containing a call to the action you removed, QuickTest informs you that the action is missing. You can either remove the call to the action or locate another action call. For more information, see "Handling Missing Resources" on page 505.

- When you remove an external action, you remove all calls and the action from the action list. The original action is not affected. For more information, see "Removing a Reusable or External Action from a Test" on page 501.

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 choose **Edit > Delete**. Alternatively, right-click the action and choose **Delete**. A delete confirmation message box opens.
- 2** Click **Yes** to confirm.

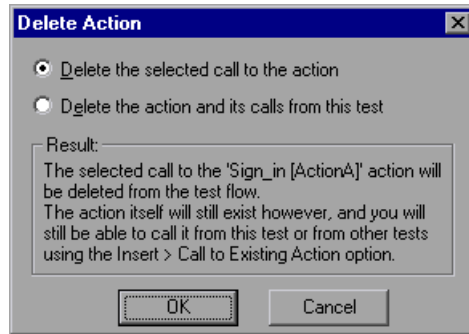
Removing a Call to a Reusable or External Action from the Test Flow

If you want to remove a reusable or external action from the test flow, but you still want the action to be available for other calls, you can remove the call to the action. 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 “Nesting Actions” on page 493.

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 and either choose **Edit > Delete** or press the **Delete** key on your keyboard. Alternatively, right-click the action and choose **Delete**. The Delete Action dialog box opens.



- 3 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 a Reusable or External Action from a Test

You can choose to completely remove a reusable or external action from a test.

When you remove a reusable action from a test, the action content is deleted entirely. Therefore, before you remove a reusable action, make sure that you no longer need the action and that no other test calls this action.

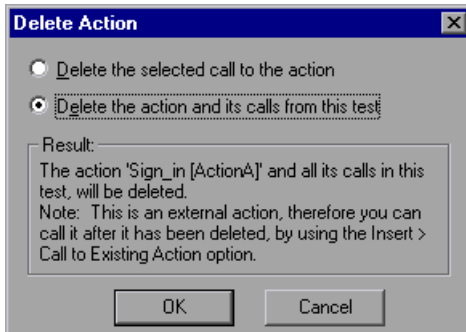
Note: Deleting a reusable action that is called by other tests will cause those tests to fail.

When you remove an external action from a 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. The original action in the source test is not affected.

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 “Nesting Actions” on page 493.

To completely remove a reusable or external action from a test:

- 1 Select the action you want to remove from the Action List.
- 2 Highlight the action you want to remove and either choose **Edit > Delete** or press the **Delete** key on your keyboard. Alternatively, right-click the action and choose **Delete**. The Delete Action dialog box opens.



- 3 Choose **Delete the action and its calls from this test** and click **OK**.

If you chose to remove an external action, the action and all calls to the action are removed from the test. The reusable action in the original test is not affected.

If you chose to remove a reusable action, the action is permanently deleted and can no longer be inserted in, or accessed by, any test. Any test calling this action will fail.

Creating an Action Template

If you want to include one or more statements in every new action in your test, you can create an action template. For example, if you always enter your name as the author of an action, you can add this comment line to your action template. An action template applies only to actions created on your computer.

To create an action template:

- 1** Create a text file containing the comments, function calls, and other statements that you want to include in your action template. The text file must be in the structure and format used in the Expert View.
- 2** Save the text file as **ActionTemplate.mst** in your **<QuickTest Installation Folder>\dat** folder. All new actions you create contain the script lines from the action template.

Note: Only the file name **ActionTemplate.mst** is recognized as an action template.

19

Handling Missing Resources

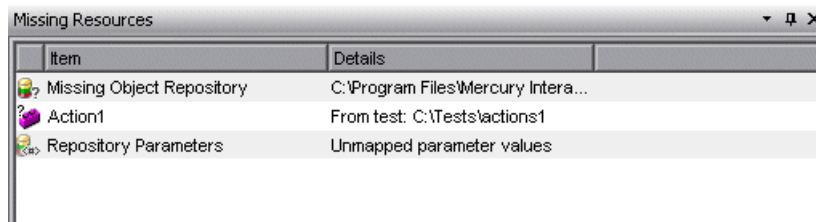
If a test has resources that cannot be found, such as missing shared object repositories or calls to missing actions, or if it uses a repository parameter that does not have a defined value, QuickTest indicates this in the Missing Resources pane. If one of the resources listed in this pane is unavailable during a run session, the test may fail. You can map a missing resource, or you can remove it from the test, as required.

This chapter describes:	On page:
About Handling Missing Resources	506
Handling Calls to Missing Actions	508
Handling Missing Shared Object Repositories	514
Handling Unmapped Shared Object Repository Parameter Values	515

About Handling Missing Resources

Each time you open a test, QuickTest verifies that the resources specified for the test are available. Specifically, QuickTest verifies that all associated shared object repositories can be found, that all defined repository parameters contain values, and that all action calls are accessible.

If one or more resources cannot be found, QuickTest opens the Missing Resources pane, if the pane is not already open. The Missing Resources pane provides a list of all resources that are currently unavailable, enabling you to remap or remove them from your test. After you successfully handle a missing resource, QuickTest removes it from the pane.



The Missing Resources pane may list any of the following types of missing resources:




<Missing Action Name>. If a test contains an action or a call to an action that cannot be found, QuickTest specifies the path it uses to search for the test containing the missing action. For more information, see “Handling Calls to Missing Actions” on page 508.



Missing Object Repository. If a test is associated with a shared object repository that cannot be found, QuickTest specifies the path it uses to search for the missing object repository. For more information, see “Handling Missing Shared Object Repositories” on page 514.



Repository Parameters. If a test has at least one test object with a property value that is parameterized using a repository parameter that does not have a default value, QuickTest adds this generic item to the Missing Resources pane. For more information, see “Handling Unmapped Shared Object Repository Parameter Values” on page 515.

Note: For missing actions, QuickTest also adds a question mark to the action's icon in the Keyword View  and displays **(Missing)** to the right of the missing action, for example:

 Action1 (MyAction) (Missing)

Filtering the Missing Resources Pane



You can choose to display all missing resources in the Missing Resources pane, or only one type of missing resource.


To filter the list of displayed missing resources:

Right-click in the Missing Resources pane and choose one of the following:

- **All.** Displays a list of all missing resources in your test.
- **Missing Action.** Displays a row for each missing action, specifying the path QuickTest uses to search for each test that contains a missing action.
- **Missing Object Repository.** Displays a row for each shared object repository that cannot be found, specifying the path QuickTest uses to search for the shared object repository.
- **Object Repository Parameter.** Displays a generic row indicating that at least one test object in the repository has at least one parameterized property value that uses a repository parameter that does not have a default value.

The Missing Resources pane is filtered according to the selected resource type and an indication of the applied filter is shown at the bottom of the pane:

  = 'Missing Object Repository'

Tip: You can cancel the filter and show all missing resources again by clicking the  icon on the left of the filter indication.

Handling Calls to Missing Actions

If your test contains a call to one or more actions that cannot be found, QuickTest lists these actions in the Missing Resources pane.

In addition, if the Test Flow does not contain a call to a particular action that is contained in the test, but the action cannot be found, QuickTest lists the action in the Missing Resources pane. For example, suppose that when you created a test, you inserted a call to a new, reusable action. Later, you deleted the call to that action by choosing the **Delete the selected call to the action** in the Delete Action dialog box (described on page 499). The action is still referenced by the test even though you deleted the call to it, and QuickTest will list it in the Missing Resources pane if it cannot be found.

You can double-click any action in the list and either map it to the required action (in the relevant test), or remove the action or the call to the action from the current test, as required.

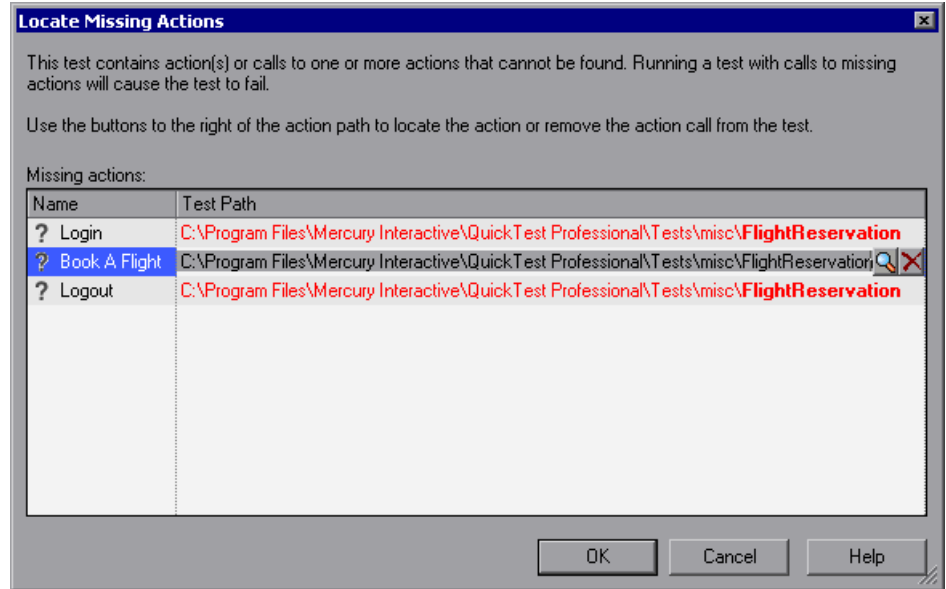
Note: If a test is opened in read-only format, you cannot view or map its missing actions.

Mapping Calls to Missing Actions

You can map both a missing action and a call to a missing action. If your test contains calls to more than one missing action, when you map a call to a missing action in another test, QuickTest may identify additional calls to missing actions that can also be mapped to the same test. This can occur, for example, if the source test (which contains the actions to which the calls were originally mapped) was renamed or was moved to another folder. You can instruct QuickTest to map calls to these actions simultaneously, or you can handle each call to a missing action individually.

To map a missing action or a call to a missing action:

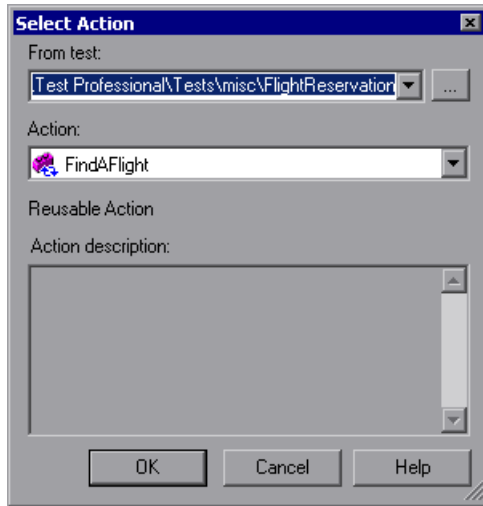
- 1 In the Missing Resources pane, double-click the action you want to map. The Locate Missing Actions dialog box opens.



- 2 Click the path for the action or action call you want to map. QuickTest displays two buttons to the right of the path—**Locate another action** and **Remove this action from current test** (also known as **Locate** and **Remove**).



- 3 Click the **Locate** button or press CTRL+INSERT to browse to the action to which you want to map the call or the action. The Select Action dialog box opens.



When the Select Action dialog box opens, the **Test** box displays either the name of the test containing the missing action (if QuickTest can identify the source test), or **<Current Test>**.

Note: If the missing action is a nested action that is called from another test, you cannot use the **Locate** button to browse to that action. Instead, you must resolve the missing action from within the external test. For example, if ActionAA (in TestA) calls ActionBB (from TestB), and ActionBB calls ActionCC (from TestC), if you open TestA and the call to ActionCC is missing, then you can only resolve the missing action by opening TestB and mapping the call to ActionCC. (You cannot resolve it from within TestA.)

- 4 Use the **Test** or **From test** browse button to find the test that contains the action you want to call, or map. The **Action** box displays all reusable actions in the test you select.

Notes:

When you select a test, the **Test** box is renamed to **From test**. If the test you select contains reusable actions, these are listed in the **Action** box.

You can enter a Quality Center folder or a relative path in the **Test/From test** box. If you enter a relative path, QuickTest searches for the test in the folders listed in the Folders tab of the Options dialog box. For more information, see “Setting Folder Testing Options” on page 712.

- 5 In the **Action** list, select the action you want to call. When you select an action, its type (Reusable Action) and description, if one exists, are displayed. This helps you identify the action you want to call. For more information on action descriptions, see “Setting General Action Properties” on page 485.
- 6 Click **OK**.
 - If only one missing action is listed in the Locate Missing Actions dialog box, QuickTest closes the Select Action dialog box, updates the test path for the selected action, and removes the missing action indication from the Missing Resources pane.
 - If more than one missing action is listed in the Locate Missing Actions dialog box, QuickTest closes the Select Action dialog box and updates the test path for the selected action in the Locate Missing Actions dialog box, replacing ? (to the left of the action name) with ✓, indicating that the test path can now be successfully mapped.

You can repeat the previous steps to map the remaining missing actions and action calls, or you can remove them, as described in “Removing Missing Actions and Calls to Missing Actions” on page 512.

After you finish mapping and/or removing calls to all missing actions, click **OK**. QuickTest updates the test with your changes and clears all handled missing resources from the Missing Resources pane.

Notes:

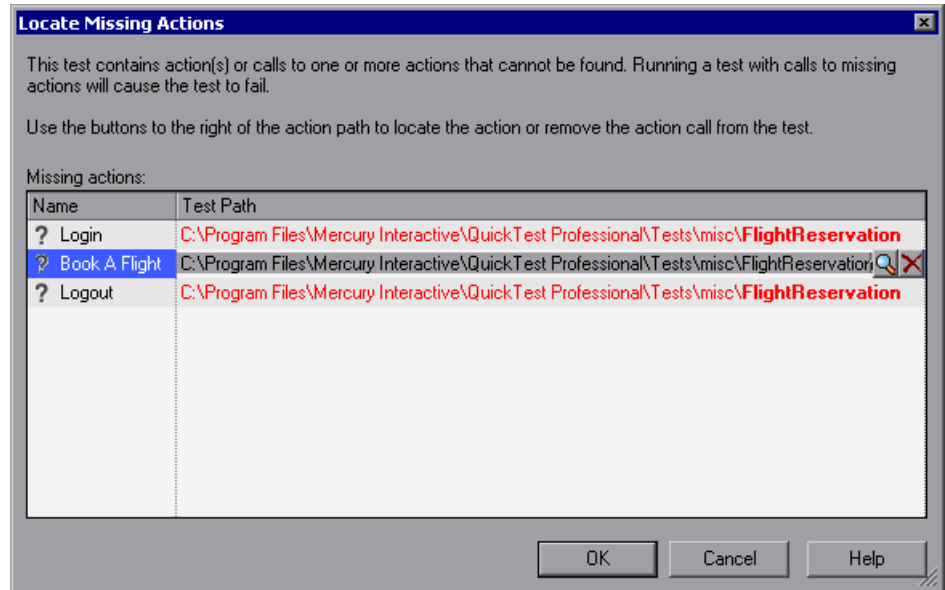
- If your test contains additional calls to missing actions that can be mapped to the same test, QuickTest opens a message box asking you if you want to map the calls to these actions, as well. Click **Yes** to map all relevant action calls, or click **No** to map only the action call you specified.
 - If one or more actions cannot be located, an error message opens listing the relevant actions.
-

Removing Missing Actions and Calls to Missing Actions

You can remove a missing action or a call to a missing action, if it is not needed.

To remove an action or a call to a missing action:

- 1 In the Missing Resources pane, double-click the action you want to remove. The Locate Missing Actions dialog box opens.



- 2 Click the path for the relevant action and click the **Remove** button or press CTRL+DELETE. Click **Yes** to confirm the removal operation. QuickTest replaces ? (to the left of the action name) with X, indicating that the action will be removed after you click **OK** to close the Locate Missing Actions dialog box.

Note: If your test contains additional calls to missing actions in the same test, QuickTest opens a message box asking whether you want to remove all calls to the actions with the same path. Click **Yes** to remove all calls to missing actions in the same path, or click **No** to remove only the action call you specified.

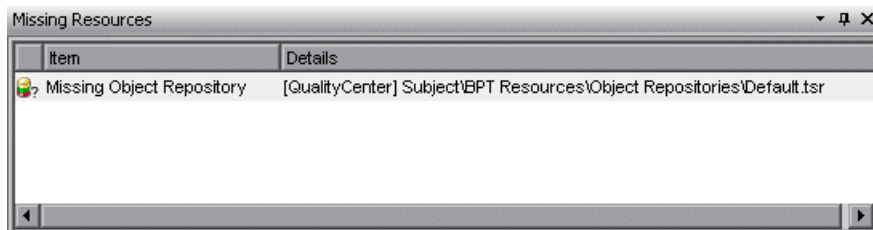
- 3 If needed, repeat the previous steps to remove additional missing actions or action calls.

- 4 After you finish removing and/or mapping all calls to missing actions (as described above), click **OK**. QuickTest updates the test with your changes and clears all handled missing resources from the Missing Resources pane.

Note: If an action cannot be located or removed, an error message opens listing the relevant actions.

Handling Missing Shared Object Repositories

When you associate a shared object repository with an action, QuickTest verifies that the repository you specified is accessible. In addition, QuickTest checks that all associated shared object repositories are accessible each time you open a test. If a shared object repository cannot be found, QuickTest displays its name and path in the Missing Resources pane when you open your test.



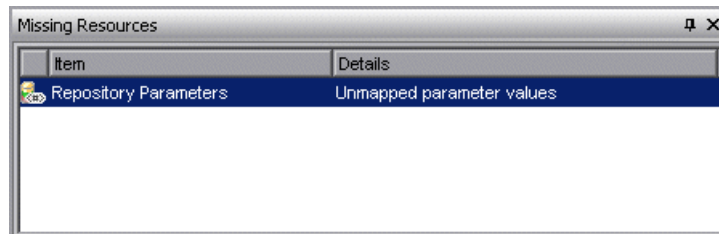
For example, if you modify the name of the shared object repository or the folder in which it is stored, you will need to map the shared object repository to the test.

If you double-click the line displaying the missing object repository, the Associate Repositories dialog box opens, enabling you to associate one or more shared object repositories with one or more actions in your test. You can also remove object repository associations from selected actions, or from all actions in your test. For more information, see “Managing Shared Object Repository Associations” on page 214.

Handling Unmapped Shared Object Repository Parameter Values

Every repository parameter used in your test must have a specified value. This can be either a default value that was specified when the parameter was created, or it can be a value that you specify in your test. (For more information on repository parameters, see “Working with Repository Parameters” on page 1133.)

When you open a test that uses an object repository that contains an object property whose value is parameterized using a repository parameter that does not have a value, QuickTest indicates this by displaying **Repository Parameters** in the Missing Resources pane.



For example, suppose your application contains an edit box whose name property changes depending on a selection made in a previous screen. If you parameterized the value of the name property in the object repository using a repository parameter, but a default value was not defined for the repository parameter, you need to define a value for it. You can map it to a DataTable, an environment variable, a random number, or a test or action parameter. You can also define a constant value for it, and so forth.

If you double-click the line displaying **Repository Parameters**, the Map Shared Object Repository Parameters dialog box opens, enabling you to specify values for any unmapped object repository parameter. You can filter the dialog box to display only unmapped parameters or all of the parameters in your test or the specific action (with mapped and unmapped values). For more information, see “Mapping Repository Parameter Values” on page 185.

20

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:	On page:
About Working with Data Tables	518
Working with Global and Action Sheets	519
Saving the Data Table	521
Editing the Data Table	522
Using Data Table Files with Quality Center	530
Importing Data from a Database	531
Using Formulas in the Data Table	534
Using Data Table Scripting Methods	538

About Working with Data Tables

The data your test uses is stored in the **design-time** Data Table, which is displayed in the Data Table pane 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 perform mathematical formulas within the cells. You can use the **DataTable**, **DTSheet** and **DTPParameter** utility objects to manipulate the data in any cell in the Data Table. For more information on these objects, refer to the **Utility** section of the *Mercury QuickTest Professional Object Model Reference*.

Note: The use of complex and/or nested formulas in the Data Table is not supported.

You can insert Data Table parameters and output values into your test. Using Data Table parameters and/or output values in a test enables you to create a **data-driven** test or action that runs several times using the data you supply. In each repetition, or **iteration**, QuickTest uses a different value from the Data Table.

During the run session, QuickTest creates a **run-time** Data Table—a live version of the Data Table associated with your test. During the run session, QuickTest displays the run-time data in the Data Table pane so that you can see any changes to the Data Table as they occur.

When the run session ends, the run-time Data Table closes, and the Data Table pane again displays the stored design-time Data Table. Data entered in the run-time Data Table during the run session is not saved with the test. The final data from the run-time Data Table is displayed in the **Run-Time Data Table** in the Test Results window. For more information on the run-time Data Table, see “Viewing the Run-Time Data Table” on page 678.

Tip: If it is important for you to save the resulting data from the run-time Data Table, you can insert a **DataTable.Export** statement to the end of your test to export the run-time Data Table to a file. You can then import the data to the design-time Data Table using the Data Table **File > Import From File** menu. Alternatively you can add a **DataTable.Import** statement to the beginning of your test to import the run-time Data Table that was exported at the end of the previous run session. For more information on these methods, refer to the *Mercury QuickTest Professional Object Model Reference*.

Working with Global and Action Sheets

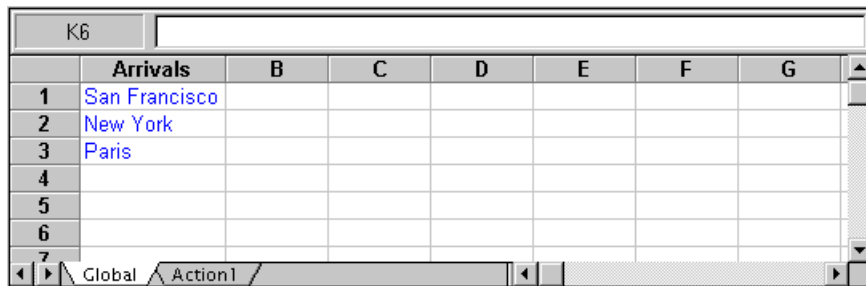
When working with tests, the Data Table has two types of data sheets—**Global** and **Action**. You can access the different sheets by clicking the appropriate tabs below the Data Table.

- You store data in the Global tab when you want it to be available to all actions in your test and you want the data to control the number of test iterations.
- You store data in the action's tab when you want to use the data in Data Table parameters for that action only and you want the data to control the number of action iterations.

For example, suppose you are creating a test on the sample Mercury Tours Web site. You might create one action for logging in, another for booking flights, and a third for logging out. You may want to create a test in which the user logs onto the site once, and then books flights for five passengers. The data about the passengers is relevant only to the second action, so it should be stored in the action tab corresponding to that action.

Global Sheet

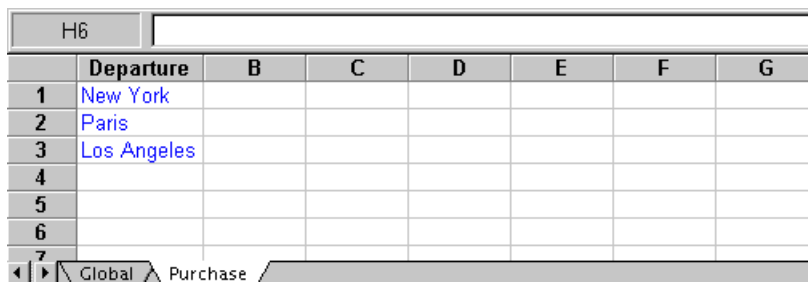
The Global sheet contains the data that replaces parameters in each iteration of the test. If you create a Global parameter called Arrivals, the Global sheet might look like this:



	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 on creating global and action parameters, see Chapter 16, “Parameterizing Values.”

Saving the Data Table

The Data Table contains the values that QuickTest substitutes for Data Table parameters when you run a test, as well as any other values or formulas you enter. Whenever you save your test, QuickTest automatically saves its Data Table as an **.xls** file.

When working with tests, the Data Table is saved with your test by default. You can save the Data Table in another location and instruct the test to use this Data Table when running a test. You specify a name and location for the Data Table in the Resources tab of the Test Settings dialog box.

For more information on the Test Settings dialog box, see Chapter 26, “Setting Options for Individual Tests.”

Saving the Data Table in a specified location can be useful in the following circumstances:

- You want to run the same test with different sets of input values. For example, you can test the localization capabilities of your application by running your test with a different Data Table file for each language you want to test. You can also vary the user interface strings that you check in each language by using a different environment parameter file each time you run the test. For more information, see Chapter 16, “Parameterizing Values.”
- You need the same input information for different tests. For example, you can test a Web version and a standard Windows version of the same application using different tests, but the same Data Table file.

Note: If you select an external file as your Data Table, you must make sure that the column names in the external Data Table match the parameter names in the test and that the sheets in the external Data Table match the action names in the test.

Editing the Data Table

You can edit information in the Data Table by typing directly into the table cells. You use the Data Table in the same way as an Microsoft Excel spreadsheet, including inserting formulas into cells. You can also import data saved in Microsoft Excel, tabbed text file (.txt), or ASCII format.

For information on supported versions of Microsoft Excel, 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							

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).

Guidelines for Working with the Data Table

- When you add data to the Data Table, you must enter the data in rows from top to bottom and left to right—you cannot leave a gap of an entire row or column. For example, if there is data in row 1, you cannot enter data in a cell in row 3 until you have entered data in row 2. Similarly, if there is data in column A, you cannot enter data in column C until you have entered data in column B.

- The value returned from the Data Table is always converted to a string. If you want the value to be converted to something other than a string, you can use VBScript conversion functions, such as **CInt**, **CLng**, **Cdbl**, and so forth. For example:

```
Window("Flight Reservation").WinComboBox("Fly From:").Select  
CInt(DataTable("ItemNumber", dtGlobalSheet))
```

- When you add content to a Data Table cell, QuickTest changes the color of the row's bottom grid line from gray to black. When you run your test using the **Run on all rows** option (defined in **File > Settings > Run** tab, or **Edit > Action > Action Call Properties > Run** tab), QuickTest runs one iteration for each row whose bottom grid line is black.

If you want to prevent QuickTest from running an iteration on a row when the **Run on all rows** option is selected, you need to delete the entire row from the Data Table using the **Edit > Delete** option (CTRL+K). (This restores the bottom grid line from black to gray.) Otherwise, if you use the **Clear** option from the table's Edit menu (or CTRL+X), or select a cell and press **Delete** on the keypad, the data is deleted from the cells, but the row is not deleted and the black line remains. This means that QuickTest will run an iteration for this row even though there is no data in it.

Data Table Specifications

The main limitations for working with the Data Table are listed below:

- **Maximum worksheet size.** 65,536 rows by 256 columns
- **Column width.** 0 to 255 characters
- **Text length.** 16,383 characters
- **Formula length.** 1024 characters
- **Number precision.** 15 digits
- **Largest positive number.** 9.999999999999999E307
- **Largest negative number.** -9.999999999999999E307
- **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)
- The use of colors and formatting in the Data Table is not supported.
- Complex and/or nested formulas are not supported in the Data Table.
- Combo box and list cells, conditional formatting, and other special cell formats are not supported in the Data Table.

Changing a Column Name

You can change the name of a column for a parameter by double-clicking the column heading cell. In the Change Parameter Name dialog box, you can type a new parameter name. This parameter name must be unique in the test. It can contain letters, numbers, commas, and underscores. The first character of the parameter name must be a letter or an underscore.

Note: If you change the name in the table, you must also change the name defined for the corresponding parameter in the test.

Using the Data Table Menu Commands

You can use the Data Table menu commands described below to edit the data in the Data Table. To open the Data Table menu, right-click a cell, a row heading or a column heading.

The following menus are available:

- File
- 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>Note: The table file you import replaces all data in all sheets of the table. The first row in each Microsoft Excel sheet also replaces the column headers in the corresponding Data Table sheet. It is therefore essential that the first row of your Microsoft Excel sheet exactly matches the parameter names in your test, and that the file contains at least the same number of sheets as the current Data Table.</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.

Data Command	Description
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 132.</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, e.g. 1/2.
Scientific	Sets format to scientific notation with two decimal places.
date (dynamic)	Sets format to Date with the M/D/YY format.
Time: h:mm AM/PM	Sets format to Time with the h:mm AM/PM format.
Custom Number	Sets format to a custom number format that you specify. This option enables you to set special and customized formats for percentages, currencies, dates, times, and so forth.

You can also perform Data Table menu commands using shortcut keys. For more information, see “Performing Commands Using Shortcut Keys” on page 45.

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 Data Table file will not affect the Data Table file in the file system and vice versa.

To use a Data Table file with Quality Center:

- 1** If you want to add a new Data Table file, create a new Microsoft Excel file in your file system with 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

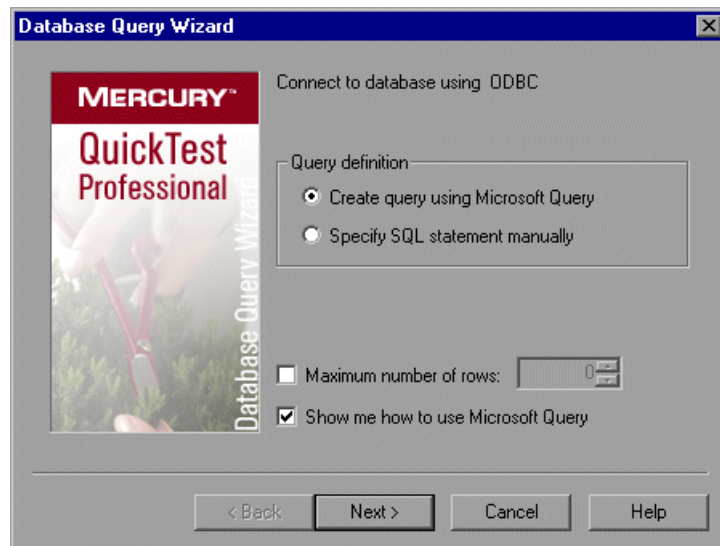
You can import data from a database by selecting a query from Microsoft Query or by manually specifying an SQL statement.

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*.

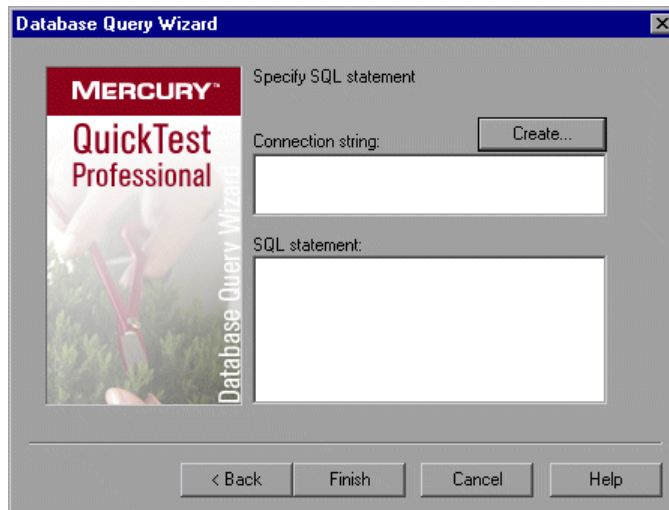
To import data from a database:

- 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 more information, see step 3.
 - **Maximum number of rows.** Select this check box and enter the maximum number of database rows to import. You can specify a maximum of 32,000 rows.
 - **Show me how to use Microsoft Query.** Displays an instruction screen before opening Microsoft Query when you click **Next**. (Enabled only when **Create query using Microsoft Query** is selected).
- 3 If you chose **Create query using Microsoft Query** in the previous step, Microsoft Query opens. Choose a data source and define a query. For more information on creating a query, see “Creating a Query in Microsoft Query,” 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 more information on working with Microsoft Query, refer to the Microsoft Query documentation.

Using Formulas in the Data Table

You can use Microsoft Excel formulas in your Data Table. This enables you to create contextually relevant data during the run session. You can also use formulas as part of a checkpoint to check that objects created on-the-fly (dynamically generated) or other variable objects in your Web page or application have the values you expect for a given context.

When you use formulas in a Data Table to compare values (generally in a checkpoint), the values you compare must be of the same type, for example integers, strings, and so forth. When you extract values from different places in your applications using different functions, the values may not be of the same type. Although these values may look identical on the screen, a comparison of them will fail, since, for example, 8.2 is not equal to "8.2".

Note: The use of complex and/or nested formulas in the Data Table is not supported.

You can use the **TEXT** and **VALUE** functions to convert values from one type to another as follows:

- **TEXT(value, format)** returns the textual equivalent of a numeric value in the specified format, so that, for example the formula `=TEXT(8.2, "0.00")` is "8.20".
- **VALUE(string)** returns the numeric value of a string, so that, for example, `=VALUE("$8.20")` is 8.20.

For more information on using worksheet functions, 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 remaining rows to enter the above date plus one day, as shown below. By using this formula you can run the test on any day and the dates will always be valid.

A2		=A1+1
	Date	B
1	1/3/2006	
2	1/4/2006	
3	1/5/2006	
4	1/6/2006	

For more information on using parameters, see Chapter 16, “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.

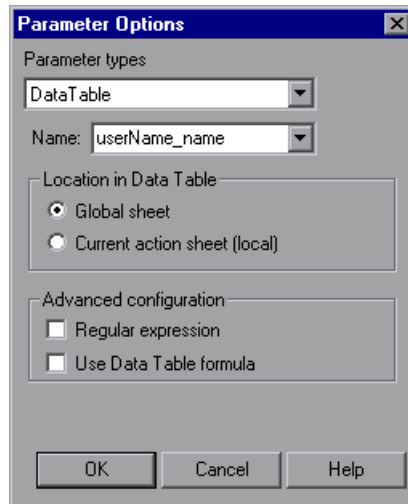
After you finish adding the checkpoint, you can modify the default formula in the first column to perform the check you need.

To use a formula in a checkpoint:

- 1 Select the object or text for which you want to create a checkpoint and open the Insert Checkpoint dialog box as described in Chapter 8, “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 8, “Understanding Checkpoints.”

- 7 Click **OK**. The two columns are added to the table, and the checkpoint step is inserted into your test.
- 8 Highlight the value in the first (formula) column to view the formula and modify the formula to fit your needs.
- 9 If you want to run several iterations, add the appropriate formula in subsequent rows of the formula column for each iteration in the test or action.

Tip: You can encode passwords to use the resulting strings as method arguments or Data Table parameter values. For more information, see “Inserting Encoded Passwords into Method Arguments and Data Table Cells” on page 132.

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 527.

Using Data Table Scripting Methods

QuickTest provides several Data Table methods that enable you to retrieve information on the run-time Data Table and to set the value of cells in the run-time Data Table. You enter these statements manually in the Expert View. For more information on working in the Expert View, see Chapter 34, “Working in the Expert View and Function Library Windows.”

From a programming perspective, the Data Table is made up of three types of objects—DataTable, DTSheet (sheet), and DTParameter (column). Each object has several methods and properties that you can use to retrieve or set values.

For more details on the Data Table methods, refer to the *Mercury QuickTest Professional Object Model Reference*.

21

Adding Steps Containing Programming Logic

After recording a test, you can use special QuickTest tools to enhance it with programming statements, even if you choose not to program manually in the Expert View.

This chapter describes:	On page:
About Adding Steps Containing Programming Logic	540
Inserting Steps Using the Step Generator	541
Using Conditional Statements	560
Using Loop Statements	566
Generating With Statements for Your Test	569
Generating Messages	575
Adding Comments	578
Synchronizing Your Test	579

About Adding Steps Containing Programming Logic

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 the power and flexibility of your test, 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 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.

This chapter shows you how to insert different types of statements, mostly from the Keyword View, aided by the Step Generator and other dialog boxes.

The Step Generator dialog box helps you add steps that use test object 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 34, “Working in the Expert View and Function Library Windows.”

You can incorporate decision-making into your test and define messages for the test results by using the appropriate dialog boxes.

In addition, you can improve the readability of your test using **With** statements. You can instruct QuickTest to automatically generate **With** statements as you record. But even after your basic test is recorded, you can convert its statements, in the Expert View, to **With** statements—by selecting a menu command.

You can handle synchronization issues between the run session and your application, using synchronization points.

When working with tests, you can also measure how long it takes certain parts of your test to run, using transaction statements.

Inserting Steps Using the Step Generator

The Step Generator enables you to add steps by selecting from a range of context-sensitive options and entering the required values. In the Step Generator dialog box you can define steps that use:

- test object methods and properties (tests only)
- Utility object methods and properties
- calls to library functions (tests only), VBScript functions, and internal script functions

For example, you can add a step that checks that an object exists, or that stores the returned value of a method as an output value or as part of a conditional statement. You can parameterize any of the values in your step.

Note: You can use the Step Generator to insert steps in tests and function libraries. However, in function libraries, you cannot use the Step Generator to access test object names or collections, or to access the list of library functions.

Before you open the Step Generator to define a new step, you first select where in your test the new step should be inserted. For more information on the hierarchy of steps and objects, see “Understanding the QuickTest Recorded Object Hierarchy” on page 118.

After you open the Step Generator, you first select the category for the step operation (test object, Utility object or function) and the required object or the function library source (for example, built-in or local script functions). You can then select the appropriate 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. 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. 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 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 544.

To open the Step Generator from a function library:

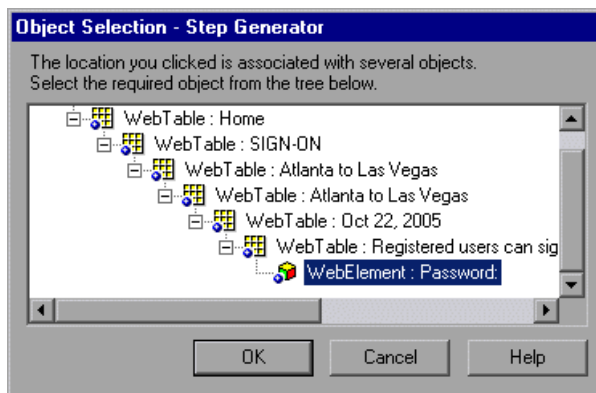
- 1** In the function library, click the location in which you want to insert the new step.
- 2** Choose **Insert > Step Generator**, or right-click and choose **Step Generator**. Alternatively, press F7.

The Step Generator dialog box opens. For more information, see “Defining Steps in the Step Generator Dialog Box” on page 544.

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**.

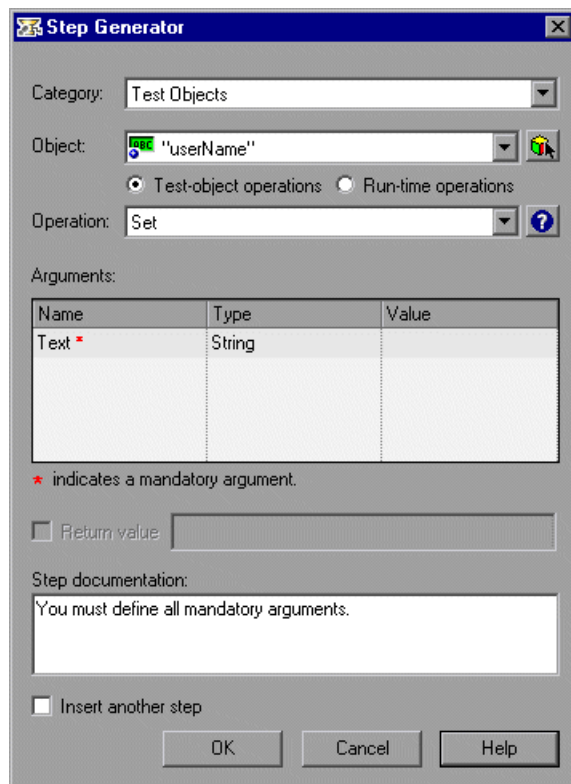
If the location you clicked is associated with more than one object, the **Object Selection - Step Generator** dialog box opens.



- 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 (for tests only), Utility object methods, or function calls.




Step Generator

Category: Test Objects

Object:  "userName" 

☒ Test-object operations ☐ Run-time operations

Operation: Set 

Arguments:

Name	Type	Value
Text *	String	

* indicates a mandatory argument.

☐ Return value

Step documentation:

You must define all mandatory arguments.

☐ Insert another step

OK Cancel Help

Note: The Step Generator dialog box that opens from the Expert View and from a function library is similar to the dialog box that opens from the Keyword View (shown in the example above).

In the Expert View, the Step Generator contains additional Utility objects and the box at the bottom of the dialog box shows a preview of the step that will be inserted in the Expert View. For more information, see “Viewing the Generated Step in the Expert View” on page 548.

In a function library, the Step Generator has a different title, contains only Utility objects and built-in and local script functions, and the box at the bottom of the dialog box shows a preview of the statement that will be inserted in the function library. For more information, see “Viewing the Generated Step in a Function Library” on page 548.

When the Step Generator dialog box opens, the object from the selected step is displayed in the **Object** box and the default method for the object is shown in the **Operation** box.

Defining a New Step

When you define a new step, you first select the type of step that you want to add to your test. You can then select the specific object and 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 or statement to your test or function library.

Note: Although the Step Generator shows information regarding the currently selected step, selections that you make in the Step Generator add a new step to your test; they do not modify the existing step.

Selecting the Type of Step to Add

In the **Category** list box, you can choose one of the following options:

- **Test Objects.** Enables you to select a test object and method for the step (for tests only). For more information, see “Specifying a Test Object and Method for the Step” on page 549.
- **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 554.
- **Functions.** Enables you to select a function for the step from the available library functions (tests only), VBScript functions, and internal script functions. For more information, see “Specifying a Function for the Step” on page 556.


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 pointer over the cell).

- **Required arguments.** If you specify a value for an optional argument, then you must also specify the values for any optional arguments that are listed before this argument. If you do not specify these values, QuickTest uses the default values for all required arguments. You can see the default value for each argument in a tooltip, by moving the pointer over the **Value** column.
- **Parameterized arguments.** You can use a parameter for any argument value by clicking the parameterization button . For more information, see “Configuring a Selected Value” on page 350.

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 557.

Viewing the Step Documentation in the Keyword View

If you open the Step Generator from the Keyword View, the **Step documentation** box at the bottom of the Step Generator dialog box can display summary information on the current step in an easy-to-read sentence.

If you select either the **Test Object** or **Utility Object** category and you define all the mandatory and required values for the current operation, the **Step documentation** box describes the operation performed by the step. When the step is inserted into your test, this description is displayed in the **Documentation** column in the Keyword View.

If all the mandatory and required argument values have not been defined for the operation, the **Step documentation** box displays a warning message.

Note: If you select the **Functions** category, step documentation is available for user-defined functions, if you provided this information when defining them. For more information, see “Documenting the Function” on page 1066.

Viewing the Generated Step in the Expert View

If you open the Step Generator from the Expert View, the **Generated step** box displays the defined statement for the step.

If all the mandatory and required argument values have not been defined for the operation, the names of the undefined arguments are highlighted in bold text. If you attempt to insert the step, an error message is displayed.

Viewing the Generated Step in a Function Library

If you open the Step Generator from a function library, the **Generated step** box displays the defined statement for the step.

If all the mandatory and required argument values have not been defined for the statement, the names of the undefined arguments are highlighted in bold text. If you attempt to insert the step, an error message is displayed.

Inserting Steps

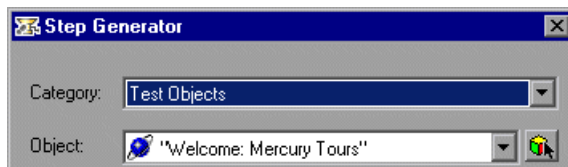
After you define all mandatory argument values for the current operation, the following options are available:

- To insert the current step and close the Step Generator, make sure the **Insert another step** check box is cleared. When you click **OK**, the step is added to your test and the Step Generator dialog box closes.
- To insert the current step and continue adding steps at the same location, select the **Insert another step** check box. The **OK** button changes to **Insert**. When you click **Insert**, the current step is added to your test and the Step Generator dialog box remains open, enabling you to define another step.

When you insert a new step using the Step Generator, it is added to your test after the selected step, and the new step is selected. For more information on the hierarchy of steps and objects and the positioning of new steps, see “Understanding the QuickTest Recorded Object Hierarchy” on page 118.

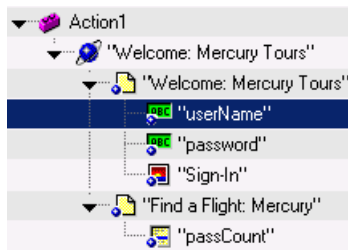
Specifying a Test Object and Method for the Step

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. Alternatively, you can select any object from the object repository or from your application.

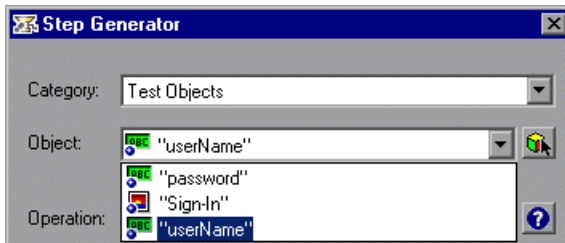


The list in the **Object** box contains all the objects in the object repository that are at the same hierarchical level and location as the currently selected step. You can select any of these objects for your new step.

For example, suppose that you selected the step for the **userName** object in the **Welcome: Mercury Tours** web page, as shown below:



When you open the Step Generator, **Test Objects** is selected in the **Category** box, and the **Object** box lists the **userName**, **password** and **Sign-in** objects.



Note: The objects are listed by name in alphabetical order.



You can select an object from the object repository or from your application, by clicking the **Select Object** button. For more information, see “Selecting an Object from the Repository or Application” on page 552.

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 on 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.

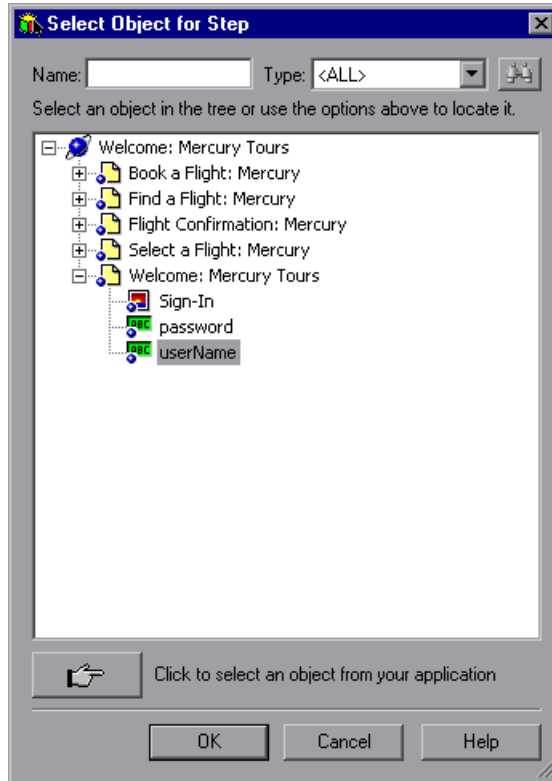
If you click the **Operation Help** button when a run-time method is selected, the *QuickTest Professional Object Model Reference* opens for the selected test object. For more information on specific run-time methods, refer to the documentation for the environment or application you are testing.

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 1029.

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 546.

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 6, “Working with 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.

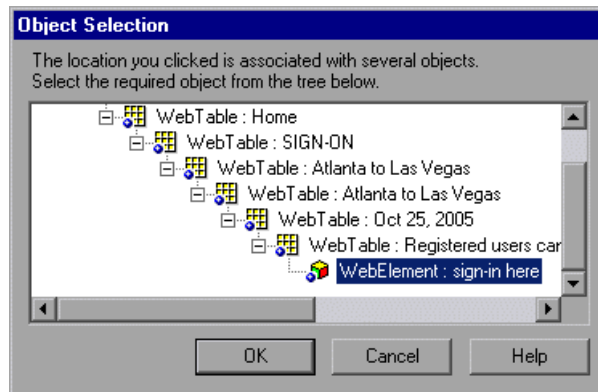
When you click **OK**, the selected object is displayed in the dialog box from which you opened the Select Object for Step dialog box.

To select an object in your application for the new step:

- 1** Click the pointing hand button. QuickTest is minimized, and your cursor changes to a pointing hand.
- 2** Use the pointing hand to click on the required object in your application.

Tip: You can hold the left CTRL key to change the window focus or perform operations such as right-clicking or moving the pointer over an object to display a context menu. If the window containing the object you want to 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. Additionally, if the window containing the object you want to select is minimized, you can display it by holding the left CTRL key, right-clicking the application in the Windows task bar, and choosing **Restore** from the context menu.

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

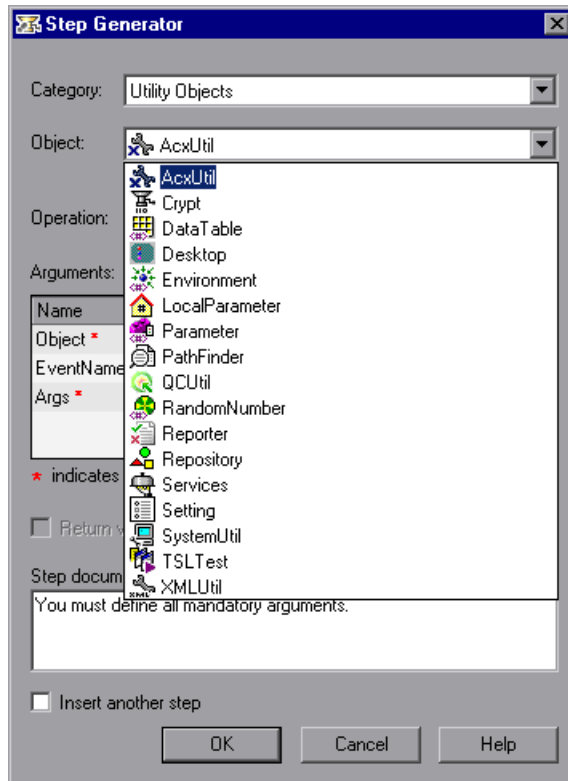


- 3 Select the object for the new step and click **OK**. The object is displayed in the dialog box from which you opened the Select Object for Step dialog box.

Tip: If you select an object in your application that is not in the object repository, a test object is added to the object repository when you insert the new step.

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 or a function library, the list includes a number of additional Utility objects. If you have one or more add-ins installed, the list may include additional Utility objects for those add-ins.

For more information on Utility objects, refer to the Utility Objects section of the *Mercury 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.

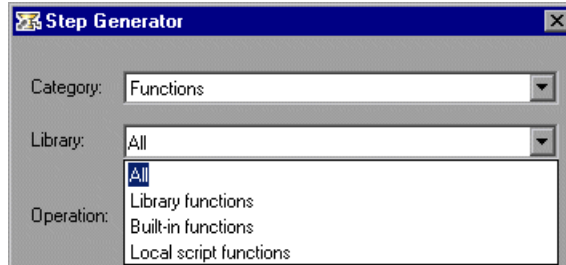


For detailed information on a Utility object method and its syntax, you can click the **Operation Help** button to open the *QuickTest Professional Object Model Reference* for the selected method.

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 546.

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 function library associated with your test (for tests only). For more information on defining and using associated function libraries, see “Working with Associated Function Libraries” on page 1052.
- **Built-in functions.** Enables you to select any standard VBScript function supported by QuickTest. For more information on working with VBScript, you can open the VBScript documentation from the QuickTest Help menu (**Help > QuickTest Professional Help > VBScript Reference**).
- **Local script functions.** Enables you to select any local function defined directly in the current action or function library.


You can select the required function from the **Operation** box list, which displays the functions available for the selected function type in alphabetical order.





For detailed information on a selected built-in VBScript function, you can click the **Operation Help** button to open Microsoft's VBScript Reference or the *QuickTest Professional Object Model Reference*. This option is not available for library and local script functions.

After you select the function for the operation, you can define the relevant argument values. For more information, see “Specifying Argument Values” on page 546.

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.



The Storage Location Options dialog box is shown with the following settings:

- Variable:** ☒ Variable:
- Output Type:** ☐ Output Type:
- Name:**
- Location in Data Table:**
 - ☒ Global sheet
 - ☐ Current action sheet (local)

Buttons: OK, Cancel, Help

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 test or action output parameter, Data Table column or environment variable. You can specify the output type and settings as for any other output value.

When a return value or a test or action output parameter is first selected, the default output definition for the value is displayed. For more information on default output definitions for a return value, see “Understanding Default Output Definitions” on page 425.

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 Parameter” on page 426
- “Outputting a Value to the Data Table” on page 427
- “Outputting a Value to an Environment Variable” on page 428

Understanding Default Output Definitions for Action Parameter Values

When you select **Output Type** for an output action parameter value for a nested action:

- If at least one output action parameter is defined in the action calling the nested action, the default output type is **Test/action parameter** and the default output name is the first output parameter displayed in the Action Properties dialog box of the calling action.
- If no output action parameters are defined in the calling action, the default output type is Data Table and QuickTest creates a new Data Table output name based on the selected value. The value is created in the Global sheet of the Data Table.

When you select **Output Type** for an output action parameter value for a top-level action:

- If at least one output action parameter is defined in the test, the default output type is **Test/action parameter** and the default output name is the first output parameter displayed in the Test Properties dialog box.
- If no output action parameters are defined in the test, the default output type is **Data Table** and QuickTest creates a new Data Table output name based on the selected value. The value is created in the Global sheet of the Data Table.

Using Conditional Statements

You can control the flow of your test with conditional statements. Using conditional **If...Then...Else** statements, you can incorporate decision-making into your tests.

The **If...Then...Else** statement is used to evaluate whether a condition is true or false and, depending on the result, to specify one or more statements to run. Usually the condition is an expression that uses a comparison operator to compare one value or variable with another. The following comparison operators are available: less than <, less than or equal to <=, greater than >, greater than or equal to >=, not equal <>, and equal =.

Your **If...Then...Else** statement can be nested to as many levels as you need. It has the following syntax:

If *condition* **Then** *statements* [**Else** *elstatements*] **End If**

Or, you can use the block form syntax:

```
If condition Then
    statements
[Elseif condition-n Then
    elseifstatements] . . .
[Else
    elstatements]
End If
```

For example:

'Set the focus on (activate) the Open Order dialog box

```
Window("Flight Reservation").Dialog("Open Order").Activate
```

'Insert a check mark in the Order No. check box

```
Window("Flight Reservation").Dialog("Open Order").WinCheckBox("Order No.").  
Set "ON"
```

Insert an order number in the displayed box and save the value as "OrderNo" for use later in the script. If the value is less than or equal to 0, generate a message box. (If the value is illegal and a message box is generated, end the run session when the user clicks OK.)

```
OrderNo = InputBox("Enter Order Number")
```

```
If OrderNo <= 0 Then
```

```
Msgbox "You entered an invalid order number."
```

```
ExitAction
```

```
End If
```

'Insert the saved order number value in the Order No. box

```
Window("Flight Reservation").Dialog("Open Order").WinEdit("OrderNumber  
Edit").Set OrderNo
```

'Click OK to close the Open Order dialog box

```
Window("Flight Reservation").Dialog("Open Order").WinButton("OK").Click
```

'Check if an error message opens and send a report to the test results

```
If Window("Flight Reservation").Dialog("Open Order").Dialog("Flight  
Reservations").Exist Then
```

```
Reporter.ReportEvent micFail, "Check that the value of the order  
number is legal", "The order number does not exist."
```

```
Window("Flight Reservation").Dialog("Open Order").Dialog("Flight  
Reservations").WinButton("OK").Click
```

```
Else
```

```
Reporter.ReportEvent micPass, "Check that the value of the order  
number is legal", "The order number exists."
```

```
End If
```

The above example checks whether the application being tested will identify whether a valid order number is being entered in the Order No. box in the Open Order dialog box.

To do this, QuickTest activates the Open Order dialog box (brings it into focus), selects the Order No. check box, and opens a box in which the user inserts a value—the relevant order number—and clicks **OK**. The first conditional statement instructs QuickTest to verify that the value entered by the user is greater than zero. **If** it is not greater than zero, QuickTest opens a message box stating that the value entered is invalid. When the user clicks **OK** to close the message box, QuickTest ends the run session.

Otherwise, if the value entered is greater than zero, QuickTest inserts the above value in the Order No. box.

The next **If** statement instructs QuickTest to check whether the order number exists in the application and to send a report to the Test Results indicating that the step passed or failed. If the step failed because the order number was invalid, the Flight Reservations error message opens, and QuickTest clicks **OK** to close this message box before ending the run session.

Note: You can insert conditional statements in the Expert View and in the Keyword View. You can also switch between the views, as needed. For information on working with conditional steps in the Expert View, see “Using Comments, Control-Flow, and Other VBScript Statements” on page 1019, and refer to the VBScript documentation (choose **Help > QuickTest Professional Help > VBScript Reference**).

To insert a conditional statement in the Keyword View:

- 1 In the Keyword View, select the step that you want the conditional statement to follow.

The following example shows the `userName` row selected:

Welcome: Mercury Tours			
Welcome: Mercury Tours			
userName	Set	"Mercury"	Enter "Mercury" in the "userName" edit box.
password	SetSecure	"43c1028da...	Enter the encrypted string "43c1028da7b83180"
Sign-In	Click	14,6	Click the "Sign-In" image.


- 2 Choose **Insert > Conditional Statement** and choose **If...Then**. The new statement is added to the Keyword View below the selected step. For example:

Welcome: Mercury Tours			
Welcome: Mercury Tours			
userName	Set	"Mercury"	Enter "Mercury" in the "userName" edit box.
IF Statement		True	Check whether (True) is true. If so:
password	SetSecure	"43c1028da...	Enter the encrypted string "43c1028da7b83180"
Sign-In	Click	14,6	Click the "Sign-In" image.

Note: Each statement type is indicated by one of the following icons:

 (If...Then statement)

 (Elseif...Then statement)

 (Else statement)

- 3 Click in the **Item** cell for the **If** statement. Then click the down arrow and select the object on which you want to perform the conditional statement. For example:

Welcome: Mercury Tours			
Welcome: Mercury Tours			
userName	Set	"Mercury"	Enter "Mercury" in the "userName" edit box.
IF userName	Set		<No documentation summary is available for the c
password	SetSecure	"43c1028da...	Enter the encrypted string "43c1028da7b83180"
Sign-In	Click	14,6	Click the "Sign-In" image.

- 4 Click in the **Operation** cell and select the operation you want to perform. For example:

Welcome: Mercury Tours			
Welcome: Mercury Tours			
userName	Set	"Mercury"	Enter "Mercury" in the "userName" edit box.
IF userName	Exist		Check whether the "userName" edit box exists
password	SetSecure	"43c1028da...	Enter the encrypted string "43c1028da7b8318
Sign-In	Click	14,6	Click the "Sign-In" image.

- 5 If needed, click in the **Value** cell and enter the required condition. (In this example, because we are using the **Exist** method, it is not necessary to add a value to the **Value** cell.)
- 6 Insert a **Then** statement by selecting the **If** statement step and inserting a new statement (**Insert > New Step**) or by recording a new step. For example:

Welcome: Mercury Tours			
Welcome: Mercury Tours			
userName	Set	"Mercury"	Enter "Mercury" in the "userName" edit box.
IF userName	Exist		Check whether the "userName" edit box exists
userName	Set	DataTable("p_Us...	Enter <the value of the 'p_UserName' Data
password	SetSecure	"43c1028da7b83...	Enter the encrypted string "43c1028da7b8318
Sign-In	Click	14,6	Click the "Sign-In" image.

Make sure you set the values for the new step in the **Operation** and **Value** columns.

- 7 Delete the row immediately above the **If** statement. For example:

Welcome: Mercury Tours			
Welcome: Mercury Tours			
IF userName	Exist		Check whether the "userName" edit box exists. If so
userName	Set	DataTable("p_Us...	Enter <the value of the 'p_UserName' Data Table
password	SetSecure	"43c1028da7b83...	Enter the encrypted string "43c1028da7b831801c87
Sign-In	Click	14,6	Click the "Sign-In" image.

- 8 You can now complete the statement with an **Else** statement, or you can nest an additional level in your statement. To do this, select the **If** statement and choose one of the following options:

To add:	Choose:
an If statement	Insert > Conditional Statement > If...Then
an Elseif statement	Insert > Conditional Statement > Elseif...Then
an Else statement	Insert > Conditional Statement > Else

For example, the statements below check that the User Name edit box exists in the Mercury Tours site. **If** the edit box exists, **Then** a user name is entered; **Else** a message is sent to the Test Results.

Welcome: Mercury Tours			
<ul style="list-style-type: none"> Welcome: Mercury Tours <ul style="list-style-type: none"> If userName <ul style="list-style-type: none"> userName Else <ul style="list-style-type: none"> Statement 	Exist		Check whether the "userName" edit box exists. If .
	Set	DataTable('p_Us...	Enter <the value of the 'p_UserName' Data Tab
			Otherwise:
Reporter	ReportEvent	micFail,"UserNam...	Report "The UserName 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 DataTable ("p_UserName", dtGlobalSheet)
Else
    Reporter.ReportEvent micFail, "UserName Check", "The User Name field
        does not exist."
End If
```

- 9 After you have finished creating the conditional statement, use the **Insert Step After Block** option if you want to insert a step outside of the conditional statement block. For more information, see “Adding a Standard Step After a Conditional or Loop Block” on page 135.

Using Loop Statements

You can control the flow of your test with loop statements. Using loop statements, you can run a group of steps repeatedly, either while or until a condition is True. You can also use loop statements to repeat a group of steps a specific number of times.





The following loop statements are available in the Keyword View:

- **While...Wend.** Performs a series of statements as long as a specified condition is True.
- **For...Next.** Uses a counter to perform a group of statements a specified number of times.
- **Do...While.** Performs a series of statements indefinitely, as long as a specified condition is True.
- **Do...Until.** Performs a series of statements indefinitely, until a specified condition becomes True.

Note: For more information on loop statements, refer to the VBScript documentation (choose **Help > QuickTest Professional Help > VBScript Reference**).

To insert a loop statement in the Keyword View:

- 1** Select the step that you want the loop statement to follow.
- 2** Choose **Insert > Loop Statement** and choose the statement type that you want to insert from the sub-menus. The new statement is added to the Keyword View below the selected step. Each statement type is indicated by one of the following icons:

Icon	Type
	While...Wend statement
	For...Next statement
	Do...While statement
	Do...Until statement

- 3** In the **Value** column, enter the required condition, for example:
For i = 0 to ItemsCount - 1
- 4** To complete the loop statement, you can:
 - Select the loop statement step and record a new step to add it to your loop statement.
 - Select the loop statement step and choose **Insert > New Step** or press F8 to insert a new step into your loop statement.

Note: For more information on working in the Expert View, see Chapter 34, “Working in the Expert View and Function Library Windows.”

The following example counts the number of items in a list and then selects them one by one. After each of the items has been selected, the test continues.

Find a Flight: Mercury Tours:			
Find a Flight: Mercury T...			
toDay	GetROProperty	"items count"	Retrieve the current value of the "items count"
Statement		For i = 1 To ItemsCount - 1	Repeat the following step(s) one or more times
toDay	GetItem	i	Retrieve the value of the item with index i
toDay	Select	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 135.

Generating With Statements for Your Test

You can instruct QuickTest to automatically generate **With** statements when you record a test or to generate **With** statements for any existing action. You can also remove **With** statements from an action.

Note: Using **With** statements in your test has no effect on the run session itself, only on the way your test appears in the Expert View. Generating **With** statements for your test does not affect the Keyword View in any way.

Understanding With Statements

With statements make your script (in the Expert View) more concise and easier to read by grouping consecutive statements with the same parent hierarchy.

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

```
With object  
    statement  
    statement  
    statement  
End With
```

For example, you could replace this script:

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

with the following:

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

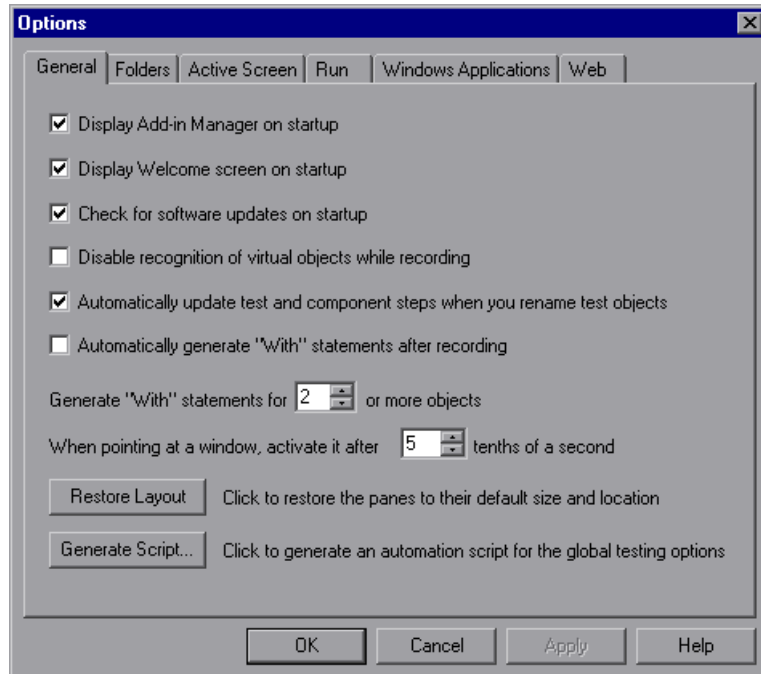
Automatically Generating With Statements

You can instruct QuickTest to automatically generate **With** statements for the steps you record. When you select this option, statements are displayed in their normal format while recording. When you stop recording, the statements in all actions recorded during the current recording session are automatically converted to the **With** format.

To generate With statements automatically when you record:



- 1 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. 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.

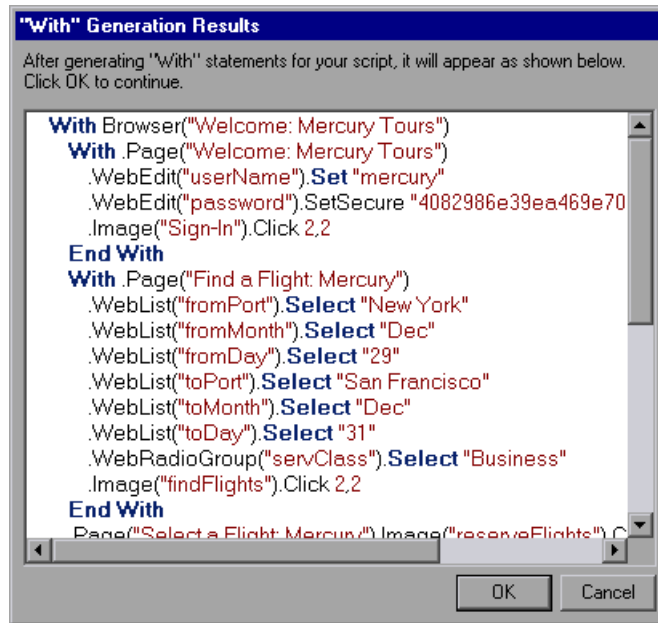
Generating With Statements for Existing Actions

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 > Advanced > Apply "With" to Script**. The "With" Generation Results window opens.



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 990.

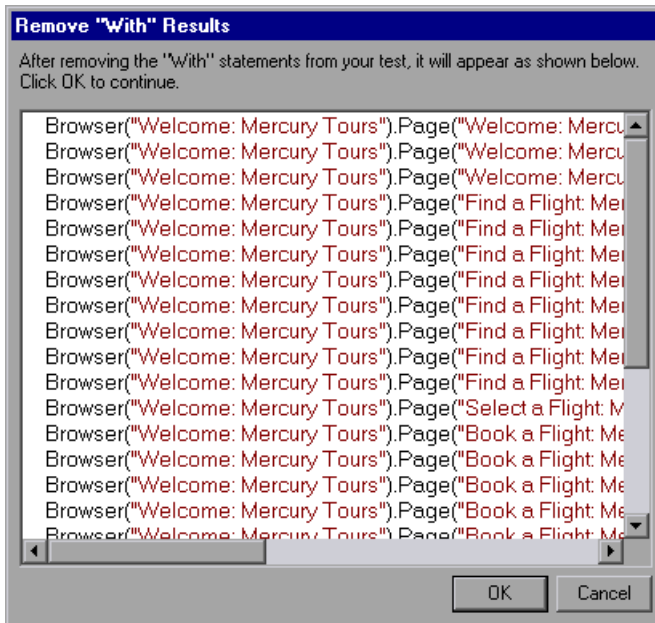
- 4 To confirm the generated results, click **OK**. The **With** statements are applied to the action.

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 > Advanced > Remove “With” Statements**. The Remove “With” Results window opens.



- 3 To confirm the results, click **OK**. The **With** statements are replaced with the standard statement format.

Generating Messages

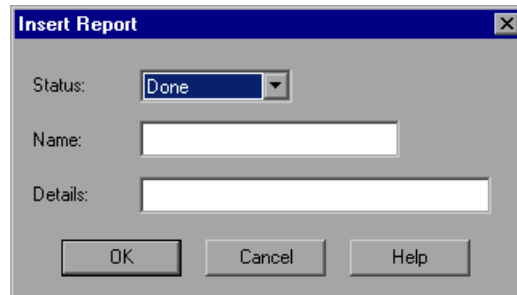
You can generate messages in your test that are displayed in the Test Results window. You can also choose to display messages on screen while running your test.

Sending Messages to the Test Results

You can define a message that QuickTest sends to your test results. For example, suppose you want to check that a password edit box exists in the Mercury Tours site. If the edit box exists, then a password is entered. Otherwise, QuickTest sends a message to the test results indicating that the object is absent.


To send a message to your test results:

- 1 In the Keyword View, select a step and choose **Insert > 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 itself) to fail. Sends the specified message to the report.
Done	Sends a message to the report without affecting the pass/fail status of the step.
Warning	Sends a warning status for the step, but does not cause the test to stop running, and does not affect its pass/fail status.

- 3 In the **Name** box, type a name for the step, for example, **Password edit box**.
- 4 In the **Details** box, type a detailed description of this step to send to your test results, for example, **Password edit box does not exist**.
- 5 Click **OK**. A report step is inserted into the Keyword View  and a **Reporter.ReportEvent** statement is inserted into your script in the Expert View. For example:

```
Reporter.ReportEvent micFail, "Password edit box", "Password edit box does not exist"
```

In this example, `micFail` indicates the status of the report (failed), `Password edit box` is the report name, and `Password edit box does not exist` is the report message.

For more information on test results, see Chapter 24, “Analyzing Test Results.”

Note: After you add a report step, you can modify it in the Keyword View by right-clicking the step and choosing **Report Properties**, or by modifying any of the arguments in the **Value** column. (You can also modify the **Reporter.ReportEvent** statement directly in the Expert View.)

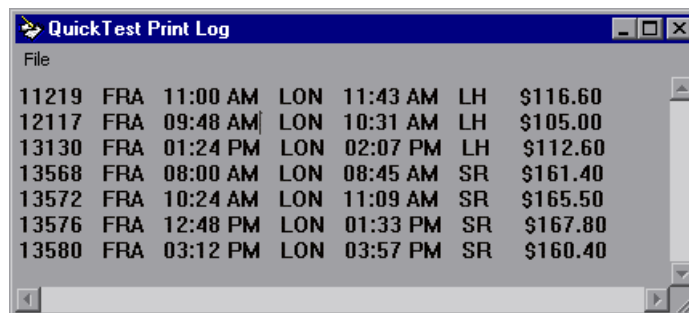
Displaying Messages During the Run Session

In addition to sending messages to the Test Results, you can generate messages in the following ways:

- Use the **MessageBox** VBScript function in your test to display information during the run session. The run session pauses until the message box is closed. For more information, refer to the VBScript documentation from the QuickTest Help menu (**Help** > **QuickTest Professional Help** > **VBScript Reference**).
- Use the **Print** Utility statement in your test to display information in the QuickTest Print Log window while still continuing the run session. For example, the following example iterates all the items in the **Flight Table** dialog (in the sample Flight application) and uses the **Print** Utility statement to print the content of each item to the QuickTest Print Log window.

```
Set FlightsList = Window("Flight Reservation").Dialog("Flights Table").
    WinList("From")
For i = 1 to FlightsList.GetItemsCount
    Print FlightsList.GetItem(i - 1)
Next
```

The QuickTest Print Log window remains open throughout the run session, until you close it.

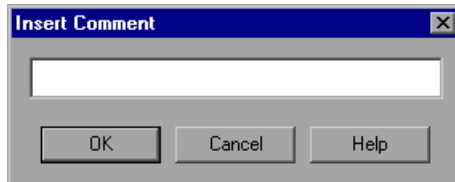


Adding Comments

While editing your test, you can add comments in the Keyword View or in the Expert View. You can also add comments to function libraries. A comment is an explanatory remark in a program. When you run a test, QuickTest does not process comments. You can use comments to explain sections of your tests to improve readability and to make them easier to update. You can add comments directly to the Keyword View or the Expert View, or you can use the Insert Comment dialog box. You can also modify comments at any time directly in the Keyword View or the Expert View, or using the Comment Properties dialog box.

To add a comment in the Keyword View:

- 1 If the **Comment** column is not visible, right-click any column header and select **Comment**.
- 2 Add a comment in one of the following ways:
 - To add a comment on the same line as a step, select the step and type your comment in the **Comment** column.
 - To add a comment on a separate line, select a step and choose **Insert > Comment**, or right-click a step and choose **Insert Step > Comment**. The Insert Comment dialog box opens. Type a comment and click **OK**. A comment statement is added to your test.



In the Keyword View, the  icon indicates a comment.

To add a comment in the Expert View or a function library:

Type an apostrophe (') and then type your comment. You can add a comment at the end of a line or at the beginning of a separate line.

To modify a comment:

- In the Keyword View, you can modify the comment text directly in the **Comment** column, or you can right-click any column in the step and choose **Comment Properties** to open the Comment Properties dialog box (which is similar to the Insert Comment dialog box).
- In the Expert View, you can overwrite any existing comment.

Tip: If you want to add the same comment to every action that you create, you can add the comment to an action template. For more information, see “Creating an Action Template” on page 503.

Synchronizing Your Test

When you run a test, your application may not always respond with the same speed. For example, it might take a few seconds:

- for a progress bar to reach 100%
- for a status message to appear
- for a button to become enabled
- for a window or pop-up message to open

You can handle these anticipated timing problems by synchronizing your test to ensure that QuickTest waits until your application is ready before performing a certain step.

There are several options that you can use to synchronize your test:

- You can insert a **synchronization point**, which instructs QuickTest to pause the test until an object property achieves the value you specify. When you insert a synchronization point into your test, QuickTest generates a **WaitProperty** statement in the Expert View.

- You can insert **Exist** or **Wait** statements that instruct QuickTest to wait until an object exists or to wait a specified amount of time before continuing the test.
- You can modify the default amount of time that QuickTest waits for a Web page to load.
- When working with tests, you can increase the default timeout settings for a test to instruct QuickTest to allow more time for objects to appear.

Creating Synchronization Points

If you do not want QuickTest to perform a step or checkpoint until an object in your application achieves a certain status, you should insert a synchronization point to instruct QuickTest to pause the test until the object property achieves the value you specify (or until a specified timeout is exceeded).

For example, suppose you record a test on a flight reservation application. You insert an order, and then you want to modify the order. When you click the **Insert Order** button, a progress bar is displayed and all other buttons are disabled until the progress bar reaches 100%. Once the progress bar reaches 100%, you record a click on the **Update Order** button.

Without a synchronization point, QuickTest may try to click the **Update Order** button too soon during a test run (if the progress bar takes longer than the test's object synchronization timeout), and the test will fail.

You can insert a synchronization point that instructs QuickTest to wait until the **Update Order** button's **enabled** property is 1.

Tip: QuickTest must be able to identify the specified object to perform a synchronization point. To instruct QuickTest to wait for an object to open or appear, use an **Exist** or **Wait** statement. For more information, see “Adding Exist and Wait Statements” on page 583.

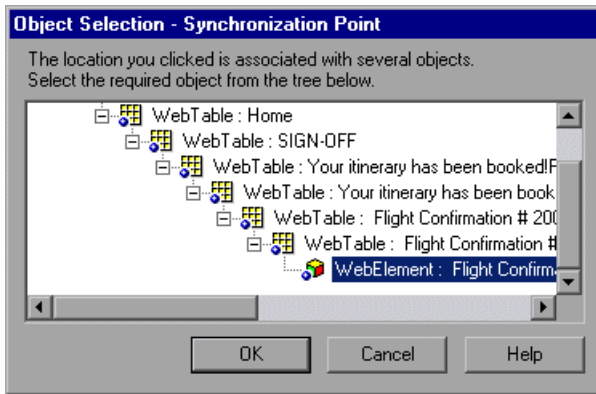
To insert a synchronization point:

- 1** Begin recording your test.
- 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 > Synchronization Point**. The 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 left CTRL key to change the window focus or perform operations such as right-clicking or moving the pointer over an object to display a context menu to select the required object. If the window containing the object you want to select is minimized, you can display it by holding the left CTRL key, right-clicking the application in the Windows task bar, and choosing **Restore** from the context menu.

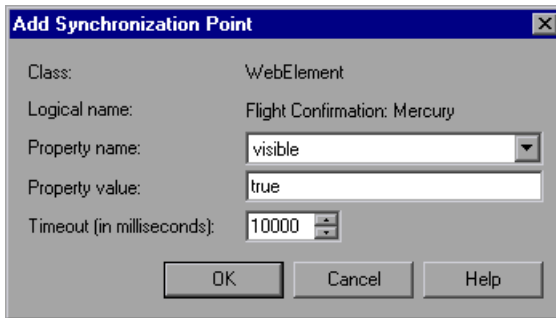
Note: It does not matter what property values the object has at the time that you insert the synchronization point.

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.
- 7** Enter the synchronization point timeout (in milliseconds) after which QuickTest should continue to the next step, even if the specified property value was not achieved.
- 8** Click **OK**. A **WaitProperty** step is added to your test.

Because the `WaitProperty` step is a method of the selected object, it is displayed in the Keyword View with the icon for the selected object. For example, if you insert a synchronization point for the **Update Order** button, it may look something like this:

▼ Flight Confirmation: Mercury	Sync		Wait for the Web page to synchronize before
Flight Confirmation: Mercury	WaitProperty	"visible",true,10000	Check whether the value of the "visible" pr

In the Expert View, this appears as:

```
Browser("Welcome: Mercury Tours").Page("Flight Confirmation: Mercury").Sync
Browser("Welcome: Mercury Tours").Page("Flight Confirmation: Mercury").
    WebElement("Flight Confirmation #").WaitProperty "visible", true, 10000
```

For more information on the **WaitProperty** method, refer to the *Mercury QuickTest Professional Object Model Reference*.

Adding Exist and Wait Statements

You can enter **Exist** and/or **Wait** statements to instruct QuickTest to wait for a window to open or an object to appear. Exist statements return a boolean value indicating whether or not an object currently exists. Wait statements instruct QuickTest to wait a specified amount of time before proceeding to the next step. You can combine these statements within a loop to instruct QuickTest to wait until the object exists before continuing with the test.

For example, the following statements instruct QuickTest to wait up to 20 seconds for the Flights Table dialog box to open.

```
blnDone=Window("Flight Reservation").Dialog("Flights Table").Exist
counter=1
While Not blnDone
    Wait (2)
    blnDone=Window("Flight Reservation").Dialog("Flights Table").Exist
    counter=counter+1
    If counter=10 then
        blnDone=True
    End if
Wend
```

For more information on While, Exist, and Wait statements, refer to the *Mercury QuickTest Professional Object Model Reference*.

Modifying Timeout Values

If you find that, in general, the amount of time QuickTest waits for objects to appear or for a browser to navigate to a specified page is insufficient, you can increase the default object synchronization timeout values for your test and the browser navigation timeout values for your test.

Alternatively, if you insert synchronization points and **Exist** and/or **Wait** statements for the specific areas in your test where you want QuickTest to wait a longer time for an event to occur, you may want to decrease the default timeouts for the rest of your test.

- When working with tests, to modify the maximum amount of time that QuickTest waits for an object to appear, change the **Object Synchronization Timeout** in the **File > Settings > Run** tab. For more information, see “Defining Run Settings for Your Test” on page 763.
- To modify the amount of time that QuickTest waits for a Web page to load, change the **Browser Navigation Timeout** in the **File > Settings > Web** tab. For more information, see “Defining Web Settings for Your Test” on page 782.

Part III

Running and Debugging Tests

22

Debugging Tests and Function Libraries

By controlling and debugging your run sessions, you can identify and handle defects in your tests, function libraries, and registered user functions.

This chapter describes:	On page:
About Debugging Tests and Function Libraries	588
Slowing a Debug Session	590
Using the Single Step Commands	590
Using the Run to Step and Start from Step Commands	593
Pausing a Run Session	595
Using Breakpoints	596
Using the Debug Viewer	600
Handling Run Errors	602
Practicing Debugging an Action or a Function	603

About Debugging Tests and Function Libraries

After you create a test or function library (including registered user functions), you should check that they run smoothly, without errors in syntax or logic. To debug a function library, you must first associate it with a test and then debug it from that test.

To detect and isolate defects in a test or function library, you can control the run session using the **Pause** command as well as various step commands that enable you to step into, over, and out of a specific step.

You can use the **Start from Step** command to begin your debug session at a specific point in your test. You can also use the **Run to Step** command to pause the run at a specific point in your test. You can set breakpoints, and then enable and disable them as you debug different parts of your test or function library.

When the test or function library run 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 run and debug the test or function library.

You can also use the **Run from Step** command to run your test or function library from a selected step to the end. This enables you to check a specific section of your application or to confirm that a certain part of your test or function library runs smoothly. For more information, see “Running Part of Your Test” on page 613.

Notes:

- While the test and function libraries are running in debug mode, they are read-only. You can modify the content after you stop the debug session (not when you pause it). If needed, you can enable the function library for editing (**File > Enable Editing**) after you stop the session. For more information, see “Editing a Read-Only Function Library” on page 1049. After you implement your changes, you can continue debugging your test and function libraries.
- If you perform a file operation (for example, open a different test or create a new test), the debug session is stopped.
- You cannot debug a file that is called using an **ExecuteFile** statement, or any of the functions contained in the file. In addition, when debugging a test that contains an **ExecuteFile** statement, the execution marker may not be correctly displayed.
- In QuickTest, when you open a test, QuickTest creates a local copy of the external resources that are saved to your Quality Center project. Therefore, any changes you apply to any external resource that is saved in your Quality Center project, such as a function library, will not be implemented in the test until the test is closed and reopened. (An external resource is any resource that was not created using QuickTest, such as, a function library created in an external editor.)

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

Slowing a Debug Session

During a run session, QuickTest normally runs steps quickly. While you are debugging a test or function library, you may want QuickTest to run the steps more slowly so you can pause the run when needed or perform another task. You can specify the time (in milliseconds) QuickTest pauses between each step by modifying the **Delay each step execution by** option in the Run tab of the Options dialog box (**Tools > Options**). For more information on the Run tab options, see “Setting Run Testing Options” on page 723.

Using the Single Step Commands

You can run a single step of a test or function library using the **Step Into**, **Step Out**, and **Step Over** commands.

Tip: To display the Debug toolbar, 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 function library. If the current line of the active test or function library calls another action or a function, the called action/function is displayed in the QuickTest window, and the test or function library pauses at the first line of the called action/function.

Step Out



Choose **Debug > Step Out**, click the **Step Out** button, or press SHIFT+F11 only after using **Step Into** to enter an 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 test or function library and pauses the run session.

Step Over



Choose **Debug > Step Over**, click the **Step Over** button, or press F10 to run only the current step in the active test or function library. 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 or function script is not displayed in the QuickTest window.

Using the Single Step Commands - An Example

Follow the instructions below to create a sample test or function library and run it using the **Step Into**, **Step Out**, and **Step Over** commands.

To create the sample test or function library:

- 1** Choose **File > New > Test** to open a new test or **File > New > Function Library** to open a new function library (in addition to your test).
- 2** If you created a new test, click the **Expert View** tab to display the Expert View.
- 3** In the test or function library, enter the following lines exactly:

```
public Function myfunc()
msgbox "one"
msgbox "two"
msgbox "three"
End Function
```


```
myfunc
myfunc
myfunc
```

4 Perform one of the following:

- If you created a test, skip to the next set of instructions describing how to run the test or function library using the **Step Into**, **Step Out**, and **Step Over** commands.
- If you created a function library, save it to the file system or your Quality Center project with the name **SampleFL.qfl**. (For more information, see “Saving a Function Library” on page 1044.)

Then choose **File > Associate Library '<Function Library Name>' with '<Test Name>'** to associate the function library with your test.

To run the test or function library using the Step Into, Step Out, and Step Over commands:

- 1** Open the test or **SampleFL.qfl** function library, if it is not already open, or click the tab for the test or **SampleFL.qfl** function library to bring it into focus.
-  **2** Add a breakpoint on the seventh line of the test or function library (the first call to the **myfunc** function) by pressing F9 (**Insert/Remove Breakpoint**). The breakpoint symbol is displayed in the left margin. For more information, see “Setting Breakpoints” on page 597.
- 3** Run the test. The test or function library pauses at the breakpoint.
- 4** Press F11 (**Step Into**). The execution arrow points to the first line within the function (**msgbox "one"**).
- 5** Press F11 (**Step Into**) again. A message box displays the text **one**.
- 6** Click **OK** to close the message box. The execution arrow moves to the next line in the function.
- 7** 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).
- 8** Press F11 (**Step Into**) to enter the function again. The execution arrow points to the first **msgbox** line within the function.

- 9 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.
- 10 Press F10 (**Step Over**). The three message boxes open again. The execution arrow remains on the last line in the test.

Using the Run to Step and Start from Step Commands

In addition to stepping into, out of, and over a step while debugging, you can use the **Run to Step** and **Start from Step** commands to instruct QuickTest to run a test or action (including any associated function library) until it reaches a particular step, or to begin debugging from a specific step.

Run to Step

You can instruct QuickTest to run from the beginning of the test or action (Expert View only)—or from the current location in the test or action—and to stop at a particular step. This is similar to adding a temporary breakpoint to a step. For example, if you are running a test or action and any associated function library in debug mode, one step at a time, you may want to run four consecutive steps and then stop at the fifth step.

You can use this option while editing or debugging your test or action.

To instruct QuickTest to run to a particular step:

- Insert your cursor in the step in which you want QuickTest to stop the run and choose **Debug > Run to Step** or press CTRL+F10
- Right-click in the step in which you want QuickTest to stop the run and choose **Run to Step** from the context menu

Note: If while editing your test, you use the **Run to Step** option, the Run dialog box opens, enabling you to specify the results location and the input parameter values for the debug run session. For more information, see step 2 in the “Start from Step” section, below.

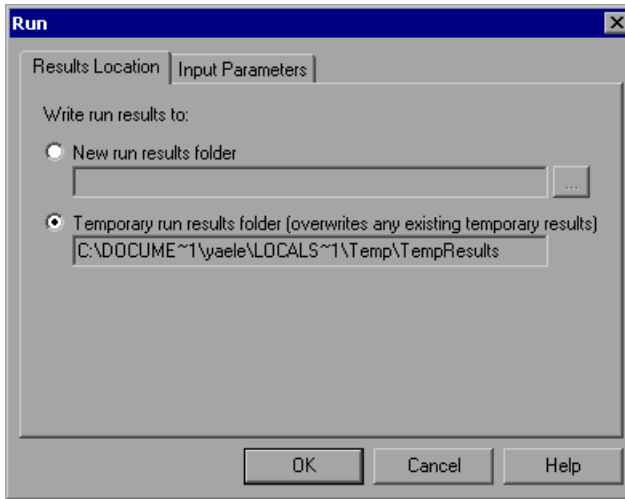
Start from Step

You can instruct QuickTest to begin your debug session from a particular step instead of beginning the run at the start of the test or action. Before you start debugging from a specific step, make sure that the application or Web site is open to the location from which you want to begin debugging. You can begin debugging from a specific step in your test or action when editing a test or action.

To instruct QuickTest to run from a particular step:

- 1 Select the step from which you want to begin debugging:
 - Insert your cursor in the step where you want QuickTest to start the run and choose **Debug > Start from Step**, or
 - Right-click in the step where you want QuickTest to start the run and choose **Debug from Step** from the context menu.

The Run dialog box opens.



Note: If your QuickTest test is saved to a Quality Center project (as opposed to the file system), the Results Location tab of the Run dialog box contains additional project-related options. For more information, see “Running a Test Stored in a Quality Center Project from QuickTest” on page 1285.

- 2 If applicable, specify the results location and the input parameter values for the debug run session. By default, the **Temporary run results folder** option is selected.

For more information on the tabs in the Run dialog box, see “Understanding the Results Location Tab” on page 611, and “Understanding the Input Parameters Tab” on page 612.

- 3 Click **OK**. The Run dialog box closes and the debug run session starts. You can use any of the QuickTest debugging options, such as **Step Into**, **Step Over**, and **Run to Step**.

By default, when the run session ends, the Test Results window opens. For more information on viewing the run results, see Chapter 24, “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 25, “Setting Global Testing Options.”

Pausing a Run Session



You can temporarily suspend a run session by choosing **Debug > Pause** or clicking the **Pause** button. A paused test or function library stops running when all previously interpreted steps have been run.



To resume running a paused run, click the **Run** button, choose **Automation > Run**, or press **F5**. The run continues from the point it was suspended.



Tip: You can also stop a run session by clicking the **Stop** button or choosing **Automation > Stop**. After the run session stops, the Test Results window opens (unless you selected not to view results at the end of a run session (**Tools > Options > Run** tab)).

Using Breakpoints

You can use breakpoints to instruct QuickTest to pause a run session at a predetermined place in a test or function library. QuickTest pauses the run when it reaches the breakpoint, before executing the step. You can then examine the effects of the run up to the breakpoint, make any necessary changes, and continue running the test or function library from the breakpoint.

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 function library using the step commands

You can set breakpoints, and you can temporarily enable and disable them. After you finish using them, you can remove them from your test or function library.

Note: Breakpoints are applicable only to the current QuickTest session and are not saved with your test or function library.

Setting Breakpoints

By setting a breakpoint, you can pause a run session at a predetermined place in a test or function library. A breakpoint is indicated by a filled red circle icon in the left margin adjacent to the selected step.

To set a breakpoint:

Perform one of the following:

- Click in the left margin of a step in the test or function library where you want the run to stop
- Click a step and then:



- Click the **Insert/Remove Breakpoint** button
- Choose **Debug > Insert/Remove Breakpoint**




The breakpoint symbol is displayed in the left margin of the test or function library.


Tip: You can also use the **Enable/Disable Breakpoint** option to add a breakpoint to a step. For more information, see “Enabling and Disabling Breakpoints” on page 598.

Enabling and Disabling Breakpoints

You can instruct QuickTest to ignore an existing breakpoint during a debug session by temporarily disabling the breakpoint. Then, when you run your test or function library, QuickTest runs the step containing the breakpoint, instead of stopping at it. When you enable the breakpoint again, QuickTest pauses there during the next run. This is particularly useful if your test or function library contains many steps, and you want to debug a specific part of it.

You can enable or disable breakpoints individually or all at once. For example, suppose you add breakpoints to various steps throughout your test or function library, but for now you want to debug only a specific part of your document. You could disable all breakpoints in your test or function library, and then enable breakpoints only for specific steps. After you finish debugging that section of your document, you could disable the enabled breakpoints, and then enable the next set of breakpoints (in the section you want to debug). Because the breakpoints are disabled and not removed, you can find and enable any breakpoint, as needed.

An enabled breakpoint is indicated by a filled red circle icon in the left margin  adjacent to the selected step.

A disabled breakpoint is indicated by an empty circle icon in the left margin  adjacent to the selected step.

To enable/disable a specific breakpoint:

- 1 Click in the line containing the breakpoint you want to disable/enable.
- 2 Choose **Debug > Enable/Disable Breakpoint** or press CTRL+F9. The breakpoint is either disabled or enabled (depending on its previous state).

To enable/disable all breakpoints:



Choose **Debug > Enable/Disable All Breakpoints** or click the **Enable/Disable All Breakpoints** button. If at least one breakpoint is enabled, QuickTest disables all breakpoints in the test or function library. Alternatively, if all breakpoints are disabled, QuickTest enables them.

Removing Breakpoints

You can remove a single breakpoint or all breakpoints defined for the current test or function library.

To remove a single breakpoint:

Perform one of the following:

- Click the breakpoint.
- Click the line in your test or function library with the breakpoint symbol and:



- Click the **Insert/Remove Breakpoint** button.
- Choose **Debug > Insert/Remove Breakpoint**.

The breakpoint symbol is removed from the left margin of the QuickTest window.

To remove all breakpoints:



Click the **Clear All Breakpoints** button, or choose **Debug > Clear All Breakpoints**. All breakpoint symbols are removed from the left margin of the QuickTest window.

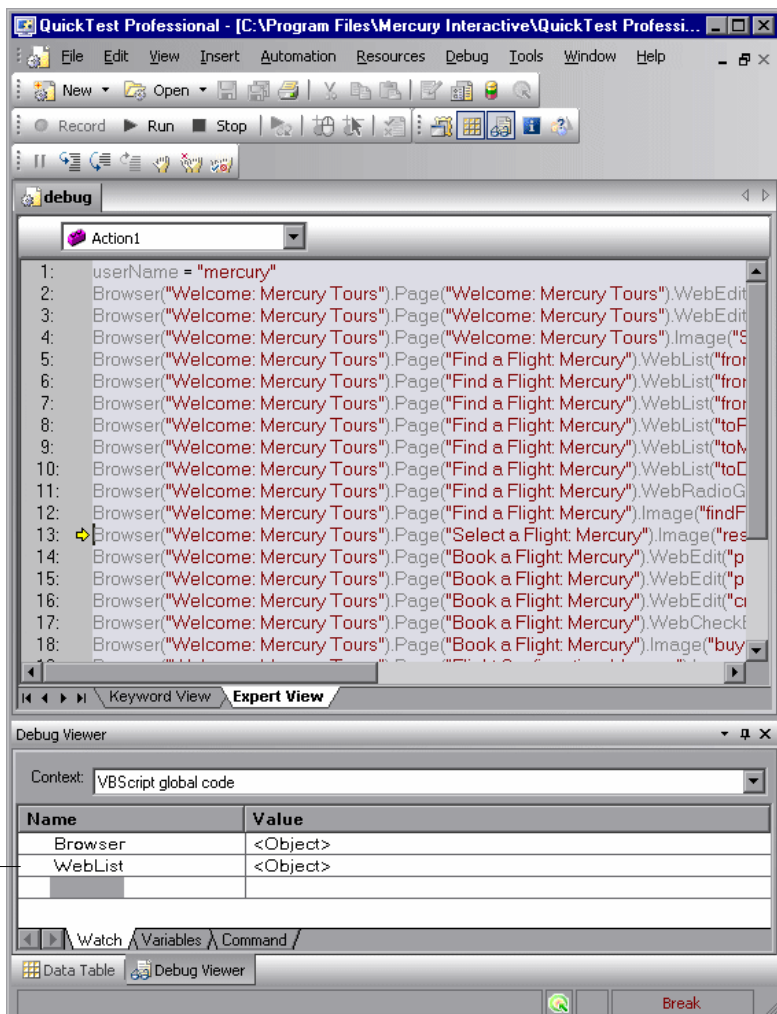
Using the Debug Viewer

You use the Debug Viewer pane to view, set, or modify the current value of objects or variables in your test or function library, when it stops at a breakpoint, or when a step fails and you select the **Debug** option.

To open the Debug Viewer pane:



Choose **View > Debug Viewer** or click the **Debug Viewer** button. The Debug Viewer pane opens.



Debug Viewer

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 Tab

You can view the current value of any variable or VBScript object in your test or function library by adding it to the Watch tab. As you continue stepping into the subsequent steps in your test or function library, QuickTest automatically updates the Watch tab with the current value for any object or variable whose value changes. You can also change the value of the variable manually when the test or function library pauses at a breakpoint.

To add an expression to the Watch tab:

Perform one of the following:

- Click the expression and choose **Debug > Add to Watch**.
- Click the expression and press CTRL+T.
- Right-click the expression and choose **Add to Watch** from the context menu.
- In the Watch tab, 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.

Note: You can add an expression to the Watch tab from the Expert View or from a function library.

Variables Tab

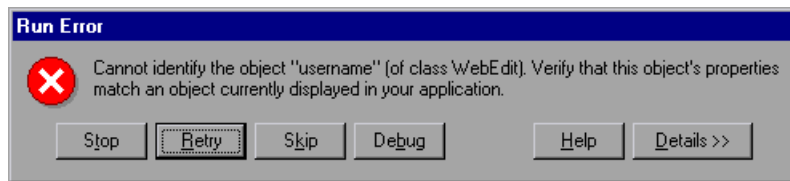
QuickTest automatically displays the current value of all variables in the current action or function in the Variables tab—up to the point where the test or function library is stopped or paused. For example, if you are stepping through a function, as you step into each step, QuickTest adds the current value for any step variable to the Variables tab grid. As you continue stepping into the subsequent steps, QuickTest automatically updates the value displayed in the Variables tab for any variable whose value changes. You can also change the value of the variable manually, during the breakpoint pause.

Command Tab

Use the Command tab to execute a line of script to set or modify the current value of a variable or VBScript object in your test or function library. 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 buttons for dealing with errors encountered:



- **Stop.** Stops the run session. The run results are displayed if QuickTest is configured to show run results after the run.
- **Retry.** QuickTest attempts to perform the step again. If the step succeeds, the run continues.
- **Skip.** QuickTest skips the step that caused the error, and continues the run from the next step.

- **Debug.** QuickTest suspends the run, enabling you to debug the test and any associated function library that contains a function called by the test.

You can perform any of the debugging operations described in this chapter. After debugging, you can continue the run session from the step where the test or function library stopped, or you can use the step commands to control the remainder of the run session.

- **Help.** Opens the QuickTest troubleshooting Help for the displayed error message. After you review the Help topic, you can select another button in the error message box.
- **Details.** Expands the message box to display additional information on the error.

Practicing Debugging an Action or a Function

Suppose you create an action or a function that defines variables that will be used in other parts of your test or function library. You can add breakpoints to the action or function to see how the value of the variables changes as the test or function library runs. To see how the test or function library handles the new value, you can also change the value of one of the variables during a breakpoint.

Step 1: Create a New Action or Function

Open a test and insert a new action, or open a new function library and create a new function called **SetVariables**. For more information on inserting actions, see Chapter 18, “Working with Actions.” For more information on working with functions, see Chapter 35, “Working with User-Defined Functions and Function Libraries.”

In the Expert View or function library, enter the VBScript code, as follows:

Expert View

```
Dim a
a="hello"
b="me"
MsgBox a
```

Function Library

```
Function SetVariables
Dim a
a="hello"
b="me"
MsgBox a
EndFunction
```

For more information on the Expert View, see Chapter 34, “Working in the Expert View and Function Library Windows.”

Note: If you are working in the Expert View, skip to Step 4. If you are working in a function library, continue with Step 2 and Step 3.

Step 2: (For Function Libraries Only) Associate the Function Library with a Test

- 1** Make sure the function library is in focus. (If it is not in focus, activate it by clicking the function library’s tab or choosing it from the **Window** menu.)
- 2** Choose **File > Associate Library '<Function Library Name>' with '<Test Name>'**, or right-click and choose **Associate Library '<Function Library Name>' with '<Test Name>'**. QuickTest associates the function library with your test.

Step 3: (For Function Libraries Only) Add a Call to the Function

Add a call to the function by inserting a new step and typing the following:

SetVariables

Step 4: Add Breakpoints

Add breakpoints at the lines containing the text `b="me"` and `MsgBox a`. For more information on adding breakpoints, see “Setting Breakpoints” on page 597.

Step 5: Begin Running the Test

Run the test. The test or function library stops at the first breakpoint, before executing that step (line of script).

Step 6: Check the Value of the Variables in the Debug Viewer Pane

- 1** Choose **View > Debug Viewer** to open the Debug Viewer pane, if it is not already open. Then select the **Watch** tab on the Debug Viewer pane.
- 2** In the document pane, select the variable **a** and choose **Debug > Add to Watch**. QuickTest adds the variable **a** to the Watch tab. The **Value** column indicates that the value of **a** is currently **hello**, because the breakpoint stopped after the value of variable **a** was initiated.
- 3** In the document pane, select the variable **b** and choose **Debug > Add to Watch**. QuickTest adds the variable **b** to the Watch tab. The **Value** column indicates **Variable is undefined: 'b'**, because the test stopped before variable **b** was declared.
- 4** Select the **Variables** tab in the Debug Viewer pane. If you are working with a test, only variable **a** is displayed (with the value **hello**), because **a** is the only variable that was initiated up to this point. If you are working with a function library, both **SetVariables** (with the value **Empty**) and variable **a** (with the value **hello**) are displayed. Variable **b** is not displayed because the test stopped before variable **b** was declared.

Step 7: Check the Value of the Variables at the Next Breakpoint

Click the **Run** button to continue running the test. The test stops at the next breakpoint. Note that the values of variables **a** and **b** have both been updated in the Watch and Variables tabs.

Step 8: 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 keyboard. Click the **Run** button to continue running the test. The message box that appears displays the new value of **a**.

23

Running Tests

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

This chapter describes:	On page:
About Running Tests	608
Running Your Entire Test	609
Running Part of Your Test	613
Updating a Test	616
Using Optional Steps	625
Running a Test Batch	627

About Running Tests

When you run a test, QuickTest performs the steps it contains. If you have defined test parameters, QuickTest prompts you to enter values for them. When the run session is complete, QuickTest displays a report detailing the results. For more information on viewing the results, see Chapter 24, “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 more information, see Chapter 26, “Setting Options for Individual Tests,” and Chapter 18, “Working with Actions.”

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

You can run tests on objects with dynamic descriptions. For more information, see Chapter 6, “Working with Test Objects.”

You can set up a batch of tests and run them sequentially, using the QuickTest Test Batch Runner. For more information, see “Running a Test Batch” on page 627.

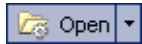
Note for WinRunner users: You can run WinRunner tests and call functions from WinRunner-compiled modules while running a QuickTest test. For information, see Chapter 44, “Working with WinRunner.”

Running Your Entire Test

QuickTest always runs a test from the first step, unless you specify otherwise. To run a test from or to a selected step or action, you can use the **Run from Step** or **Run to Step** options. These features are useful if you want to check a specific section of the test, without running the test from the beginning or to the end. For more information, see “Running Part of Your Test” on page 613.

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

To run a test:

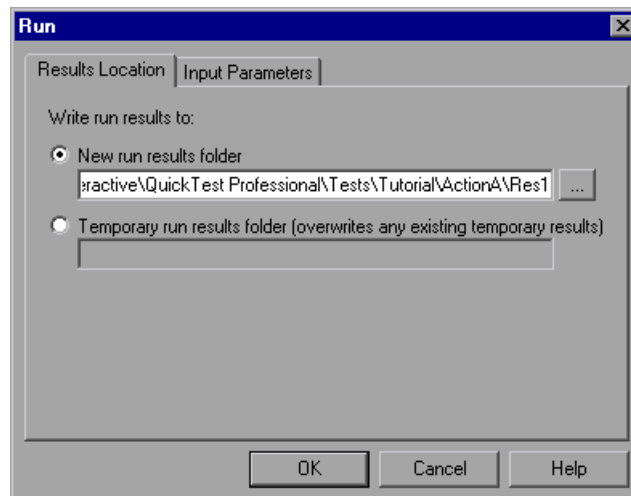


- 1 If your test is not already open, choose **File > Open > Test** or click the **Open** button to open it.

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



- 2 Click the **Run** button on the toolbar, or choose **Automation > Run**. The Run dialog box opens.



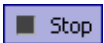
- 3 Specify the results location and the input parameter values (if applicable) for the run session. For more information, see “Understanding the Results Location Tab” on page 611, and “Understanding the Input Parameters Tab” on page 612.
- 4 Click **OK**. The Run dialog box closes and the run session starts. By default, when the run session ends, the Test Results window opens. For more information on viewing the run session results, see Chapter 24, “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 25, “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 **Automation > Run**.

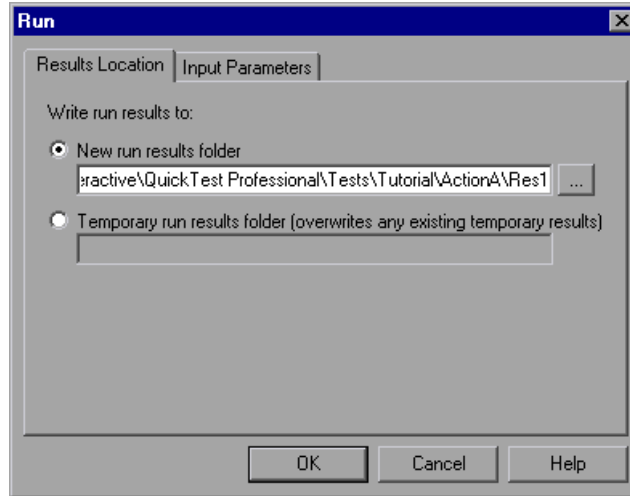


Click the **Stop** button or choose **Automation > Stop**. The run session stops and the Test Results window opens.

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

Understanding the Results Location Tab

The Results Location tab enables you to specify the location in which you want to save the run session results.



Select one of the following options:

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

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

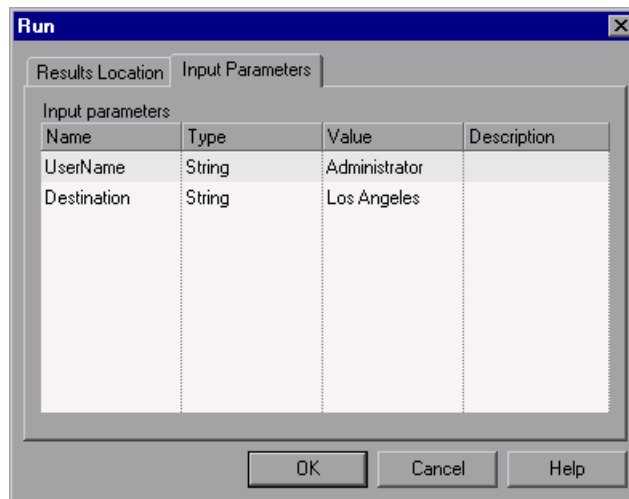
Note: If you are running a test from a Quality Center project, the **Project name**, **Run name**, **Test set**, and **Instance** options are displayed instead of the **New run results folder** option. For more information, see “Running a Test Stored in a Quality Center Project from QuickTest” on page 1285.

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

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

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 run session.



The Input Parameters tab displays the input parameters that were defined for the test (using the **File > 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 list. If you do not enter a value, QuickTest uses the default value from the Test Settings dialog box during the run session.

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

For more information on setting test parameters, see “Defining Parameters for Your Test” on page 771. For more information on using parameters, see Chapter 16, “Parameterizing Values”.

Running Part of Your Test

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

Note: You can also use the **Debug > Run to Step** option if you want to run a test in debug mode from the start of the test to a selected step. For more information, see “Using the Run to Step and Start from Step Commands” on page 593.

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.

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

Tips:

If you only want to run one iteration of your test, 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 597.

For more information on actions, see Chapter 18, “Working with Actions.”

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

- 1** Open your application to the location matching the action or step you want to run.
- 2** Select the action or step where you want to start running the test:
 - In the Keyword View, highlight a step or action row.
 - In the Expert View, place your cursor in a line of VBScript.

Make sure that the step or action you choose is not dependent on previous steps.

- 3** Choose **Automation > Run from Step** or **Run Current Action**, or right-click and choose **Run from Step**.
- 4** In the Run dialog box, choose where to save the run session results, and any input parameters you want to use, as described in “Understanding the Results Location Tab” on page 611, and “Understanding the Input Parameters Tab” on page 612.

Note: When running part of a test within the scope of an action, you need to specify the action’s parameters, not the test parameters, in the Input Parameters tab of the Run dialog box.

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

By default, when the run session ends, the Test Results window opens. For more information on viewing the run session results, see Chapter 24, “Analyzing Test Results.”

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

Note: If you cleared the **View results when run session ends** check box in the Run 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 25, “Setting Global Testing Options.”

Updating a Test

When you update a test, QuickTest runs the test to update the test object descriptions, the expected checkpoint values, and/or the Active Screen images and values. You can choose to update the data for an entire test or only part of it, according to the run and debug run options you select. After you save the test, the updated data is used for subsequent runs.

When QuickTest updates tests, it runs through only one iteration of the test and one iteration of each action in the test, according to the run option selected. For information on actions, see Chapter 18, “Working with Actions.”

Note: When QuickTest updates a test, it does not update parameterized values, such as Data Table data and environment variables. For information on parameterized values and environment variables, see Chapter 16, “Parameterizing Values.”

When QuickTest updates tests, it always saves the updated objects in the local object repository, even if the objects being updated were originally from a shared object repository. The next time you run the test, QuickTest uses the objects from the local object repository, as the local object repository has a higher priority than any shared object repositories.

Tip: After using **Update Run Mode** to update the test, you may want to use the **Update from Local Repository** option in the Object Repository Manager to merge the objects from the local repository back to a shared object repository. For more information, see Chapter 38, “Managing Object Repositories.”

To update a test:

- 1** If your test is not already open, choose **File > Open > Test** or click the **Open** arrow and choose **Test**.
- 2** Select a test and click **Open**. The test opens and the title bar displays the test name.

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



- 3** Choose **Automation > Update Run Mode**, or click the **Update Run Mode** button. The **Update Run Mode** menu option and button are toggled, and your test will be updated according to the type of run you select in the following step.
- 4** Select the type of run to perform from the following options:



- **Automation > Run.** Runs and updates your entire test. You can also click the **Run** button.
- **Automation > Run Current Action.** Runs and updates the currently selected action only.
- **Automation > Run from Step.** Runs and updates your test from the selected step to the end of the test. (In the Expert View, it runs to the end of the current action. In the Keyword View, it runs to the end of the test.)

You can also select one of the following debug run modes to debug and update your test:



- **Debug > Step Into** or click the **Step Into** button. Debugs and updates only the current step of the test.



- **Debug > Step Over** or click the **Step Over** button. Debugs and updates only the current step of the test.

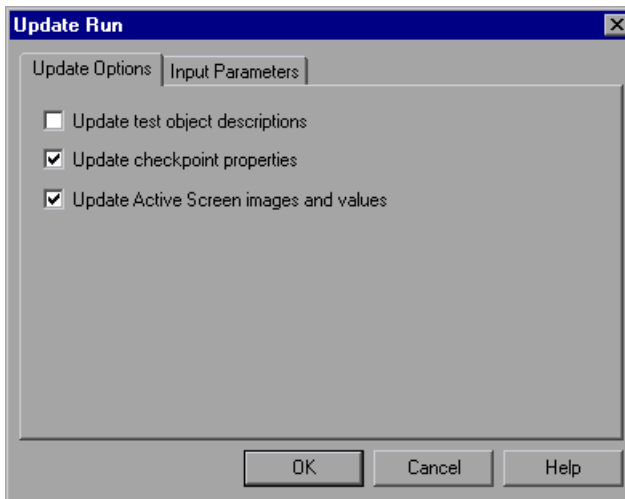


- **Debug > Step Out** or click the **Step Out** button. Debugs and updates the current step of the test.

- **Debug > Run to Step.** Debugs and updates the action (Expert View only) or test, from the current location in the test or action until it reaches the selected step.
- **Debug > Start from Step.** Debugs and updates the test or action from the selected step to the end of the test. (In the Expert View, it runs to the end of the current action. In the Keyword View, it runs to the end of the test.)

For more information on debugging, see Chapter 22, “Debugging Tests and Function Libraries.”

The Update Run dialog box opens.



- 5 Specify the settings for the update run process. For more information, see “Understanding the Update Options Tab” on page 621, and “Understanding the Input Parameters Tab” on page 612.

Note: The run results for an update run session are always saved in a temporary location.

- 6 Click **OK**. The Update Run dialog box closes and QuickTest begins running the test update. The text **Update Run** flashes in the status bar while the test is being updated.

QuickTest runs the test and updates the test object descriptions, the Active Screen information, and/or the expected checkpoint values, depending on your selections. When the run session ends, the Test Results window opens.

For information on viewing the results, see Chapter 24, “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 update run session. For more information on the Options dialog box, see Chapter 25, “Setting Global Testing Options.”



Tip: Remember to exit the update run mode by clicking the **Update Run Mode** button when you have finished updating your test.

When the update run ends, the Test Results window can show:

- updated values for checkpoints.
- updated test object descriptions.

For example:

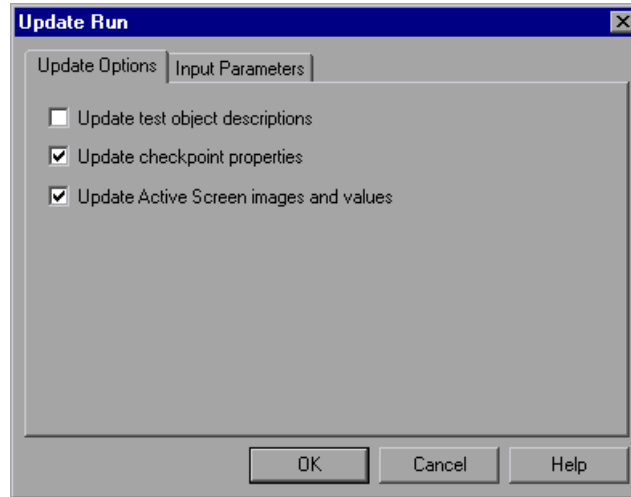
**Step Name: Notifications:-Update
Description**

Step Done

Object	Details	Result	Time
Notifications: - Update Description	Test object's previous description: Text = Selection = Native Class = ListBox Test object's new description: Attached Text = Notifications:	Done	5/20/2005 - 10:43:11

Understanding the Update Options Tab

The Update Options tab enables you to specify which aspects of your test you want to update, such as test object descriptions, expected checkpoint values, and/or Active Screen images and values. After you save the test, the results of the updated test are used for subsequent runs.



You can specify one or more of the following information types to update:

- **Update test object descriptions.** QuickTest updates the 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. When you use this option, all values are updated, even if they are parameterized or use regular expressions.

Tip: You can also update individual test object descriptions from the object in your application using the **Update from Application** option in the Object Repository window or Object Repository Manager. For more information, see “Updating Test Object Properties from an Object in Your Application” on page 172.

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 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 using property values that are easy to recognize in your application (such as object labels), but may be language or operating system dependent. After you debug your test, you can use the **Update Run Mode** option to change the object descriptions to use more universal property values.

For example, suppose you design a test for the English version of your application. The test objects are recognized according to the 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 properties that are non-language dependent. These properties will be used for object identification. For example, you can identify a link object by its **target** property value instead of its **text** property value. You can then perform an update run on the English version of your application using these new properties. This will 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 step based on its current test object description. If QuickTest cannot find the test object based on its description, it uses the Smart Identification properties to identify the test object (if Smart Identification is enabled). After QuickTest finds the test object, it then updates its description based on the mandatory and assistive properties that you define in the Object Identification dialog box.

Note: Test objects that cannot be identified during the update process are not updated. As in any run session, if an object cannot be found during the update run, the run session fails, and information on the failure is included in the Test Results.

Any properties that were used in the previous test object description and are no longer part of the description for that test object class, as defined in the Object Identification dialog box, are removed from the new description, even if the values were parameterized or defined as regular expressions.

If the same property appears both in the test object's new and previous descriptions, one of the following occurs:

- If the property value in the previous description is parameterized or specified as a regular expression and matches the new property value, QuickTest keeps the property's previous parameterized or regular expression value. For example, if the previous property value was defined as the regular expression **button.***, and the new value is **button1**, the property value remains **button.***.
- If the property value in the previous description does not match the new property value, but the object is found using Smart Identification, QuickTest updates the property value to the new, constant property value. For example, if the previous property value was **button.***, and the new value is **My button**, if a Smart Identification definition enabled QuickTest to find the object, **My button** becomes the new property value. In this case, any parameterization or use of regular expressions is removed from the test object description.

- **Update checkpoint properties.** QuickTest updates the expected checkpoint values to reflect any changes that may have occurred in your application since you recorded the test. 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 checkpoint property values are parameterized or include regular expressions, they are not updated when using this option.

If you selected the **Save only selected area** check box when creating a bitmap checkpoint, the **Update Run Mode** 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 285.

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 1250. For more information on comment lines, see “Adding Comments” on page 578.

- **Update Active Screen images and values.** QuickTest updates images and property values in the Active Screen to reflect any changes that may have occurred in your application since you recorded the test or if the Active Screen does not appear as it should. For example, suppose a dialog box in your application has changed since you recorded your test. You can update the test to update the dialog box appearance and its properties in the Active Screen.

Using Optional Steps

An optional step is a step that is not necessarily required to successfully complete a run session. For example, suppose that when recording a test, the application you are testing prompts you to enter a user name and password in a login window. When you run the test, however, the application does not prompt you to enter your user name and password because it retained the information that was previously entered. In this case, the steps that were recorded for entering the login information are not required and should, therefore, be marked optional.


During a run session, 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 run to fail.


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

By default, QuickTest automatically marks steps that open certain dialog boxes as optional. 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 32, “Defining and Using Recovery Scenarios.”

Setting Optional Steps

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.

Welcome: Mercury Tours			
userName	Set	"Amy"	Enter "Amy" in the "userName" edit box.
password	SetSecure	"425e5cd870...	Enter the encrypted string "425e5cd87021ce00d5476d94a8e4420...
Sign-In	Click	10,11	Click the "Sign-In" image.
AutoComplete			
 Yes	Click		Click the "Yes" button.
Sign-on: Mercury Tours	Sync		Wait for the Web page to synchronize before continuing the run.

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

For information on working in the Expert View, see Chapter 34, “Working in the Expert View and Function Library Windows.” For information on the **OptionalStep** element, refer to the *QuickTest Professional Object Model Reference*.

Default Optional Steps

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

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

Running a Test Batch

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

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

Notes:

To enable Test Batch Runner to run tests, you must select **Allow other Mercury products to run tests and components** in the Run tab of the Options dialog box. For more information, see Chapter 25, “Setting Global Testing Options.”

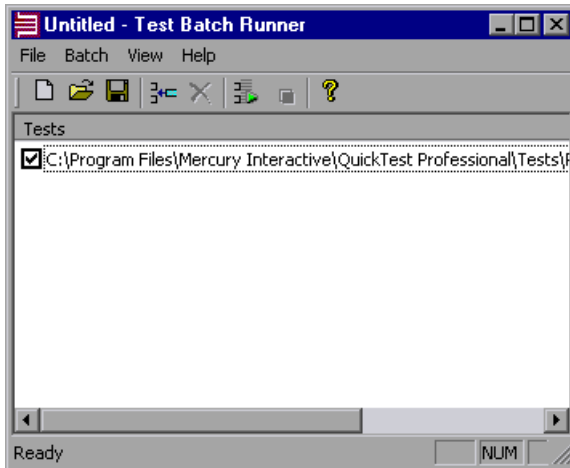
Test Batch Runner can be used only with tests located in the file system. If you want to include tests saved in Quality Center in the batch run, you must first save the tests in the file system.



You can stop a test batch run at any time by clicking the **Stop** button.

To set up and run a test batch:

- 1 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 on Test Results, see Chapter 24, “Analyzing Test Results.”

24

Analyzing Test Results

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

This chapter describes:	On page:
About Analyzing Test Results	632
Understanding the Test Results Window	634
Viewing the Results of a Run Session	638
Viewing Checkpoint Results	653
Viewing Parameterized Values and Output Value Results	675
Analyzing Smart Identification Information in the Test Results	685
Deleting Test Run Results	689
Submitting Defects Detected During a Run Session	697
Viewing WinRunner Test Steps in the Test Results	699
Customizing the Test Results Display	702

About Analyzing Test Results

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

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

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

Note: You set the test to run for one or all iterations in the Run tab of the Test Settings dialog box. For more information, see “Defining Run Settings for Your Test” on page 763.

After you run a test, 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 where failures occurred
- application snapshots that highlight any discrepancies, for each stage of the test

Note: By default, QuickTest saves screen captures only on errors. You can instruct QuickTest to save screen captures in other cases, as well, by selecting **Always** or **On errors and warnings** in the **Save step screen capture to results** in the Run tab of the Options dialog box. For more information, see “Setting Run Testing Options” on page 723.

- detailed explanations of each step and checkpoint pass or failure, at each stage of the test

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

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 “Setting Run Testing Options” on page 723.

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:

Test results title bar

Menu bar

Test results toolbar

Test results tree

Test results details for selected step in test

Status bar

MercuryTours Results Summary

Test: MercuryTours
Results name: TempResults
Time Zone: Eastern Standard Time
Run started: 12/19/2005 - 10:19:33
Run ended: 12/19/2005 - 10:22:09

Iteration #	Results
1	Passed
2	Failed
3	Failed

Status	Times
Passed	7
Failed	5
Warnings	0

For Help, press F1

Ready

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 the test's actions.





The Test Results window contains the following key elements:







- **Test results title bar.** Displays the name of the test.
- **Menu bar.** Displays menus of available commands.
- **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 637.
- **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 636.
- **Status bar.** Displays the status of the currently selected command (choose **View > Status Bar** to view the status bar).

You can change the appearance of the Test Results window. For more information, see “Changing the Appearance of the Test Results Window” on page 638.

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. Note that if a test does not contain checkpoints, no icon is displayed.
-  indicates a step that failed. Note that this causes all parent steps (up to the root action or test) to fail as well.
-  indicates a warning, meaning that the step did not succeed, but it did not cause the action or test to fail.
-  indicates a step that failed unexpectedly, such as when an object is not found for a checkpoint.

-  indicates an optional step that failed and therefore was ignored. Note that this does not cause the test to fail.
-  indicates that the Smart Identification mechanism successfully found the object.
-  indicates that a recovery scenario was activated.
-  indicates that the run session was stopped before it ended.
-  `[password].SetSecure` square brackets around a test object name indicate that the test object was created dynamically during the run session. A dynamic test object is created either using programmatic descriptions or by using an object returned by a ChildObjects method, and is not saved in the object repository.
-  Displays the **Run-Time Data Table**, 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 to change the level of detail that the tree displays.

Test Results Details

By default, when the Test Results window opens, a test summary is displayed in the right pane of the window. When you select a branch or step in the tree, the right pane displays detailed information for the selected item.

The Test Results details area indicates the test name, results name, the start and end date and time of the run, the number of iterations, and whether an iteration passed or failed. If an iteration contains checkpoints, the possible results are **Passed** or **Failed**. If an iteration does not contain checkpoints, the possible results are **Done** or **Failed**.

If the test was run from Quality Center, the name of the test set and the test instance are also shown.

Test1_RO Results Summary

Test: Test1_RO

Results name: Run_1-29_10-52-27

Time Zone: Eastern Standard Time

Run started: 1/29/2006 - 9:55:01

Run ended: 1/29/2006 - 9:55:12

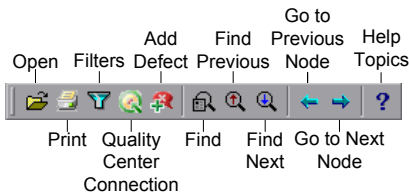
Test set: Root\temp\gabby\TestSet1

Test instance: 1

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.

Test Results Toolbar

The Test Results toolbar contains buttons for viewing test results.



Changing the Appearance of the Test Results Window

By default, the Test Results window has the same look and feel as the QuickTest window, using the Microsoft Office 2003 theme. You can change the look and feel of the Test Results window, as required.

To change the appearance of the Test Results window:

In the Tests Results window, choose **View > Window Theme**, and then select the way the window should appear from the list of available themes. For example, you can apply a Microsoft Office 2000 or Microsoft Windows XP theme.

Note: You can apply the Microsoft Windows XP theme to the Tests Results window only if your computer is set to use a Windows XP theme.

Tip: You can also change the theme used for the main QuickTest window. For more information, see “Changing the Appearance of the QuickTest Window” on page 23.

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 “Setting Run Testing Options” on page 723.)

In addition, you can view the results of previous runs of the current test, and results of other tests. You can also preview test results on screen and print them to your default Windows printer, as well as export them to an HTML file.

To view the results of a run:

- 1** If the Test Results window is not already open, click the **Results** button or choose **Automation > 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, they are displayed in the Test Results window. For information on the Test Results window, see “Understanding the Test Results Window” on page 634.
- If there are several test results for the current test, or if there are no test results for the current test, the Open Test Results dialog box opens. You can select the test results for any test, 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 more information on viewing test results, see “Opening Test Results to View a Selected Run” on page 644.

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 window, 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 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, 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:

- **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 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.
- **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 more information, see “Opening Test Results to View a Selected Run” on page 644.



- 8** To print test results, click the **Print** button or choose **File > Print**. For more information, see “Printing Test Results” on page 648. (You can preview the run results before you print them. For more information, see “Previewing Test Results” on page 649.)

Note: If you have Quality Center installed, you can add a defect to a Quality Center project. For more information, see “Submitting Defects Detected During a Run Session” on page 697.

- 9** To export the run results to an HTML file, choose **File > Export to HTML File**. For more information, see “Exporting Test Results” on page 651.
- 10** Choose **File > Exit** to close the Test Results window.

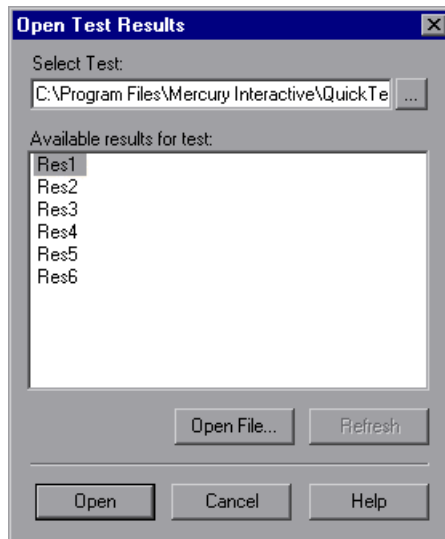
Note: You can use **Reporter.Filter** statements in the Expert View to disable or enable the saving of selected steps, or to save only steps with **Failed** or **Warning** status. For more information on saving run session information, see “Choosing Which Steps to Report During the Run Session” on page 1035, 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 you can view the saved results for other tests.

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 **Automation > Results**, when there are several results, or no results, for the current test.



The results of run sessions for the current test 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 path, click **Refresh**.

To view results of runs for other tests, you can search in your file system by test, or by test result file. If you are connected to Quality Center, you can also search in Quality Center 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. When you run your test, you can specify a different location to store the results, using the Results Location tab of the Run dialog box. Specifying your own location for the results file can make it easier for you to locate the results file in the file system. For more information, see “Understanding the Results Location Tab” on page 611.

You can search for results in the file system by test or by result file.

To search for results in the file system by test:

- 1 In the Open Test Results dialog box, enter the path of the folder that contains the results file for your test, or click the browse button to open the Open Test dialog box.
- 2 Find and highlight the test 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 File** button to open the Select Results File dialog box.
- 2 Browse to the folder where the test results file is stored.

- 3 Highlight the 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 Saved in Quality Center

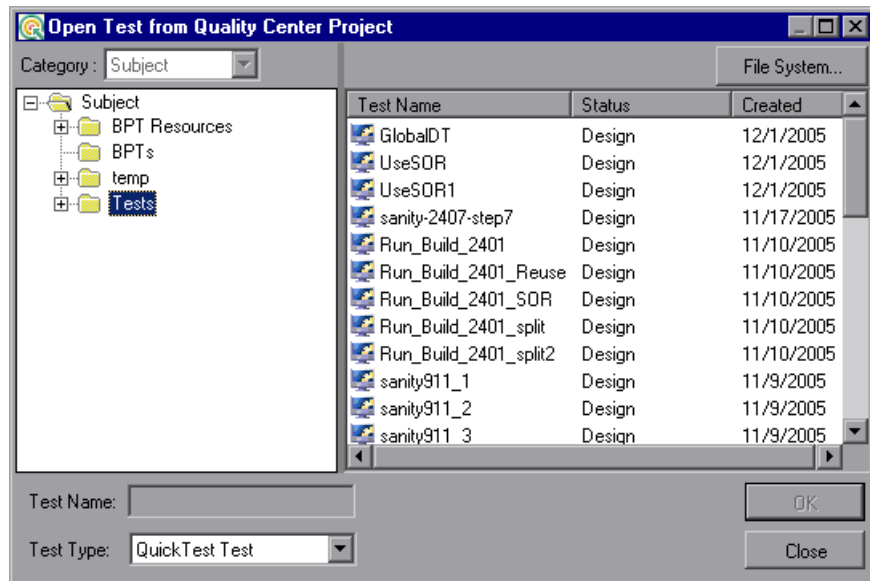
If your QuickTest test is stored in Quality Center, the results are stored in the test folder in Quality Center. You cannot change the location of the test results.

To search for test results saved in Quality Center:



- 1 In the Test Results window, choose **Tools > Quality Center Connection** or click the **Quality Center Connection** button and connect to your Quality Center project.

- 2 In the Open Test Results dialog box, enter the path of the folder that contains the results file for your QuickTest test, or click the browse button to open the Open Test from Quality Center Project dialog box.



- 3 Select **QuickTest Test** in the **Test Type** list.
- 4 Find and highlight the test whose test results you want to view, and click **OK**.
- 5 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.

Viewing Results of Tests Run From Quality Center

When you run test sets containing QuickTest tests from Quality Center, the Quality Center server opens QuickTest on the host computer and runs the tests from that computer. All run results are then saved to the default location for those tests.

You can view the results of QuickTest tests run from Quality Center. These run results contain the same information described in “Understanding the Test Results Window” on page 634 plus the following additional fields:

Test set. Specifies the location of the test set.

Test instance. Specifies the instance number of the test in the test set. For example, if the same test is included twice in the test set, you can view the results of Test instance 1 and Test instance 2.

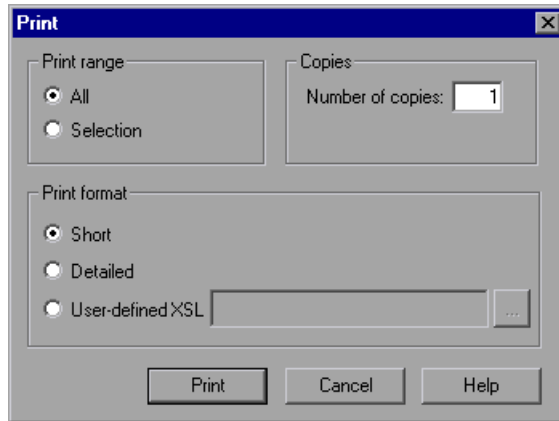
Printing Test Results

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.
 - **Selection.** Prints the test results 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, or for the selected branch, according to your selection in step 2.
- **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 702.

Note: The **Print format** options are available only for test results created with QuickTest version 8.0 and later.

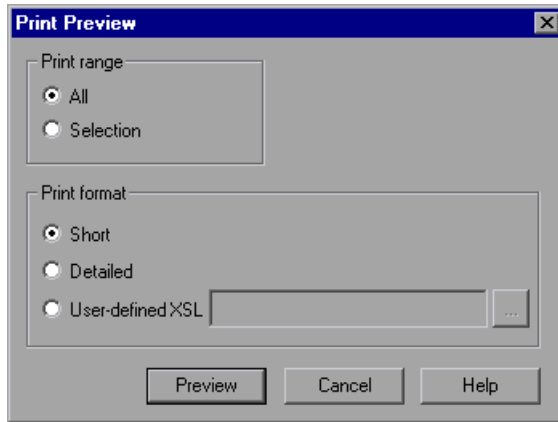
5 Click **Print** to print the selected test results information to your default Windows printer.**Previewing Test Results**

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.
- **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, or for the selected branch, according to your selection in step 2.
- **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 702.

- 4 Click **Preview** to preview the appearance of your test results on screen.



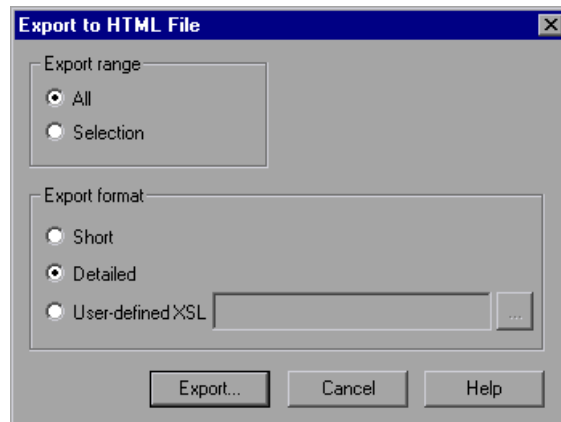
Tip: If some of the information is cut off in the preview, for example, if checkpoint names are too long to fit in the display, click the **Page Setup** button in the Print Preview window and change the page orientation from **Portrait** to **Landscape**.

Exporting Test Results

You can export the test results from the Test Results window to an HTML file. This enables you to easily view the test results when you are not in a QuickTest environment. For example, you can send the HTML file containing the test results in an e-mail to a third-party who does not have QuickTest installed. You can select the type of report you want to export, and you can also create and export a customized report.

To export the test results:

- 1 Choose **File > Export to HTML File**. The Export to HTML File dialog box opens.



2 Select an **Export range** option:

- **All.** Exports the results for the entire test.
- **Selection.** Exports test result information for the selected branch in the test results tree.

3 Select an **Export format** option:

- **Short.** Exports 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.** Exports all available information for each item in the test results tree, or for the selected branch, according to your selection in step 2.
- **User-defined XSL.** Enables you to browse to and select a customized **.xsl** file. You can create a customized **.xsl** file that specifies the information to be included in the exported report, and the way it should appear. For more information, see “Customizing the Test Results Display” on page 702.

Note: The **Export format** options are available only for test results created with QuickTest version 8.0 and later.


- 4** Click **Export**. The Save As dialog box opens, enabling you to change the default destination folder and rename the file, if required. By default, the file is named <name of test> [<name of run results>], and is saved in the test results folder.
- 5** Click **Save** to save the HTML file and close the dialog box.

Viewing Checkpoint Results

By adding checkpoints to your tests, you can compare expected values in, for example, Web pages, text strings, object properties, and tables to the values of these elements in your application. This enables you to ensure that your application functions as desired.

When you run the test, QuickTest compares the expected results of the checkpoint to the current results. If the results do not match, the checkpoint fails, which causes the test to fail. You can view the results of the checkpoint in the Test Results window.

To view the results of a checkpoint:

- 1 Display the test results for your test in the Test Results window. For more information, see “Viewing the Results of a Run Session” on page 638.
-  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 on the selected checkpoint only if it has the status **Failed**. You can change the conditions for when a step’s image is saved, in the Run tab of the Options dialog box. For more information, see “Setting Run Testing Options” on page 723.

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,” below
- “Analyzing Table and Database Checkpoint Results” on page 656
- “Analyzing Bitmap Checkpoint Results” on page 658
- “Analyzing Text Checkpoint Results” on page 660
- “Analyzing XML Checkpoint Results” on page 661
- “Analyzing Accessibility Checkpoint Results” on page 670

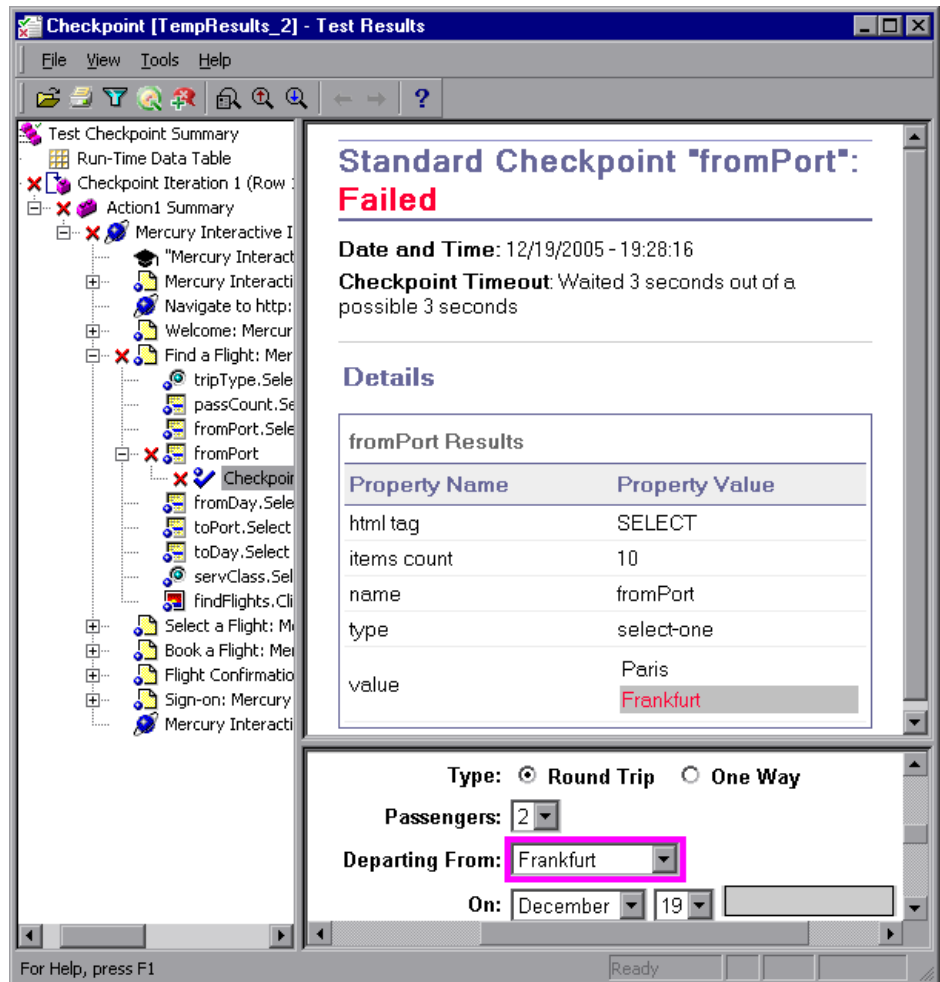
3 Choose **File > Exit** to close the Test Results window.

For more information on checkpoints, see Chapter 8, “Understanding Checkpoints.”

Analyzing Standard Checkpoint Results

By adding standard checkpoints to your tests, you can compare the expected values of object properties to the object’s current values during a run session. If the results do not match, the checkpoint fails. For more information on standard checkpoints, see “Checking Object Property Values” on page 273.

You can view detailed results of the standard checkpoint in the Test Results window. For information on displaying the results for a checkpoint, see “Viewing Checkpoint Results” on page 653.



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 **Paris**, but the actual value is **Frankfurt**.

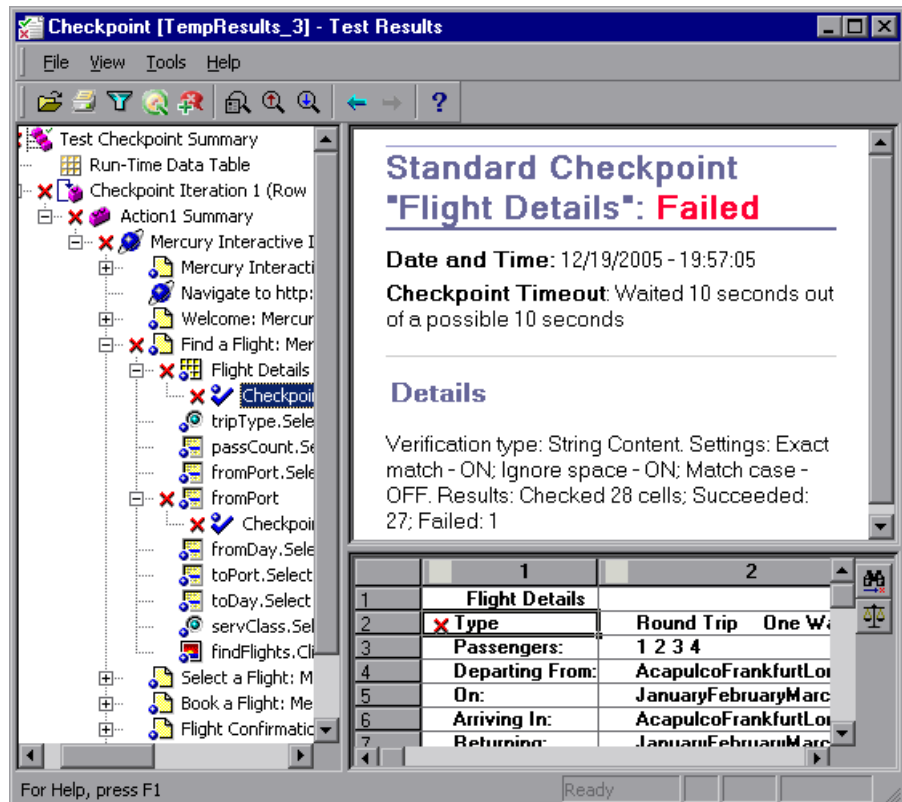
Analyzing Table and Database Checkpoint Results

By adding table checkpoints to your tests, you can check that a specified value is displayed in a cell in a table on your application. By adding database checkpoints to your tests, you can check the contents of databases accessed by your application.


The results displayed for table and database checkpoints are similar. When you run your test, QuickTest compares the expected results of the checkpoint to the actual results of the run session. If the results do not match, the checkpoint fails.

For more information on table and database checkpoints, see Chapter 9, “Checking Tables” and Chapter 13, “Checking Databases.”

You can view detailed results of the table or database checkpoint in the Test Results window. For information on displaying the results for a checkpoint, see “Viewing Checkpoint Results” on page 653.



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  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.



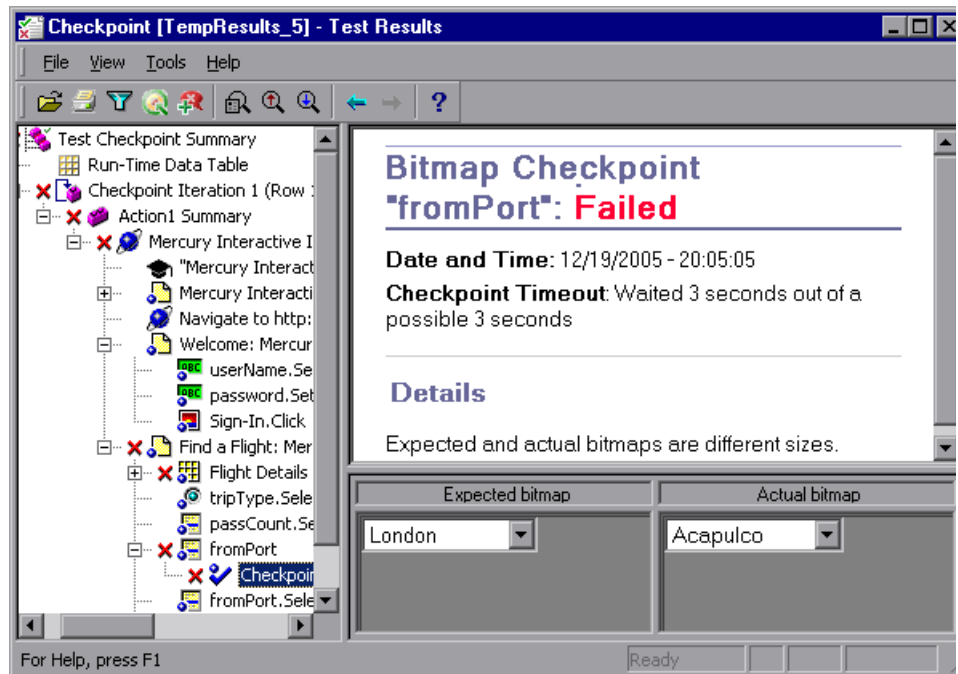
You can click the **Compare Values** button in the bottom right pane to display the expected and actual values of the selected table cell or database record.

Analyzing Bitmap Checkpoint Results

By adding bitmap checkpoints to your tests, you can check the appearance of elements in your application by matching captured bitmaps. When you run your test, QuickTest compares the expected results of the checkpoint to the actual results of the run session. If the results do not match, the checkpoint fails.

For more information on bitmap checkpoints, see Chapter 12, “Checking Bitmaps.”

You can view detailed results of the bitmap checkpoint in the Test Results window. For information on displaying the results for a checkpoint, see “Viewing Checkpoint Results” on page 653.



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).

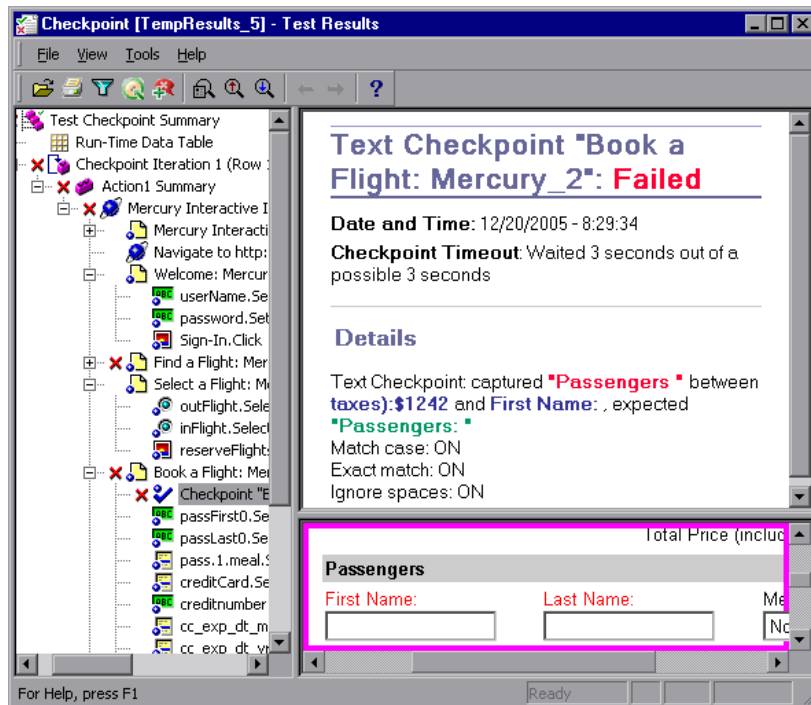
The bottom right pane shows the expected and actual bitmaps that were compared during the run session.

Analyzing Text Checkpoint Results

By adding text checkpoints to your tests, you can check that a text string is displayed in the appropriate place in your application. When you run your test, QuickTest compares the expected results of the checkpoint to the actual results of the run session. If the results do not match, the checkpoint fails.

For more information on text checkpoints, see Chapter 10, “Checking Text.”

You can view detailed results of the text checkpoint in the Test Results window. For information on displaying the results for a checkpoint, see “Viewing Checkpoint Results” on page 653.

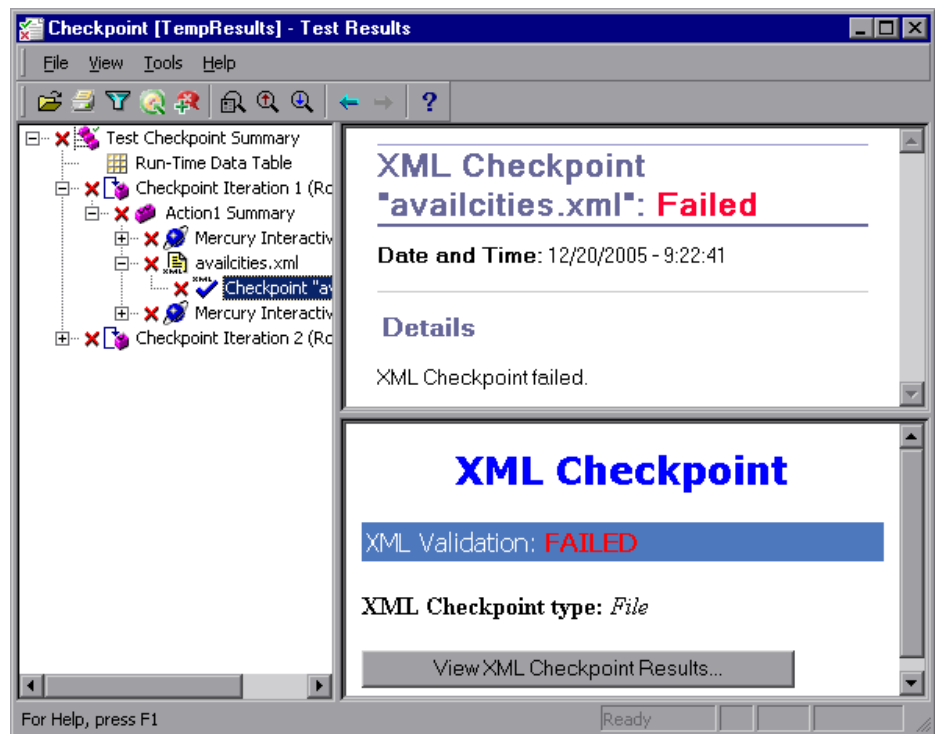


The top right pane displays the checkpoint step results, including its status (**Passed** or **Failed**), the date and time the checkpoint was run and the portion of the checkpoint timeout interval that was used (if any). It also shows the expected text and actual text that was checked, and the verification settings you specified for the checkpoint.

Analyzing XML Checkpoint Results

By adding XML checkpoints to your tests, you can verify that the data and structure in your XML documents or files has not changed unexpectedly. When you run your test, QuickTest compares the expected results of the checkpoint to the actual results of the run session. If the results do not match, the checkpoint fails. For more information on XML checkpoints, see Chapter 14, “Checking XML.”

You can view summary results of the XML checkpoint in the Test Results window. For information on displaying the results for a checkpoint, see “Viewing Checkpoint Results” on page 653.



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

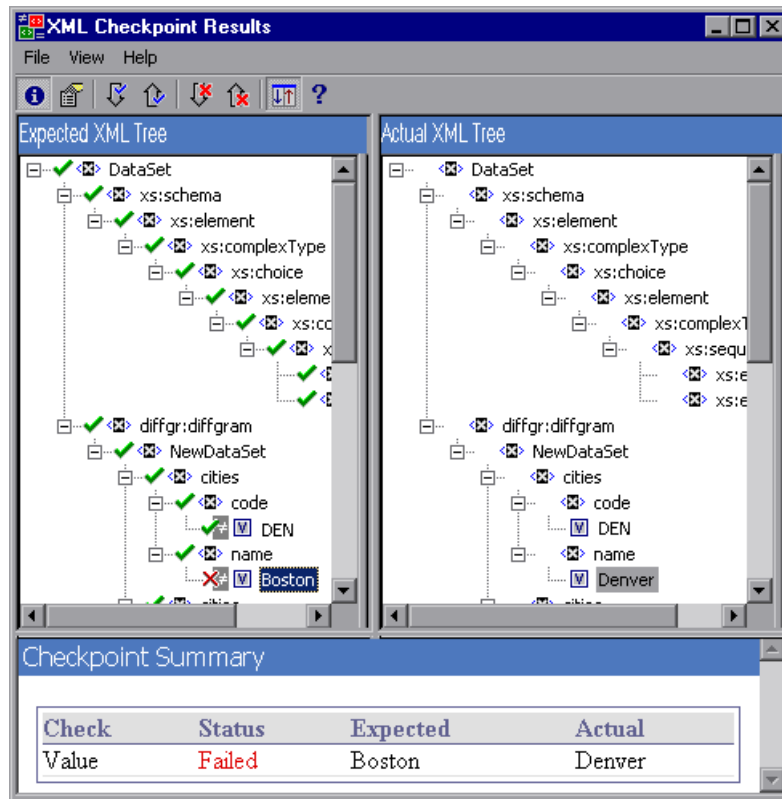
When you click the **View XML Checkpoint Results** button from the Test Results window, the XML Checkpoint Results window displays the XML file hierarchy.

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.



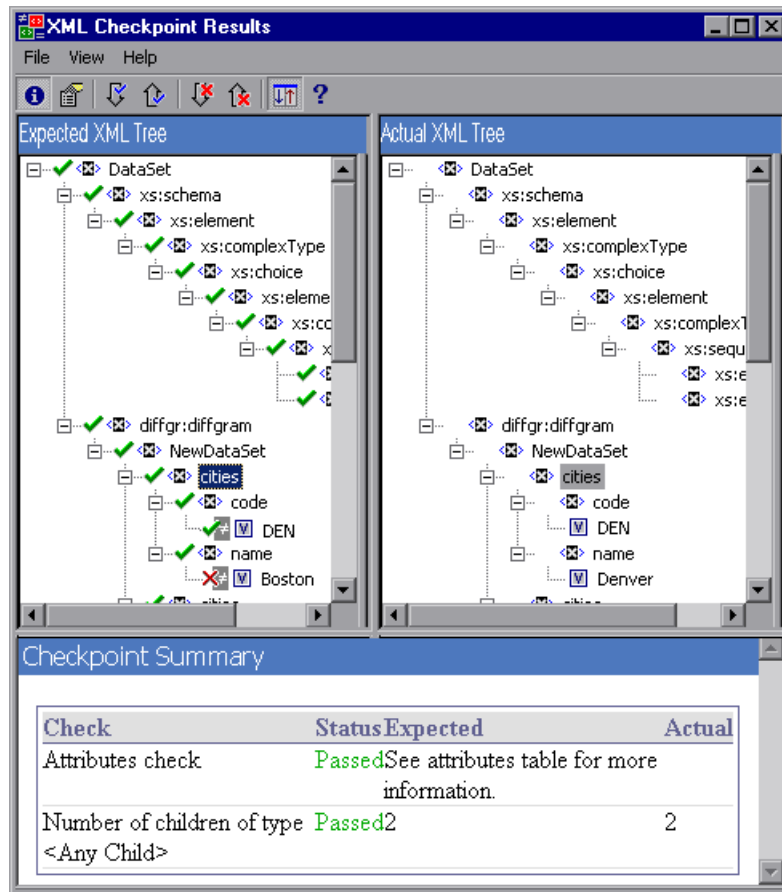
Navigating the XML Checkpoint Results Window

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.

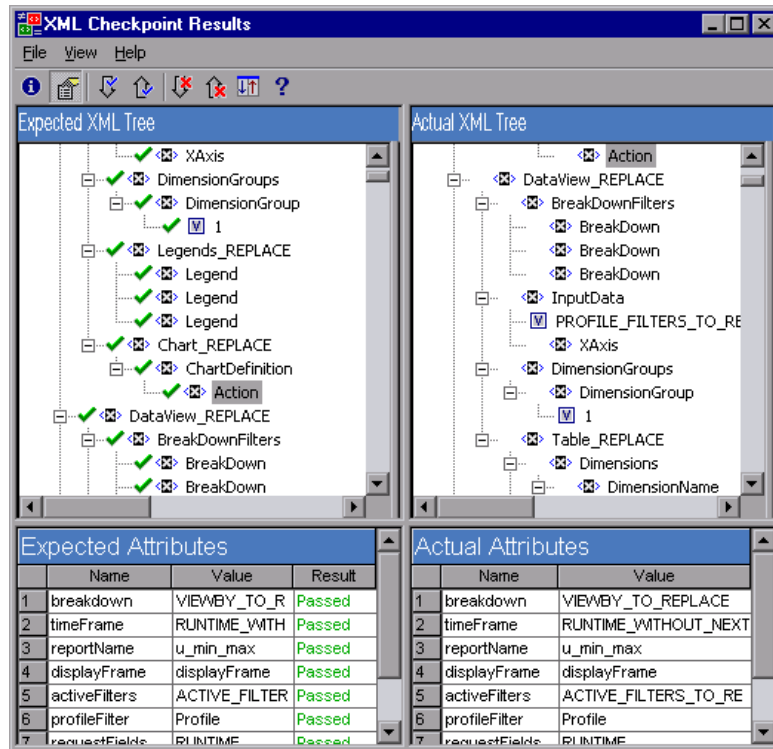
The following example displays the Checkpoint Summary for the cities element in an XML file.



- **View Attribute Details.** In the XML Tree, select an element whose attributes were checked. Click the **View Attribute Details** button or 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.
- **Find Next Error.** Choose **View > Find Next Error** or click the **Find Next Error** button to jump directly to the next error in the XML Tree.
- **Find Previous Error.** Choose **View > Find Previous Error** or click the **Find Previous Error** button to jump directly to the previous error in the XML Tree.



- **Scroll Trees Simultaneously.** 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

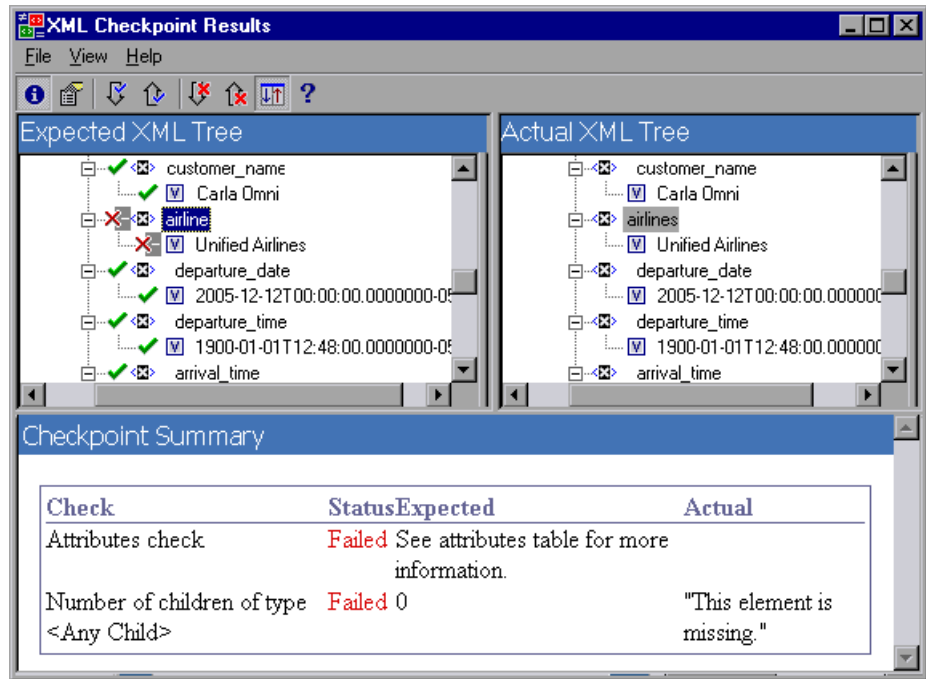
Below are four sample XML checkpoint scenarios. Each example describes the changes that occurred in the actual XML document, explains how you locate the cause of the problem in the XML checkpoint results, and displays the corresponding XML Checkpoint Results window.

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.

The screenshot shows the 'XML Checkpoint Results' window with four panes. The top-left pane, 'Expected XML Tree', shows an XML structure with a red 'X' icon next to the 'orders' element. The top-right pane, 'Actual XML Tree', shows the same structure but with a blue 'X' icon next to the 'orders' element. The bottom-left pane, 'Expected Attributes', contains a table with two rows. The bottom-right pane, 'Actual Attributes', contains a table with two rows. The 'Result' column in the 'Expected Attributes' table shows 'Failed' for the first row and 'Passed' for the second row.

Expected XML Tree			
diffgr:diffgram	✓		
NewDataSet	✓		
orders	✗		
order_number	✓	1060	
customer_name	✓	Carla Omni	
airline	✓	Unified Airlines	
departure_date	✓		

Actual XML Tree			
diffgr:diffgram	✗		
NewDataSet	✗		
orders	✗		
order_number	✗	1060	
customer_name	✗	Carla Omni	
airline	✗	Unified Airlines	
departure_date	✗		

Expected Attributes			
	Name	Value	Result
1	diffgr:id	orders1	Failed
2	msdata:rowOrder	0	Passed

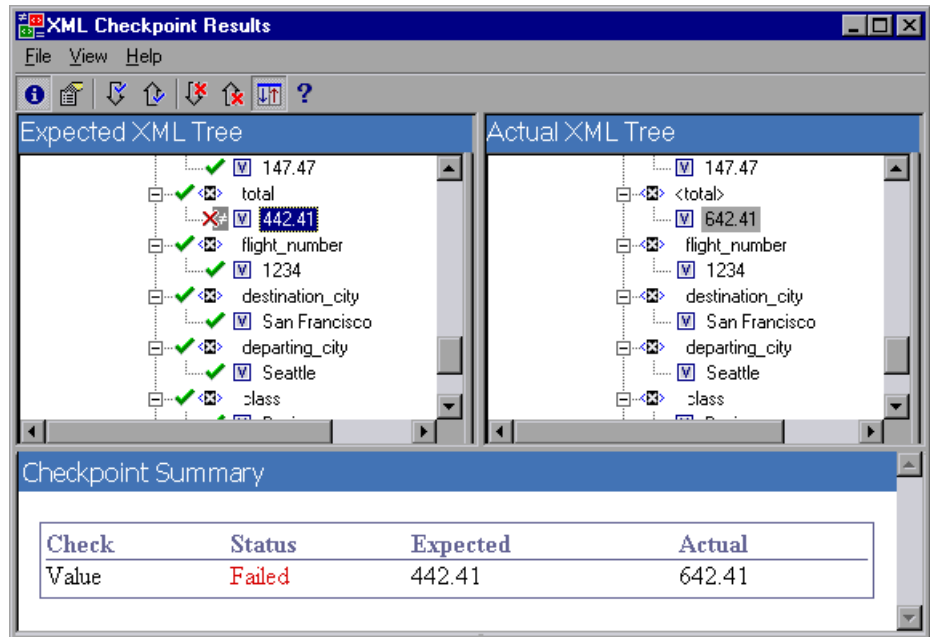
Actual Attributes			
	Name	Value	
1	diffgr:id	orders2	
2	msdata:rowOrder	0	

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.



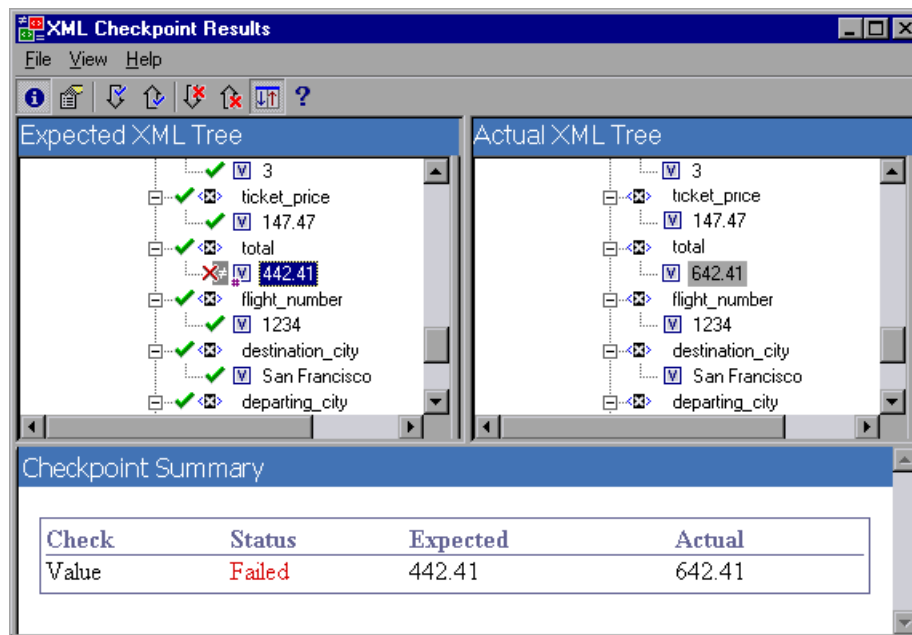
Scenario 4

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.

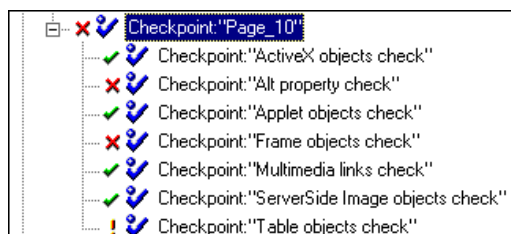
Using the Checkpoint Summary pane, you can compare the expected and actual values of the **total** element.



Analyzing Accessibility Checkpoint Results

When you include accessibility checkpoints in your test, 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 29, “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	ControlX

Alt Property Check

Guideline 1.1 of the W3C Web Content Accessibility Guidelines requires you to provide a text equivalent for every non-text element. The Alt property check checks whether objects that require the Alt property under this guideline, do in fact have this attribute. If the selected frame or page does not contain any such objects, or if all such objects have the required attribute, the checkpoint passes. If one or more objects that require the property do not have it, the test fails and the test 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 fails and the test results details display a list that shows which objects are lacking the tag. For example:

Frame titles check			
Object Class	Object Tag	Object Name	Title Value
Frame	IFRAME	takeOver	Takeover Ad
Frame	IFRAME	adSpotFrame5	Click here to find out more!
Frame	IFRAME	theFrame	[NONE]
Page		NBA.com	NBA.com

The bottom 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:

Server-side Image 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	

Viewing Parameterized Values and Output Value Results

You can view information on parameterized values and the results of output value steps in the Test Results window. You can also view the contents of the run-time Data Table.

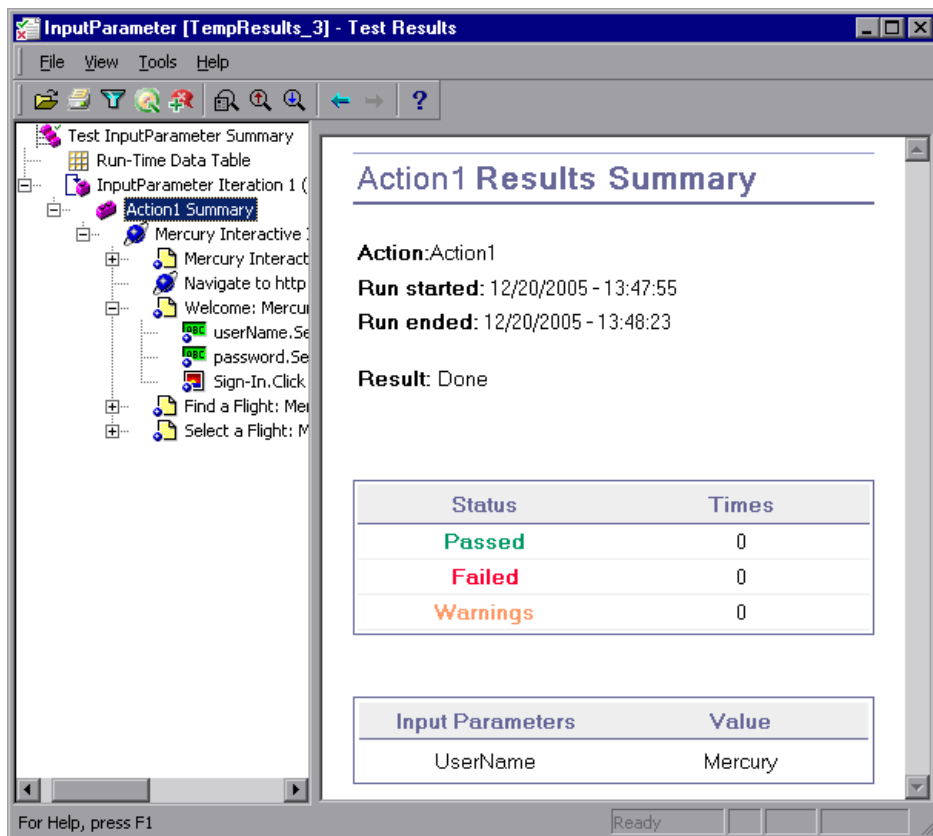
Viewing Parameterized Values in the Test Results Window

A **parameter** is a variable that is assigned a value from an external data source or generator. You can view the values for the parameters defined in your test in the Test Results window.

To view parameterized values:

- 1** Display the test results for your test in the Test Results window. For more information, see “Viewing the Results of a Run Session” on page 638.
- 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 or action that contains parameterized values.

The name and value of the input parameters are displayed at the bottom of the right pane.



The example above shows the input parameter `UserName` defined for the action with the value `Mercury`.

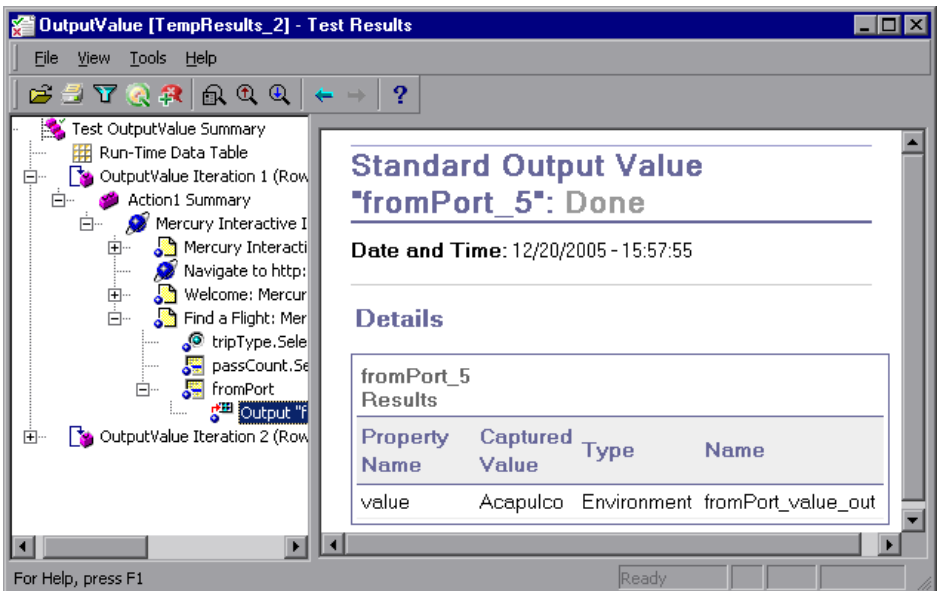
For more information on defining and using parameters in your tests, see Chapter 16, "Parameterizing Values."

Viewing Output Value Results in the Test Results Window

An **output value** is a step in which one or more values are captured during the run session for use at another point in the run. When one of the values is needed later in the run as input, QuickTest retrieves it from the specified output location.

To view the results of an output value step:

- 1 Display the test results for your test in the Test Results window. For more information, see “Viewing the Results of a Run Session” on page 638.
- 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 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 17, “Outputting Values.”

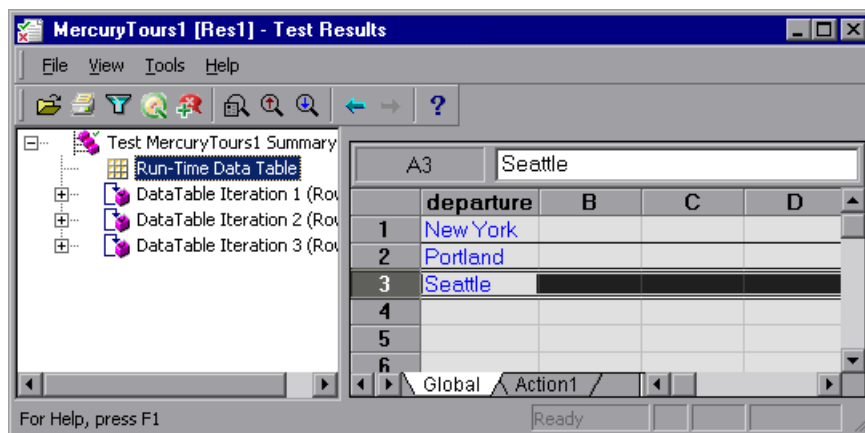
For information on viewing the results of XML output value steps, see “Analyzing XML Output Value Results” on page 679.

Viewing the Run-Time Data Table

After running a test 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.

To view the run-time Data Table:

- 1 Display the test results for your test in the Test Results window. For more information, see “Viewing the Results of a Run Session” on page 638.
- 2 Highlight **Run-Time Data** in the left pane of the Test Results window.



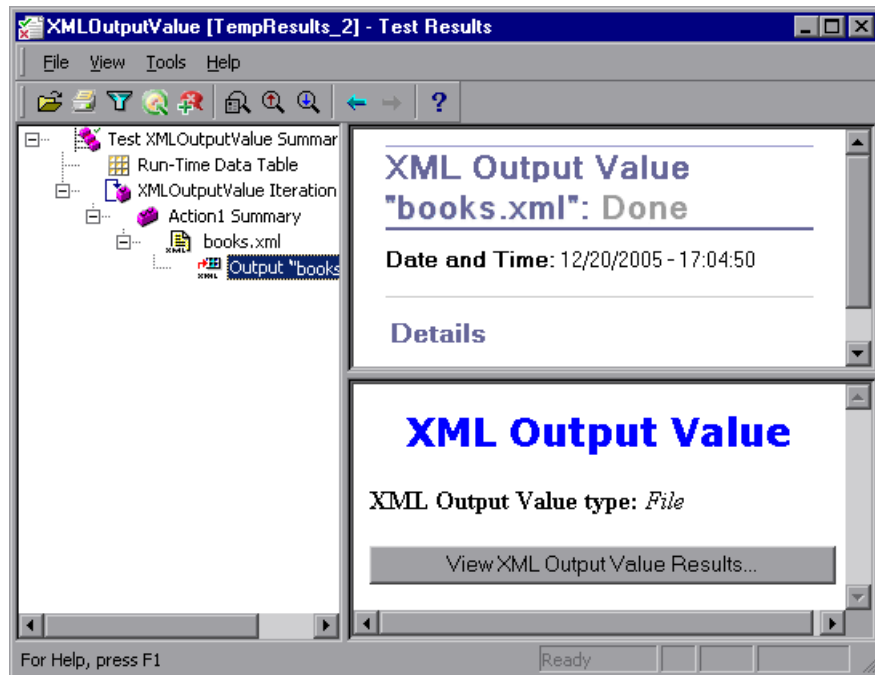
In the above example, the Run-Time Data Table contains the parameterized flight departure values.

For more information on the run-time Data Table, see Chapter 20, “Working with Data Tables.”

Analyzing XML Output Value Results

You can output element or attribute values to your test from XML documents used in your application. For more information on XML output values, see “Outputting XML Values” on page 454.

You can view summary results of the XML output value in the Test Results window. For information on displaying the results for a checkpoint, see “Viewing Checkpoint Results” on page 653.



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.

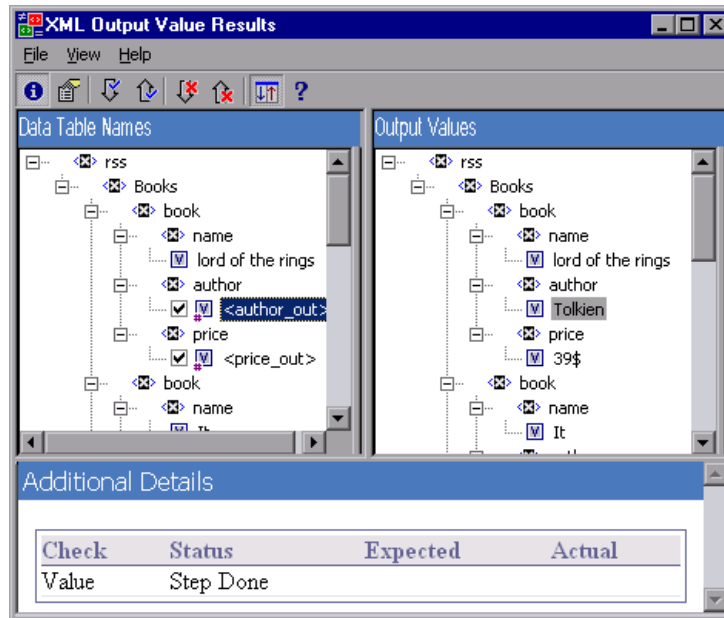
Understanding the XML Output Value Results Window

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.



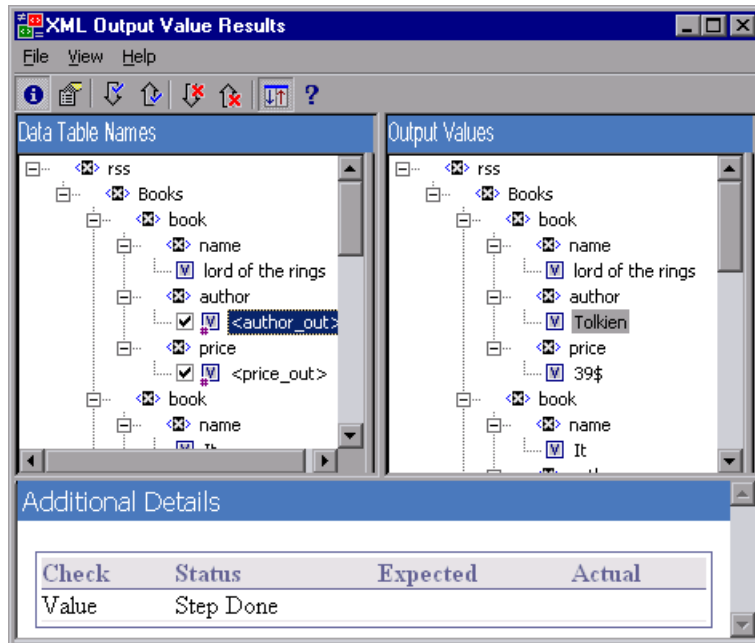
Navigating the XML Output Value Results Window

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:

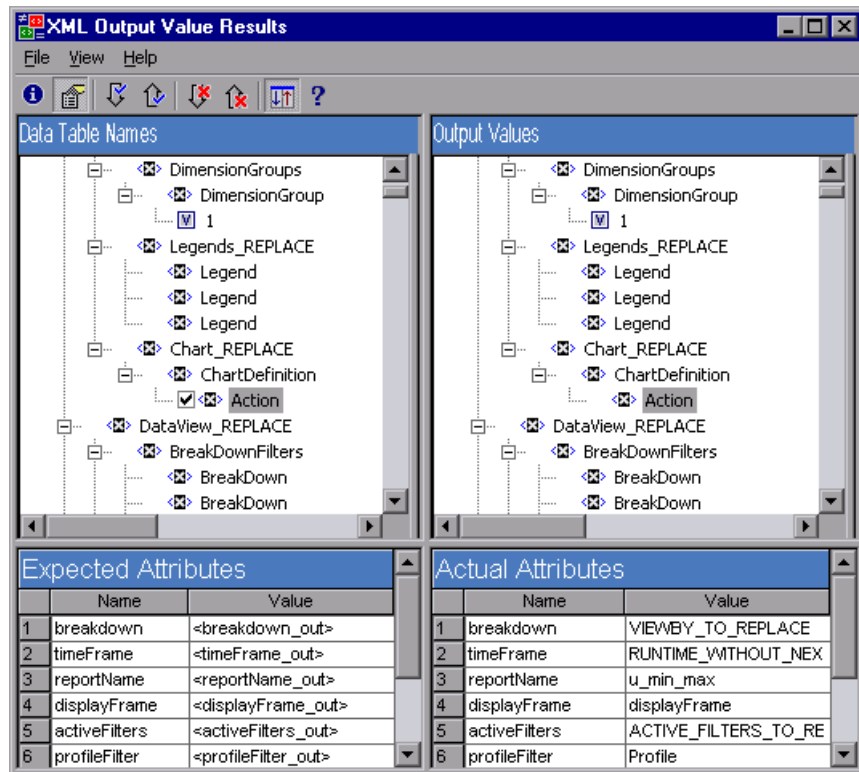


- **View Output Value Summary.** Select an element in the XML Tree and click the **View Output Value Summary** button or choose **View > Output Value Summary**. The Additional Details pane, which provides information regarding the output value for the selected element, attribute, or value, is displayed 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.



- **Find Next Error.** Choose **View > Find Next Error** or click the **Find Next Error** button to jump directly to the next error in the XML Tree.



- **Find Previous Error.** Choose **View > Find Previous Error** or click the **Find Previous Error** button to jump directly to the previous error in the XML Tree.



- **Scroll Trees Simultaneously.** 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. The following examples illustrate two possible scenarios.

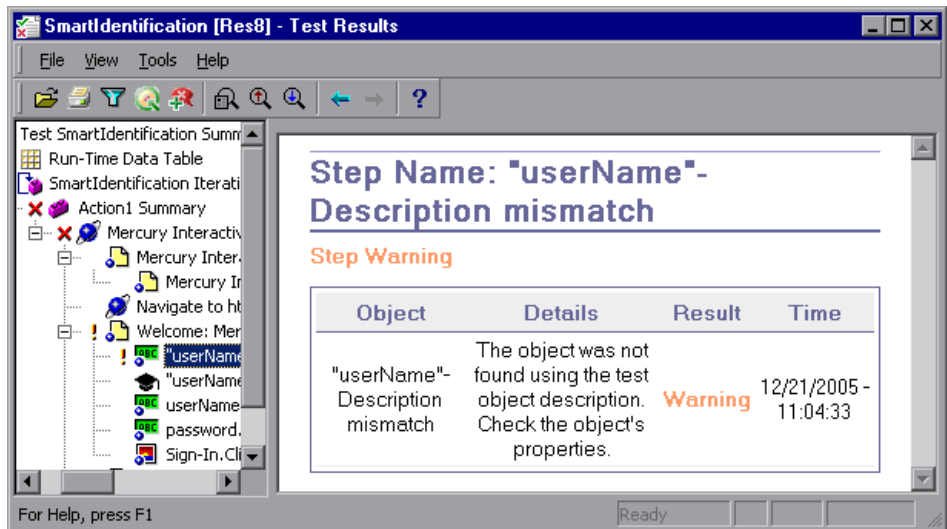
Smart Identification—No Object Matches the Recorded Description

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:
<p>A description mismatch icon for the missing object. For example:</p>  "userName"- Description mismatch	<p>An indication that the object (for example, the <code>userName</code> WebEdit object) was not found.</p>
<p>A Smart Identification icon for the missing object. For example:</p>  "userName"- Smart Identification	<p>An indication that the Smart Identification mechanism successfully found the object, and information on the properties used to find the object. You can use this information to modify the recorded test object description, so that QuickTest can find the object using the description in future run sessions.</p>
<p>The actual step performed. For example:</p>  <code>userName.Set "Mercury"</code>	<p>Normal result details for the performed step.</p>

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 `userName` WebEdit object after one of the recorded description property values changed.



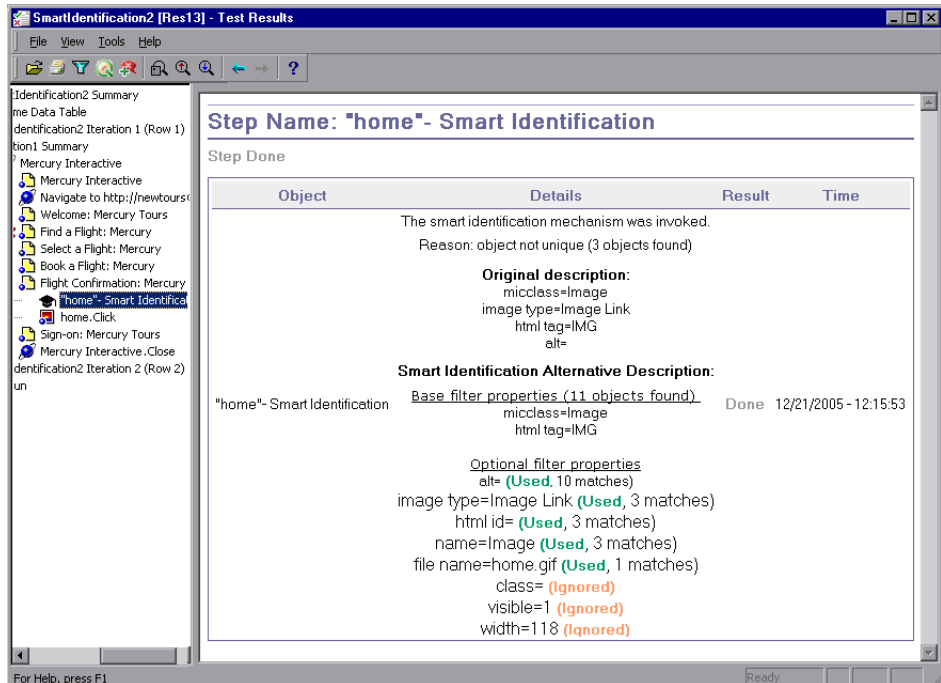
Smart Identification—Multiple Object Match the Recorded Description

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 window. The step still receives a passed status, because in most cases, if Smart Identification was not used, the test object description plus the ordinal identifier could have potentially identified the object.

In such a situation, the Test Results show the following information:

In the results tree:	In the results details:
<p>A Smart Identification icon for the missing object. For example:</p> <p> "home"- Smart Identification</p>	<p>An indication that the Smart Identification mechanism successfully found the object, and information on the properties used to find the object. You can use this information to create a unique object description for the object, so that QuickTest can find the object using the description in future run sessions.</p>
<p>The actual step performed. For example:</p> <p> home.Click</p>	<p>Normal result details for the performed step.</p>

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 fails and a normal failed step is displayed in the Test Results.

Deleting Test Run Results

You can use the Test Results Deletion Tool to remove unwanted or obsolete test results from your system, according to specific criteria that you define. This enables you to free up valuable disk space.

You can use this tool with a Windows-style user interface, or you can use the Windows command line to run the tool in the background (silently) to directly delete results that meet criteria that you specify.

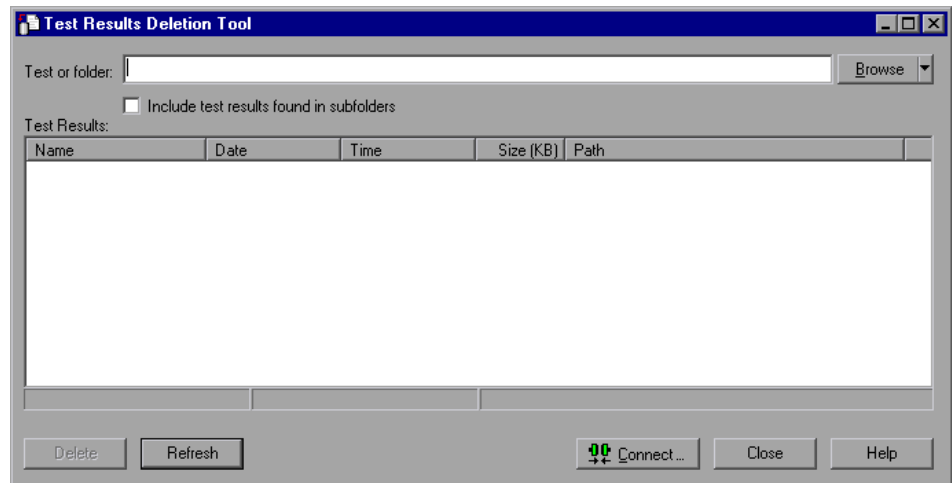
Deleting Results Using the Test Results Deletion Tool

You can use the Test Results Deletion Tool to view a list of all the test results in a specific location in your file system or in a Quality Center project. You can then delete any test results that you no longer require.

The Test Results Deletion Tool enables you to sort the test results by name, date, size, and so forth, so that you can easily identify the results you want to delete.

To delete test results using the Test Results Deletion Tool:

- 1 Choose **Start > Programs > QuickTest Professional > Tools > Test Results Deletion Tool** from the **Start** menu. The Tests Results Deletion Tool window opens.



- 2** In the **Test or folder** box, specify the 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 1261.

- 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, 2005.

TestResultsDeletionTool.exe -Silent -Test "C:\tests\test1" -FromDate "11/1/2005"

-Log *log_file_path*

Creates a log file containing an entry for each test results file in the folder or test you specified. The log file indicates which results were deleted and the reasons why other results were not. For example, results may not be deleted if they are smaller than the minimum file size you specified.

You can specify a file path and name or use the default path and name. If you do not specify a file name, the default log file name is

TestResultsDeletionTool.log in the folder where the Test Results Deletion Tool is located.

The following example creates a log file in **C:\temp\Log.txt**.

TestResultsDeletionTool.exe -Silent -Log "C:\temp\Log.txt" -Test "C:\tests\test1"

The following example creates a log file named **TestResultsDeletionTool.log** in the folder where the Test Results Deletion Tool is located.

```
TestResultsDeletionTool.exe -Silent -Log -Test "C:\tests\test1"
```

-MinSize *minimum_file_size*

Deletes test results larger than or equal to the specified minimum file size. Specify the size in bytes.

Note: The **-MinSize** option is available only for test results in the file system. It is not supported when working with tests in Quality Center.

The following example deletes all results larger than or equal to 10000 bytes. Results that are smaller than 10000 bytes are not deleted.

```
TestResultsDeletionTool.exe -Silent -Test "C:\tests\test1" -MinSize "10000"
```

-Name *result_file_name*

Specifies the 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 on regular expressions and regular expression syntax, see “Understanding and Using Regular Expressions” on page 352.

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 opens the Test Results Deletion Tool with a list of the results in the **F:\Tests\Keep\webobjects** folder.

```
TestResultsDeletionTool.exe -Test "F:\Tests\Keep\webobjects"
```

The following example deletes all results in the Quality Center **Tests\webobjects** test:

```
TestResultsDeletionTool.exe -Domain "Default" -Server "http://QCServer/qcbin" -  
    Project "Quality Center_Demo592" -User "Admin" -Password "PassAdmin" -  
    Test "Subject\Tests\webobjects"
```

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, 2005.

```
TestResultsDeletionTool.exe -Silent -Test "C:\tests\test1" -UntilDate "11/1/2005"
```

-User Quality_Center_user_name

Specifies the user name for the Quality Center project to which you want to connect. This option should be used in conjunction with the **-Domain**, **-Server**, **-Project**, and **-Password** options.

This option is available only when used in conjunction with the **-Silent** option.

Submitting Defects Detected During a Run Session

You can instruct QuickTest to automatically submit a defect to a Quality Center project for each failed step in your test. You can also manually submit a defect for a specific step to Quality Center directly from within your QuickTest Test Results window. These options are only available when you are connected to a Quality Center project.

For more information on working with Quality Center and QuickTest, see Chapter 45, “Working with Quality Center.” For more information on 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 more information on connecting to Quality Center, see Chapter 45, “Working with Quality Center”.

Note: If you do not connect to a Quality Center project before proceeding to the next step, QuickTest prompts you to connect before continuing.



- 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 on the test 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 'Flight Details' Failed	

- 4 Click **Submit** to add the defect information to the Quality Center project.
- 5 Click **Close** to close the Add Defect dialog box.

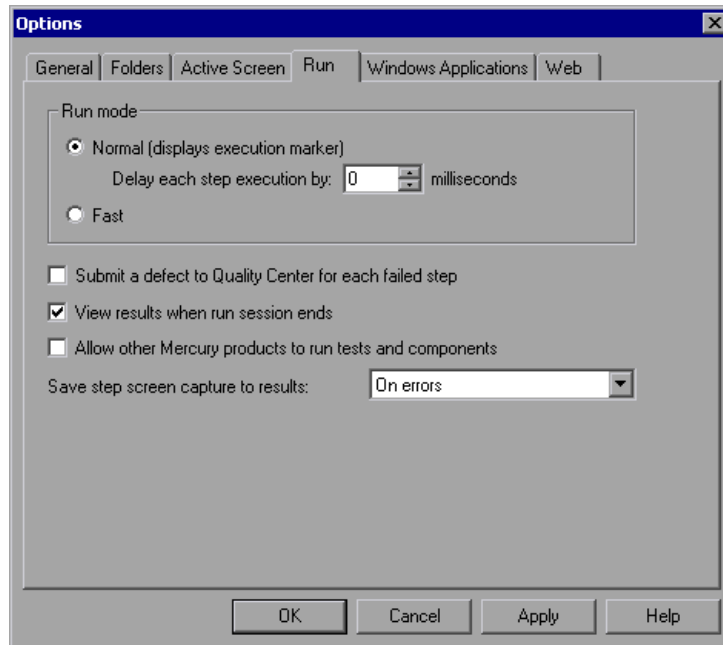
Automatically Submitting Defects to a Quality Center Project

You can instruct QuickTest to automatically submit a defect to the Quality Center project specified in the Quality Center Connection dialog box (**File > Quality Center Connection**) for each failed step in your test.

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

Standard Checkpoint "Flight Details_4" failed

Test name: Recording
Test location: C:\Program Files\Mercury Interactive\QuickTest Professional
\Tests\Tutorial\Recording on BINDER
Action name: Action1

Operating system : Windows 2000
Host: BINDER

Additional Information:
Verification type: String Content.
Settings: Exact match - ON; Ignore space - ON; Match case - OFF.
Results: Checked 28 cells;
Succeeded: 27;
Failed: 1

```

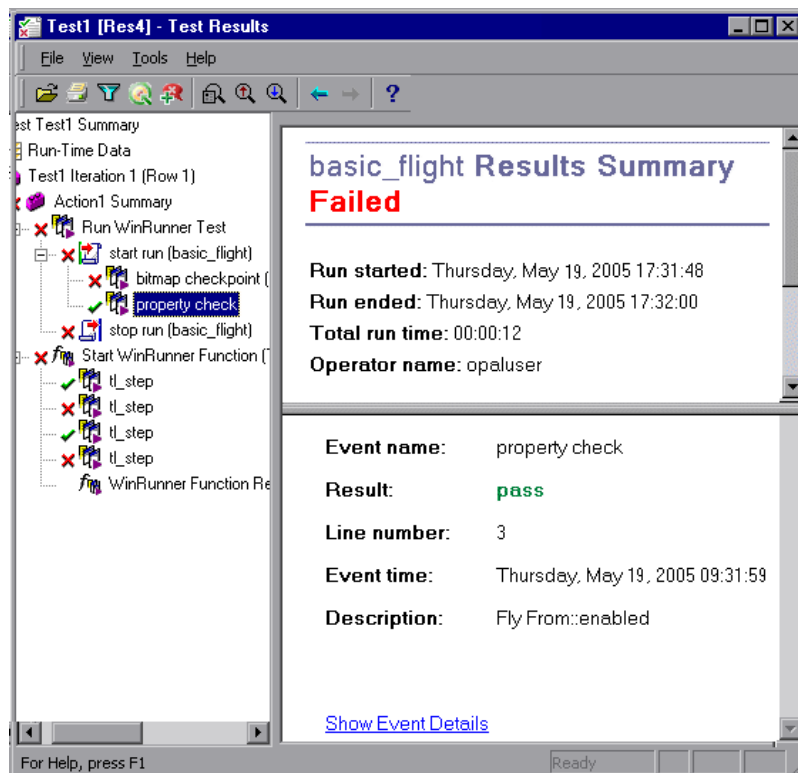
Viewing WinRunner Test Steps in the Test Results




If your QuickTest test includes a call to a WinRunner test 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 7.6 test steps of a result created using QuickTest Professional 8.0 or later only after you have installed patch **WR76P44 - Support WR/QTP integration** on WinRunner 7.6. (You do not need to install this patch on WinRunner 8.0 or later.) This patch is available in the patch database on the Mercury Customer Support site (<http://support.mercury.com>).

Tip: If you have WinRunner version 7.5 installed on your computer, you can view basic information on the WinRunner test run in the QuickTest Test Results window. You can also open the WinRunner Test Results window for additional details.

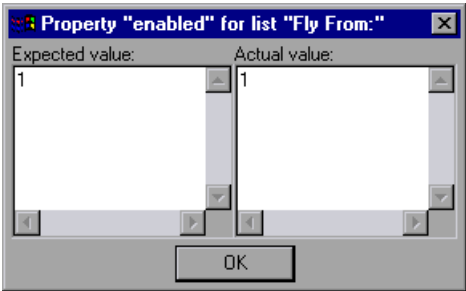
The left pane of the QuickTest test results include a node for each WinRunner event that would normally be included in the WinRunner results. When you select a node corresponding to a WinRunner test event or function call, the right pane displays a summary of the called WinRunner test or function and 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 on 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>

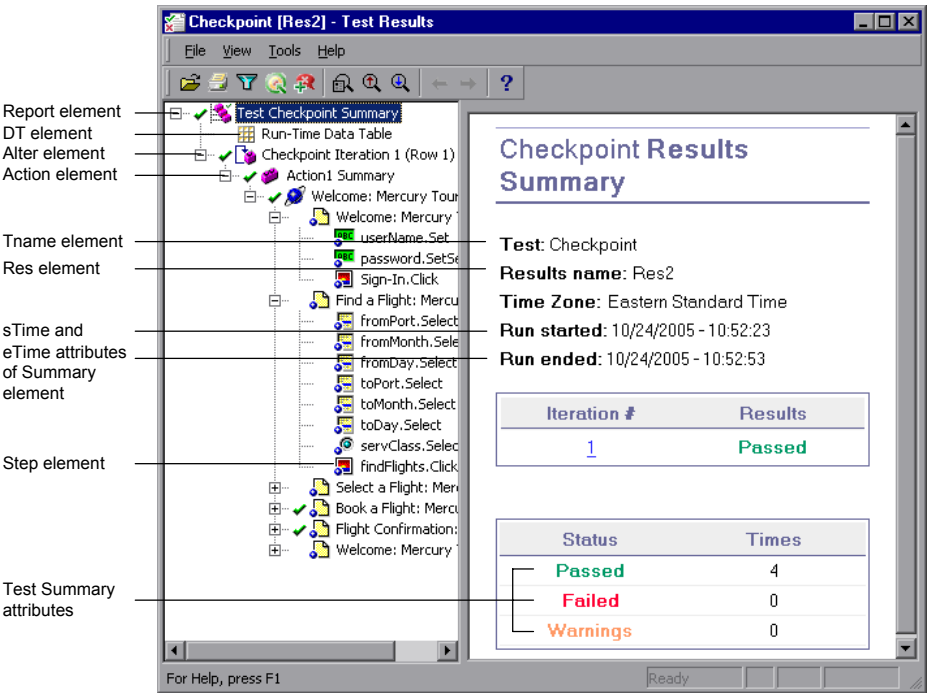
For more information on running WinRunner tests and functions from QuickTest, see Chapter 44, “Working with WinRunner.”

Customizing the Test Results Display

The results of each QuickTest run session are saved in a single **.xml** file (called **results.xml**). This **.xml** file stores information on each of the test result nodes in the display. The information in these nodes is used to dynamically create **.htm** files that are shown in the top-right pane 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, when displaying test results in your own customized results viewer, or when exporting the test results to an HTML file).

The diagram below shows the correlation between some of the elements in the .xml file and the items they represent in the test results.



Tip: You can change the appearance (look and feel) of the Test Results window. For more information, see “Changing the Appearance of the Test Results Window” on page 638.

XSL provides you with the tools to describe exactly which test result information to display and exactly where and how to display, print or export it. You can also modify the .css file referenced by the .xsl file, to change the appearance of the report (for example, fonts, colors, and so forth).

For example, in the **results.xml** file, one element tag contains the name of an action, and another element tag contains information on the time at which the run session is performed. Using XSL, you could tell your customized test results viewer that the action name should be displayed in a specific place on the page and in a bold green font, and that the time information should not be displayed at all.

You may find it easier to modify the existing **.xsl** and **.css** files provided with QuickTest, instead of creating your own customized files from scratch. The files are located in **<QuickTest Installation Folder>\dat**, and are named as follows:

- **PShort.xsl**. Specifies the content of the test results report printed, or exported to an HTML file, when you select the **Short** option in the Print or Export to HTML File dialog boxes.
- **PDetails.xsl**. Specifies the content of the test results report printed, or exported to an HTML file, when you select the **Detailed** option in the Print or Export to HTML File dialog boxes.
- **PSelection.xsl**. Specifies the content of the test results report printed, or exported to an HTML file, when you select the **Selection** option in the Print or Export to HTML File dialog boxes.
- **PResults.css**. Specifies the appearance of the test results print preview. This file is referenced by all three **.xsl** files.

For more information on printing test results using a customized **.xsl** file, see “Printing Test Results” on page 648.

For more information on exporting the test results to an HTML file using a customized **.xsl** file, see “Exporting Test Results” on page 651.

For information on the structure of the XML schema, and a description of the elements and attributes you can use to customize the test results reports, refer to the XML Report Help (**Help > QuickTest Professional Help > QuickTest Advanced References > QuickTest Test Results Schema**).

Part IV

Configuring Basic Settings

25

Setting Global Testing Options

You can control how QuickTest records and runs tests by setting global testing options.

This chapter describes:	On page:
About Setting Global Testing Options	707
Using the Options Dialog Box	708
Setting General Testing Options	710
Setting Folder Testing Options	712
Setting Active Screen Options	715
Setting Run Testing Options	723
Setting Windows Application Testing Options	726
Setting Web Testing Options	739

About Setting Global Testing Options

Global testing options affect how you record and run tests, as well as the general appearance of QuickTest. For example, you can choose not to display the Welcome screen when QuickTest starts, or you can set the timing-related settings used by QuickTest when running a test. The values you set remain in effect for all tests and for subsequent testing sessions.

You can also set testing options that affect only the test currently open in QuickTest. For more information, see Chapter 26, “Setting Options for Individual Tests.”

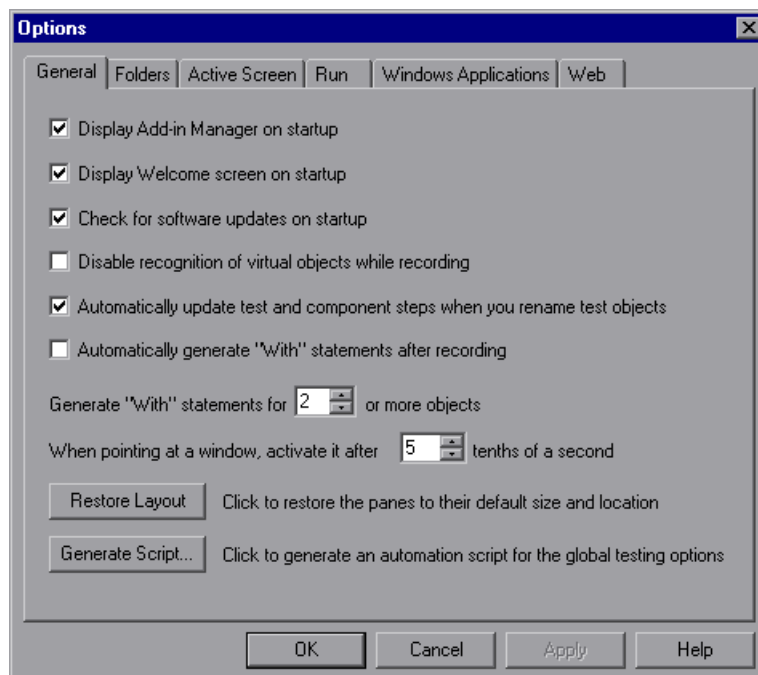
Using the Options Dialog Box

You can use the Options dialog box to modify your global testing options. The values you set remain in effect for all subsequent 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. For information on the available options in each tab, see the table below.

- 3** Click **Apply** to apply your changes and keep the dialog box open, or click **OK** to save your changes and close the dialog box.

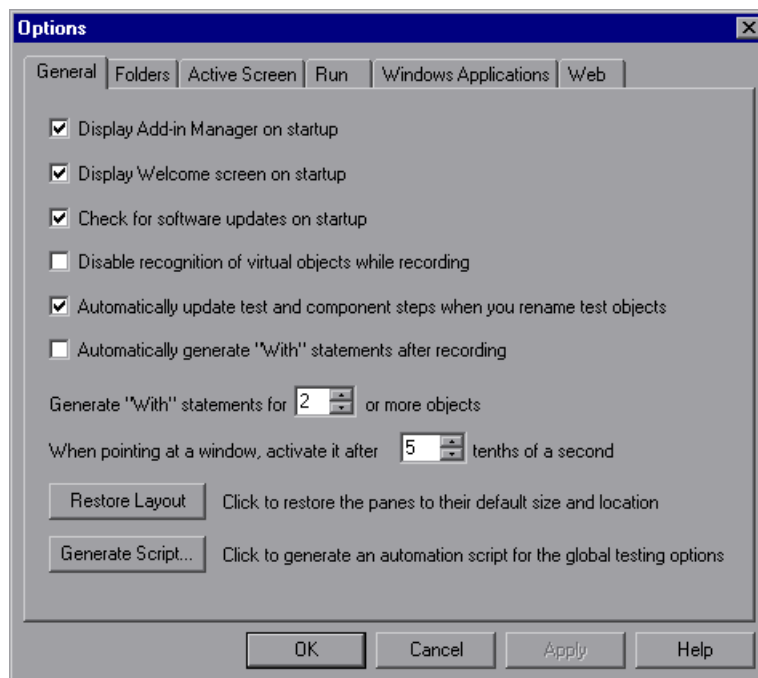
The Options dialog box can contain the following tabbed pages:

Tab Heading	Contains:
General	Options for general test settings. For more information, see “Setting General Testing Options” on page 710.
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 712.
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 715.
Run	Options for running tests. For more information, see “Setting Run Testing Options” on page 723.
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 726.
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 739.

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 811.
Display Welcome screen on startup	Determines whether the Welcome screen is displayed when starting QuickTest.
Check for software updates on startup	Instructs QuickTest to automatically check for software updates each time it starts up. For more information, see “Updating the QuickTest Software” on page 15.
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 31, “Learning Virtual Objects.”
Automatically update test and component steps when you rename test objects	Determines whether to automatically update test and component steps when you rename test objects in the local or shared object repository. For more information, see “Renaming Test Objects” on page 174.
Automatically generate “With” statements after recording	Instructs QuickTest to automatically generate With statements when you record. For more information, see “Generating With Statements for Your Test” on page 569.
Generate “With” statements for __ or more objects	Indicates the minimum number of identical, consecutive objects for which you want to apply the With statement. This setting is used when QuickTest automatically generates With statements after recording and when you select to generate With statements for an existing action. Default = 2. For more information, see “Generating With Statements for Your Test” on page 569.

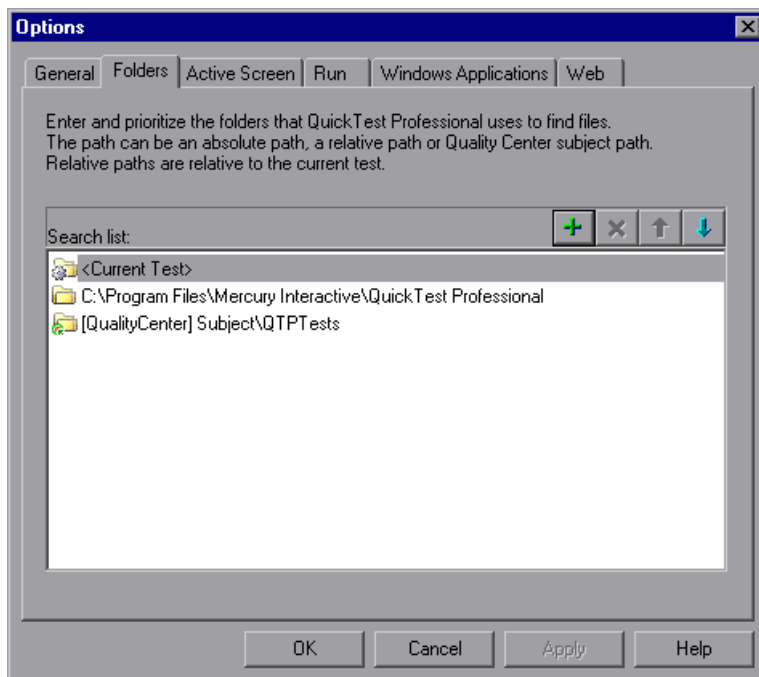
Option	Description
When pointing at a window, activate it after ___ tenths of a second	Specifies the time (in tenths of a second) that QuickTest waits before it sets the focus on an application window when using the pointing hand to point to an object in the application (for Object Spy, checkpoints, Step Generator, Recovery Scenario Wizard, and so forth). Default = 5.
Restore Layout	Restores the layout of the QuickTest window so that it displays the panes and toolbars in their default sizes and positions.
Generate Script	Generates an automation script containing the current global testing options. For more information, see “Automating QuickTest Operations” on page 1105, or refer to the <i>QuickTest Automation Reference</i> (Help > QuickTest Professional Help > QuickTest Advanced References > QuickTest Automation).

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 steps. For example, suppose you add the folder in which all of your tests are stored to the folders list. If you later insert a copy of an action to a test, you only have to enter the name of the test containing the action you want to insert in the Insert Copy of Action dialog box. QuickTest searches for the test’s path in the folders you specified in the Folders tab.

Note: The current test 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, action, or file.



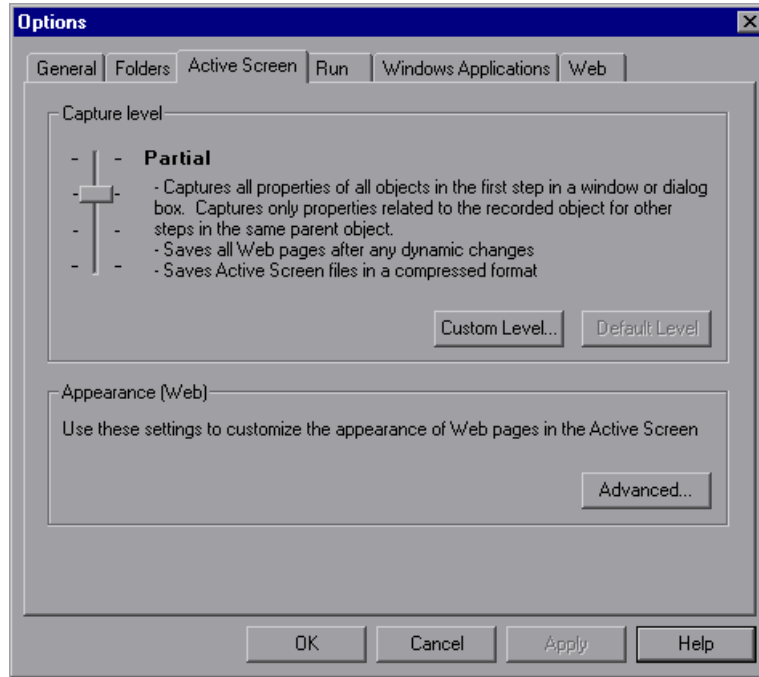
The Folders tab includes the following options:

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, 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.



Tip: The more information saved in the Active Screen, the easier it is to edit the test after it is recorded. However, more information saved in the Active Screen adds to the recording time and disk space required. This is especially critical in Visual Basic, ActiveX, and .NET Windows Forms environments.

You can increase or decrease the amount of information saved in your test after it is recorded. For more information, see “Increasing or Decreasing the Active Screen Information Saved with a Test” on page 221.

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.</p> <p>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.

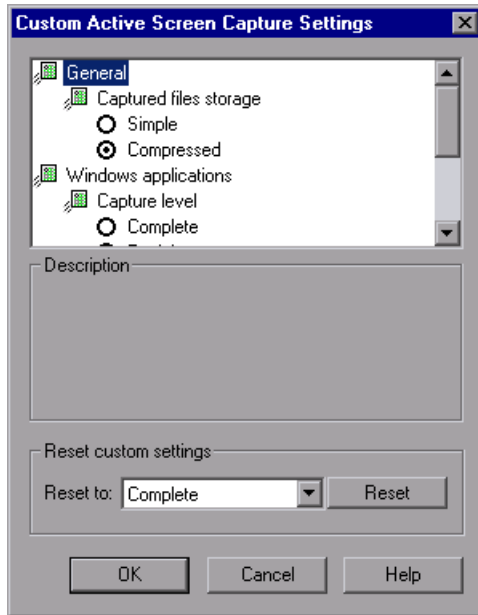
Option	Description
Custom Level	Enables you to specify custom Active Screen options. For more information, see “Custom Active Screen Capture Settings” on page 717.
Default Level	Returns the capture level settings to the predefined default level (Partial).
Advanced	Enables you to define the appearance of Web pages in the Active Screen, see “Web Page Appearance” on page 721.

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.

When you apply custom Active Screen settings, you override the capture-level setting in the Active Screen tab with all of the settings in the Custom Active Screen Capture Settings dialog box.

Note that the default settings in the Custom Active Screen Capture Settings dialog box do not reflect the selected capture-level setting in the Active Screen tab of the Options dialog box. If you want to customize only specific settings, use the **Reset to** option to ensure that all other settings are using the capture-level setting you prefer and then modify the specific settings you need.



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 105.

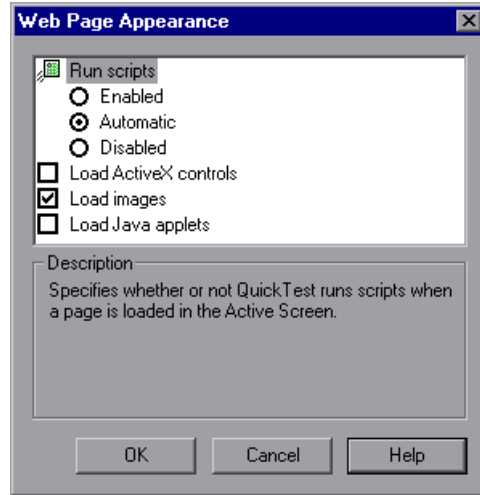
- **Capture original HTML source.** Captures the HTML source of Web pages as they appear originally, before any scripts are run. Deselecting this option instructs QuickTest to capture the HTML source of Web pages after any dynamic changes have been made to the HTML source (for example, by scripts running 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 719.

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:

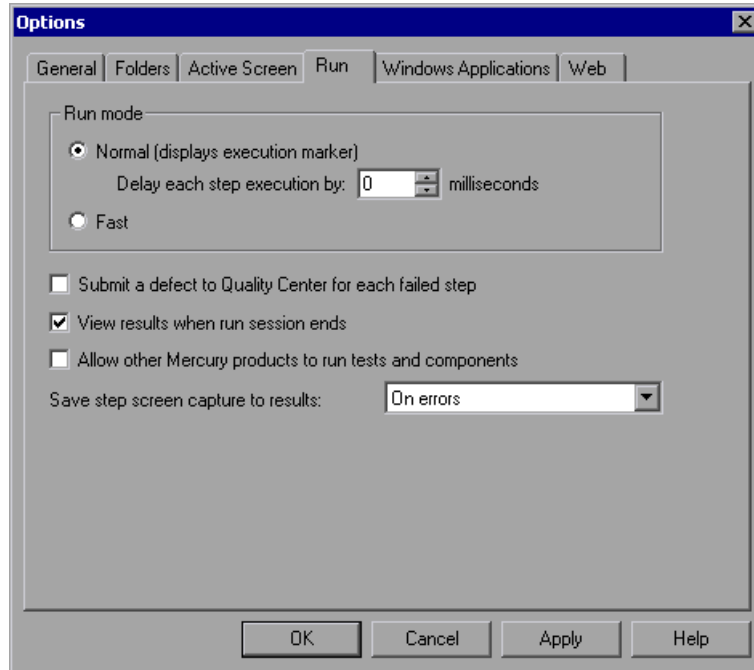
QuickTest loads ActiveX controls or Java applets to the Active Screen in view-only mode. You cannot perform operations or retrieve additional information on the loaded ActiveX or Java objects.

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 displays run session results in the Test Results window.



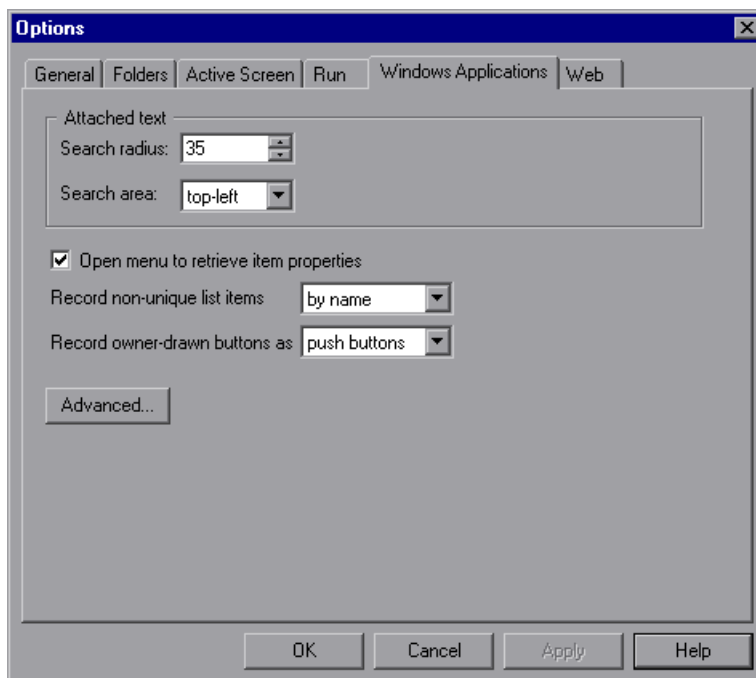
The Run tab includes the following options:

Option	Description
Run mode	<p>Instructs QuickTest how to run your test:</p> <ul style="list-style-type: none">• Normal (displays execution marker). Runs your test with the execution arrow to the left of the Keyword View or Expert View, marking each step or statement as it is performed. If the test contains multiple actions, the tree in the Keyword View Item column expands to display the steps, and the Expert View displays the script, of the currently running action. <p>Delay each step execution by. You can specify the time in milliseconds that QuickTest should wait before running each consecutive step (up to a maximum of 10000 ms.)</p> <p>The Normal run mode option requires more system resources than the Fast option, described below.</p> <p>Note: You must have Microsoft Script Debugger installed to enable this mode. For more information, refer to the <i>QuickTest Professional Installation Guide</i>.</p> <ul style="list-style-type: none">• Fast. Runs your test without the execution arrow to the left of the Keyword View or Expert View and does not expand the item tree or display the script of each action as it runs. This option requires fewer system resources. <p>Note: When running a test set from Quality Center, tests are automatically run in Fast mode, even if Normal mode is selected.</p>

Option	Description
Submit a defect to Quality Center for each failed step	Instructs QuickTest to automatically submit a defect to Quality Center for each failed step in your test. This option is available only when you are connected to a Quality Center project. For more information, see “Automatically Submitting Defects to a Quality Center Project” on page 698.
View results when run session ends	Instructs QuickTest to display the results automatically following the run session.
Allow other Mercury products to run tests and components	Enables other Mercury products such as Quality Center and Test Batch Runner to run QuickTest tests. Note: This option is not required to enable WinRunner to run QuickTest tests.
Save step screen capture to test results	Instructs QuickTest when to capture and save images of the application during the run session to display them in the test results. Choose an option from the list: <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 enable you to configure how QuickTest records and runs tests for Standard Windows, ActiveX, .NET Windows Forms, and Visual Basic applications.



The Windows Applications tab includes the following options:

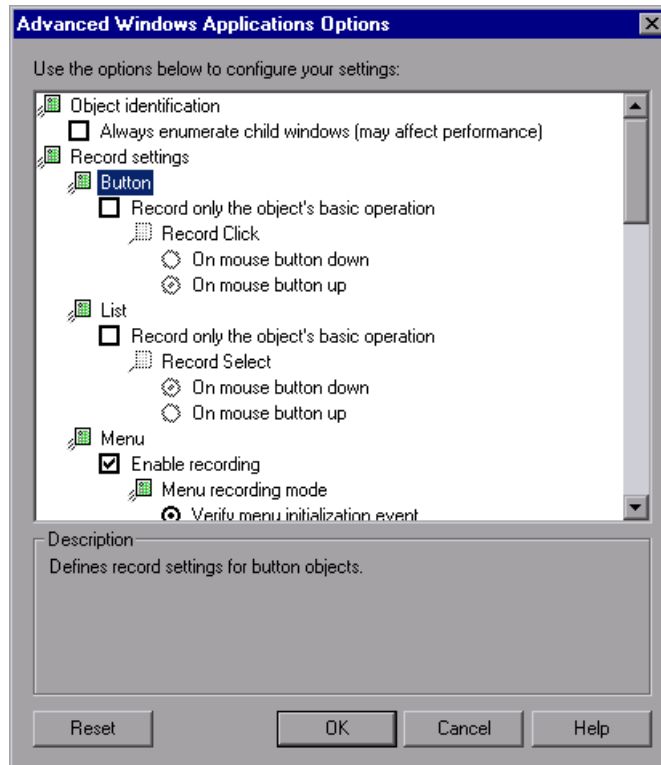
Option	Description
Attached text	<p>Enables you to specify the search criteria that QuickTest uses to retrieve an object's attached text. An object's attached text is the closest static text within a specified radius from a specified point. The retrieved attached text is saved in the object's corresponding 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 Search area list:</p> <ul style="list-style-type: none"> • Top-Left. Top-left corner • Top. Midpoint between the two top corners • Top-Right. Top-right corner • Right. Midpoint between the two right corners • Bottom-Right. Bottom-right corner • Bottom. Midpoint between the two bottom corners • Bottom-Left. Bottom-left corner • Left. Midpoint between the two left corners

Option	Description
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 property to set this behavior for a specified menu object. For more information, refer to the <i>QuickTest Professional Object Model Reference</i>.</p>
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. <p>When the test runs, QuickTest finds and selects the first instance of the name, regardless of the item chosen during recording. Select this option if the all items with the same name have identical properties.</p> <ul style="list-style-type: none"> • by index. Records the item's index number. <p>Select this option if items with the same name do not necessarily have identical properties.</p>
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 889.</p>

Option	Description
Advanced	Opens the Advanced Windows Applications Options dialog box, in which you can customize record and run options for your Windows applications. For more information, see “Advanced Windows Applications Options” on page 729.

Advanced Windows Applications Options

The Advanced Windows Applications Options dialog box enables you to modify how QuickTest records and runs tests 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.



Object Identification Options

You can specify the method QuickTest uses to identify objects when running a test.

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. 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 736.

Record Settings Options

You can specify how QuickTest treats certain objects when recording a test.

The Advanced Windows Applications Options dialog box includes the following **Record settings** options:

Category	Option
Button	<p>Defines record settings for button objects:</p> <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 736. • 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.
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 736. • 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
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 736.</p>
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 736. • 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.

Category	Option
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 736. • 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.
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 736. • 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.

Category	Option
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 736. 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.
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 736. 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
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. <p>For more information, see “Advanced Information” on page 736.</p>
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.

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. 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, refer to <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, refer to <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, refer to <http://msdn.microsoft.com/library/en-us/winui/winui/windowsuserinterface/userinput/keyboardinput/keyboardinputreference/keyboardinputfunctions/getasynckeystate.asp>.

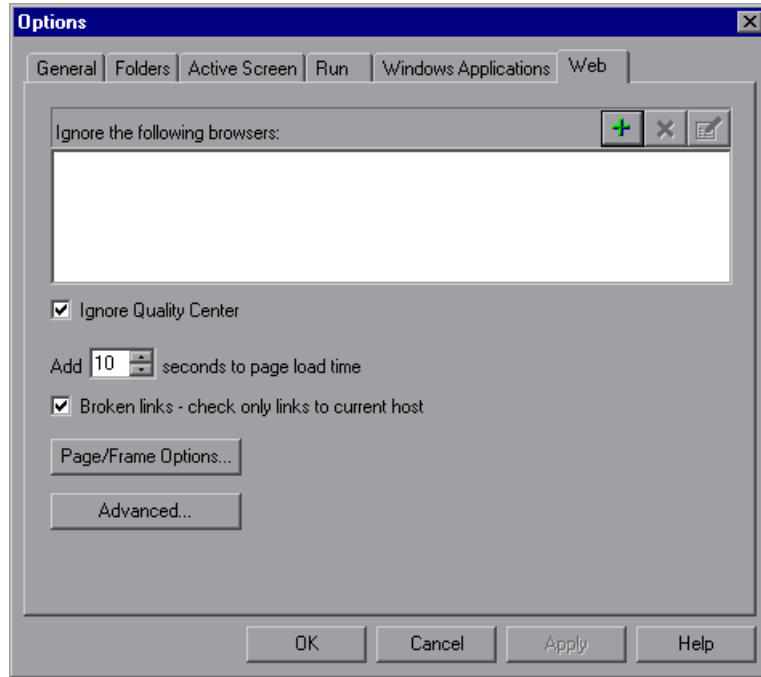
Menu recording mode

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 how QuickTest behaves when recording and running tests on Web sites.



The Web tab includes the following options:

Option	Description
Ignore the following browsers	Instructs QuickTest to ignore any specified browsers that may be open while QuickTest is recording or running a test. For more information, see “Managing the List of Browsers to Ignore” on page 741.
Ignore Quality Center	Instructs QuickTest to ignore all instances of Quality Center that are opened while recording or running a test. By default, this option is selected.
Add __ seconds to page load time	Instructs QuickTest to add a specified number of seconds to the page load time property specified in each Page checkpoint. 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.
Broken links - check only links to current host	Instructs QuickTest to check only for broken links that are targeted to your current host.
Page/Frame Options	Opens the Page and Frame Options dialog box, in which you can customize how QuickTest records Page and Frame test objects. For more information, see “Page and Frame Options” on page 745.
Advanced	Opens the Advanced Web Options dialog box, in which you can customize record and run options for your Web sites. For more information, see “Advanced Web Options” on page 748.

Managing the List of Browsers to Ignore

You can instruct QuickTest to ignore specific browsers that are open while you are recording or running a test. This enables you to keep browsers that are not related to your testing environment open, without having them affect the record or run session. For example, you may want to check your company's share price or the news headlines during the record and run session. If you instructed QuickTest to ignore these specific browsers, they will not affect the session.

Notes: QuickTest ignores browsers that match the defined criteria at the start of a record or run session. However, browsers that do not match the defined criteria at the start of a record or run session, but do match them during the session, are not ignored.

Changes made to these settings apply to new tests and new steps in existing tests only, but not to any other existing steps.

You can also modify the properties that QuickTest uses to identify the browsers to ignore, or delete them from the list of ignored browsers.

Tip: By default, QuickTest ignores all instances of Quality Center that were opened during a record or run session, if the **Ignore Quality Center** check box in the Web tab of the Options dialog box is selected. There is no need to specify Quality Center in the list of browsers to ignore.

Adding a Browser to the List

You can specify the browsers that you want QuickTest to ignore during a record or run session.

Note: QuickTest ignores these browsers only if you selected **Record and run test on any open Web 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 793.

To add a browser to the list:



- 1 To add a browser to the list, click the **Add Browser** button. The Browser Details dialog box opens.

A screenshot of the 'Browser Details' dialog box. It has a title bar with 'Browser Details' and a close button. Inside, there is a 'Name:' label followed by a text field containing 'Browser1'. Below this is a 'Properties' section with a line of text: 'Use one or both of the following properties to identify the browser. Tip: You can use regular expressions for both properties.' There are two checkboxes: 'Title:' and 'URL:'. Each checkbox is followed by an empty text field. At the bottom, there are three buttons: 'OK', 'Cancel', and 'Help'.

- 2 Enter a name for the browser definition in the **Name** field. By default, the name of the browser is **Browser<number of browser in list>**. The name you specify is used only to identify the browser in the list, and is not used by QuickTest.

- 3 Select one or both of the following properties to identify the browser to be ignored, and then enter the following details:
 - **Title.** The name of the Web page as it appears in the title bar of the browser, for example, Yahoo! Finance.*
 - **URL.** The URL of the Web page, for example, `http://www.finance.yahoo.com`. Any descendants of this Web page are automatically included in the list of browsers to ignore.

Tip: You can use regular expressions when specifying the values of these properties. For example, you can use `.*finance.yahoo.com` to specify all `finance.yahoo.com` domains and Web sites starting with `www.`, `http://`, or `https://`. Note that you do not have to use a regular expression to include child pages of a site, as QuickTest automatically ignores the entire domain or site. For information on supported regular expressions, see “Defining Regular Expressions” on page 355.


Note: The **Title** and **URL** properties have an AND relationship, meaning that a browser must match both property values (if defined) to be ignored by QuickTest.

- 4 Click **OK**. The browser is added to the list of ignored browsers.
- 5 Repeat steps 1 to 4 for each browser to be added to the list.

Modifying a Browser in the List

You can modify the definitions of browsers that you want QuickTest to ignore during a record or run session.

To modify a browser in the list:

- 1 Highlight the browser you want to modify.
- 2  Click **Edit Browser Details** button. The Browser Details dialog box opens.
- 3 Make the required changes in the Browser Details dialog box and click **OK**.

Removing a Browser from the List

You can remove a browser from the list if you no longer want QuickTest to ignore it during a record or run session.

Tip: If a browser in the list is required for running a specific test, you can temporarily remove it from the list by clearing the check mark next to its name in the list of browsers.

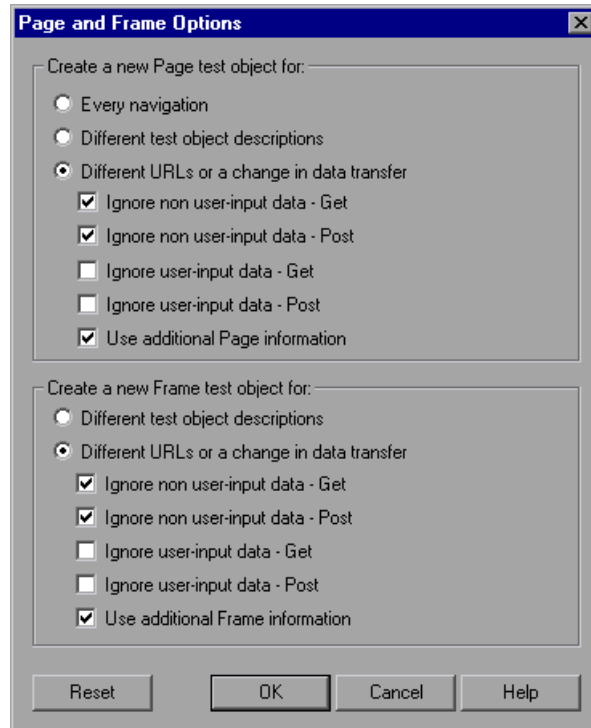
To remove a browser from the list:

- 1 Highlight the browser you want to remove from the list.
- 2 Click the **Remove Browser** button.



Page and Frame Options

The Page and Frame Options dialog box enables you to modify how QuickTest records Page and Frame objects.



Note: You can click the **Reset** button at any time to reset all options to their default core settings. Some external add-ins modify the default settings to optimize page and frame recording. If you are using an external add-in, it is recommended that you keep the default add-in settings and do not use the **Reset** button.

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 748.

The following Page options are available:

- **Every navigation.** Creates a new Page object every time a navigation is performed in 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.** 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).

Note: 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.

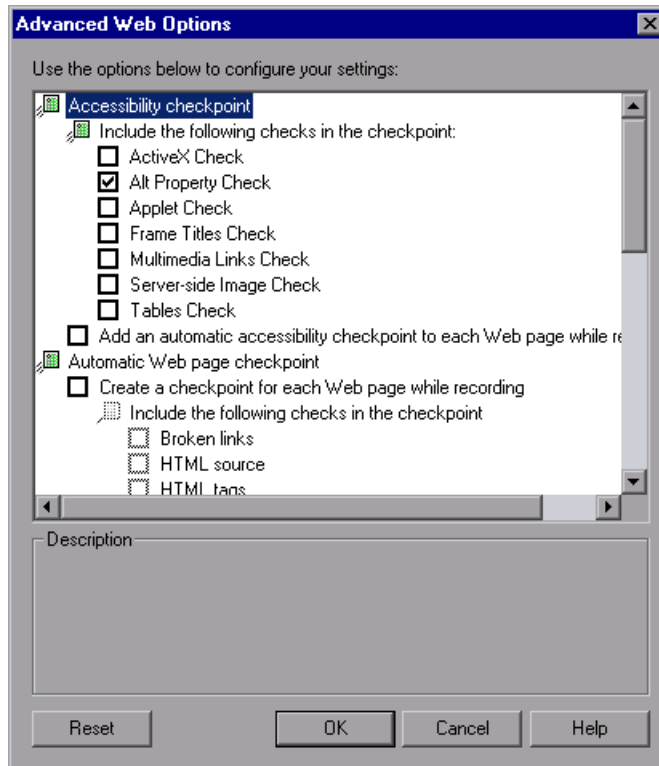
Tip: Select this option to instruct QuickTest to recognize existing pages when the **Back** and **Forward** navigation buttons are used.

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 746.

Advanced Web Options

The Advanced Web Options dialog box enables you to modify how QuickTest records and runs tests 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. All accessibility checkpoints in a test 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:

- **Include the following checks in the checkpoint.** Instructs QuickTest to check the selected accessibility elements for all accessibility checkpoints. Choose from the following:
 - **ActiveX Check.** Checks whether the page or frame contains ActiveX objects. If so, QuickTest sends a warning and displays a list of the objects in the Test Results.
 - **Alt Property Check.** Checks that the <alt> attribute exists for all relevant objects (such as images). If one or more objects lack the required attribute, the test fails and QuickTest displays a list of the objects with the missing attribute in the Test Results. (Selected by default.)
 - **Applet Check.** Checks whether the page or frame contains Java objects. If so, 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 fails and QuickTest displays a list of the frames that lack titles in the Test Results.
 - **Multimedia Links Check.** Checks whether the page or frame contains links to multimedia objects. If so, 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 so, 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 so, QuickTest sends a warning and displays the table format and the tags used in each cell in the Test Results.

For more information, see “Checking Web Content Accessibility” on page 841.

- **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:

- **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 to fail as these checkpoints assume that the page content is static between record and run sessions.

All automatic Page checkpoints include the checks that you select from among the following options:

- **Broken links.** Displays the number of broken links contained in the page during the run session.

Note: If the **Broken links - check only links to current host** option is selected (see “Setting Web Testing Options” on page 739), this number includes only those broken links that are targeted to the current host.

- **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 739).
- **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.
- **Ignore automatic checkpoints while running tests.** Instructs QuickTest to ignore the automatically added Page checkpoints while running your test.

Record Settings

You can set preferences for recording Web objects. The Advanced Web Options dialog box includes the following **Record** settings:

- **Enable Web support for Microsoft Windows Explorer.** 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.

Note: After modifying this setting, for the change to take effect, you must close all instances of Microsoft Windows Explorer (confirm that all **explorer.exe** processes are closed in the Windows Task Manager or restart the computer) and then restart QuickTest.

- **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.** Instructs QuickTest to use standard Windows mouse events instead of browser events for the following events:
 - **OnClick**
 - **OnMouseDown**
 - **OnMouseUp**

Note: Use this option only if the events are not properly recorded using browser events.

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 your Web event configuration to recognize such events. For more information, see Chapter 41, “Configuring Web Event Recording.”

Run Settings

You can set preferences for working with Web objects during a run session. The Advanced Web Options dialog box includes the following Run settings:

- **Browser cleanup.** Closes all open browsers when the current test closes.

When this option is selected, all currently open browsers are closed when the current test 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:
 - **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.
- **Resize browser on run if resized during a recording session.** 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.

Note: To use this option, select the **Open the following browser** option in the Record and Run Settings dialog box before recording.

When this option is cleared, QuickTest does not change the browser size when a run session begins.

26

Setting Options for Individual Tests

You can control how QuickTest records and runs different tests by setting specific testing options for any individual test.

This chapter describes:	On page:
About Setting Options for Individual Tests	756
Using the Test Settings Dialog Box	757
Defining Properties for Your Test	759
Defining Run Settings for Your Test	763
Defining Resource Settings for Your Test	767
Defining Parameters for Your Test	771
Defining Environment Settings for Your Test	774
Defining Web Settings for Your Test	782
Defining Recovery Scenario Settings for Your Test	784

About Setting Options for Individual Tests

You can set testing options that affect how you record and run a specific test. For example, you can instruct QuickTest to run a parameterized test for only certain lines in the Data Table. The individual testing options that you specify are saved when you save the test.

Note: You can also set testing options that affect all tests and components. For more information, see Chapter 25, “Setting Global Testing Options.”

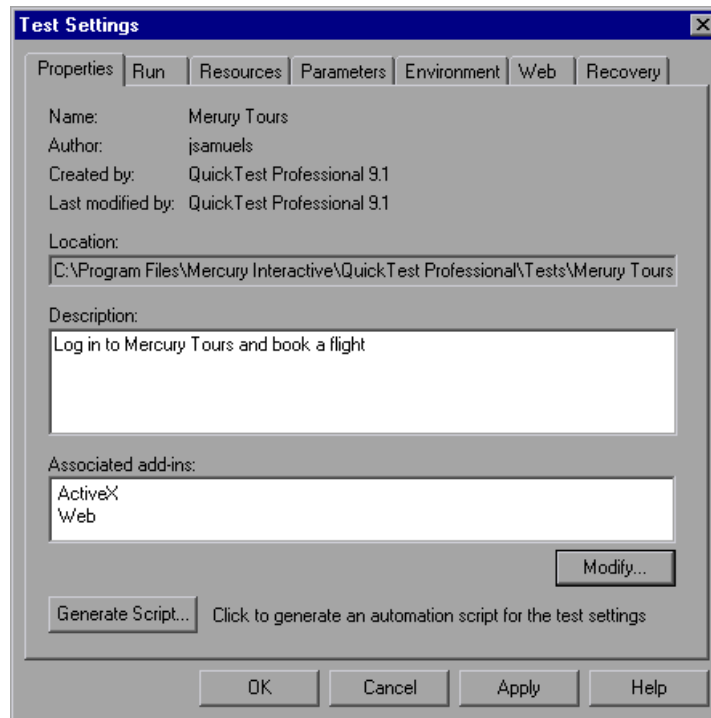
Using the Test Settings Dialog Box

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 **File > Settings** or click the **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.

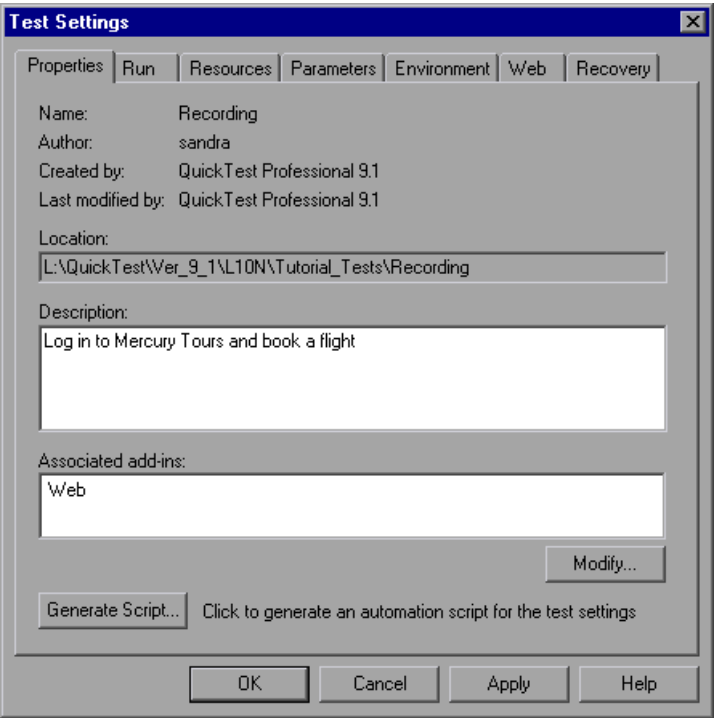
The Test Settings dialog box contains the following tabbed pages:

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 759.
Run	Options for setting the run session preferences. For more information, see “Defining Run Settings for Your Test” on page 763.
Resources	Options for specifying resources you want to associate with your test, such as function libraries stored in VBScript function libraries. For more information, see “Defining Resource Settings for Your Test” on page 767.
Parameters	Options for specifying input and output parameters for your test. For more information, see “Defining Parameters for Your Test” on page 771.
Environment	Options for viewing existing built-in and user-defined environment variables, adding, modifying and saving user-defined environment variables, and selecting the active external environment variables file. For more information, see “Defining Environment Settings for Your Test” on page 774.
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” on page 782.
Recovery	Options for setting how QuickTest recovers from unexpected events and errors that occur in your testing environment during a run session. For more information, see “Defining Recovery Scenario Settings for Your Test” on page 784.

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.

Defining Properties for Your Test

You can use the Properties tab of the Test Settings dialog box (**File > Settings**) 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 Windows user name of the person who created the test.
Created by	Indicates the version of QuickTest used to create the test.

Option	Description
Last modified by	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.
Associated add-ins	Displays the add-ins associated with the test. For more information, see “Associating Add-ins with Your Test” on page 760.
Modify	Enables you to select the add-ins to associate with your test. For more information, see “Modifying Associated Add-Ins” on page 761.
Generate Script	Generates an automation script containing the current test settings. For more information, see “Automating QuickTest Operations” on page 1105, or refer to the <i>QuickTest Automation Reference</i> (Help > QuickTest Professional Help > QuickTest Advanced References > QuickTest Automation).

Associating Add-ins with Your Test

When you open QuickTest, you select the add-ins to load from the Add-in Manager dialog box. You can 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.

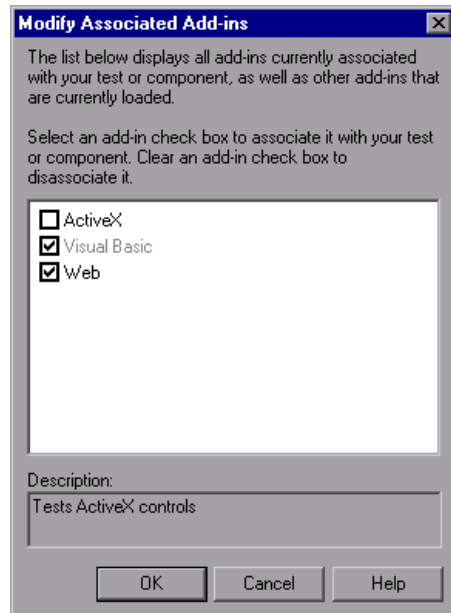
Choosing to associate an add-in with your test instructs QuickTest to check that the associated add-in is loaded each time you open that test.

When you open a test, QuickTest notifies you if an associated add-in is not currently loaded, or if you have loaded add-ins that are not currently associated with your test. This process ensures that your run session will not fail due to unloaded add-ins and reminds you to add required add-ins to the associated add-ins list if you plan to use them with the currently open test.

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 45, “Working with Quality Center.”

Modifying Associated Add-Ins

You can associate or disassociate add-ins with your test in the Modify Associated Add-ins dialog box.



This dialog box lists all the add-ins currently associated with your test, as well as any other add-ins that are currently loaded in QuickTest. Add-ins that are associated with your test but not currently loaded are shown dimmed.

You can select the check boxes for add-ins that you want to associate with your test, or clear the check boxes for add-ins that you do not want to associate with your test.

In the above example:

- Web is loaded and associated with the test.
- ActiveX is loaded, but not associated with the test.
- Visual Basic is associated with the test, but is not loaded.

Note: If a specific add-in is not currently loaded, but you want to associate it with your test, reopen QuickTest and load the add-in from the Add-in Manager. If the Add-in Manager dialog box is not displayed when you open QuickTest, you can choose to display it the next time you open QuickTest. To do so, select **Display Add-in Manager on startup** from the General tab of the Options dialog box.

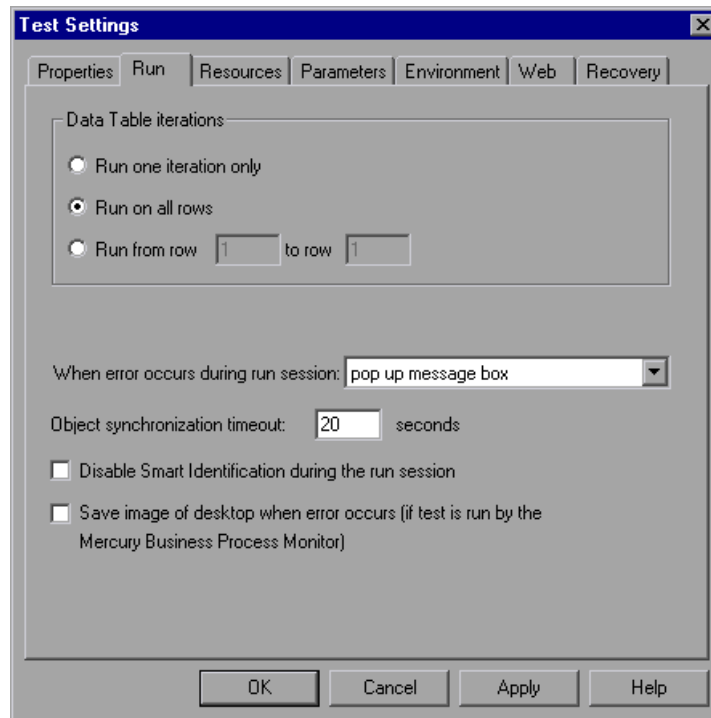
See Chapter 25, “Setting Global Testing Options” for more information on the Options dialog box. For more information on the Add-in Manager, see Chapter 28, “Working with QuickTest Add-Ins.”

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 Reference* (**Help > QuickTest Professional Help > QuickTest Advanced References > QuickTest Automation**).

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 (**File > Settings**) 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 384.

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 on action run properties, see “Setting the Run Properties for an Action” on page 874.

The Run tab includes the following options:

Option	Description
Data Table iterations	Specifies the iterations for the test. Choose an option: <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	Specifies how QuickTest responds to an error during the run session. For more information, see “Specifying the Response to an Error” on page 765.
Object synchronization timeout	Sets the maximum time (in seconds) that QuickTest waits for an object to load before running a step in the test. <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” on page 782.</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 the 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.</p> <p>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 when an error occurs.

QuickTest first performs any recovery scenarios associated with the test, and performs the option selected above only if the associated recovery scenarios do not resolve the error. For more information, see “Defining Recovery Scenario Settings for Your Test” on page 784.

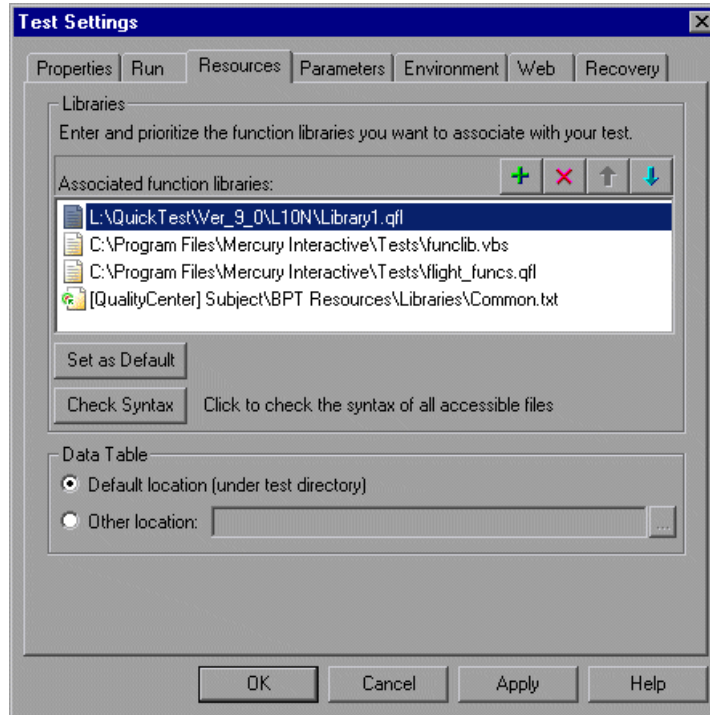
Note: This option replaces the global option found in QuickTest versions 6.0 and earlier. When you open a test created in QuickTest 6.0 and earlier, the **pop up message box** option is automatically selected by default.

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 1105, or refer to the *QuickTest Automation Reference* (**Help > QuickTest Professional Help > QuickTest Advanced References > QuickTest Automation**).

Defining Resource Settings for Your Test

You can use the Resources tab of the Test Settings dialog box (**File > Settings**) to associate specific files with your test, such as VBScript function libraries and Data Table files. You can also set the currently associated function library settings as the default settings for all new tests.



Note: Object repositories are associated with individual action(s) in your test. You can associate an object repository with an action using the Action Properties dialog box (**Edit > Action > Action Properties**) and the Associate Repositories dialog box (**Resources > Associate Repositories**).

The Resources tab in the Test Settings dialog box includes the following option areas:

Option Area	Description
Libraries	Displays the list of function libraries associated with your test. You can add, delete, and prioritize the files. You can also set the default function libraries for new tests. For more information, see “Specifying Associated Function Libraries” on page 769.
Set as Default	Sets the current list of function libraries as the default list to be associated with new tests. Note: The Set as Default option is available for tests only. It is enabled when the setting for this test is different than the default for all tests.
Check Syntax	Verifies whether any of the associated function libraries contain syntax errors that will prevent the test from running properly. Click the Check Syntax button to check the files for syntax errors before finalizing the test. If any syntax errors are found, the Information pane opens listing the files containing syntax errors. Otherwise, an information box opens confirming that the syntax in all of the function libraries is valid. Note: QuickTest checks only the associated function libraries that can be accessed. For example, if an associated function library is stored in a Quality Center project to which you are not currently connected, its syntax will not be checked.
Data Table	Specifies the location of the Data Table to be used in your test: <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. For more information on choosing a Data Table location, see “Editing the Data Table” on page 522. 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 530.

Specifying Associated Function Libraries





The **Associated function libraries** pane of the Resources tab indicates the list of function libraries associated with your test. QuickTest searches these files for the VBScript functions, subroutines, and so forth that are specified in the test.

The order of the function libraries in the list determines the order in which QuickTest searches for a function or subroutine that is called from a step in your test. If there are two functions or subroutines with the same name, QuickTest uses the first one it finds. For more information, see “Working with Associated Function Libraries” on page 1052.


You can enter an associated function library 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 712.


Note: When working with tests, if your function libraries 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 in the Folders tab of the Options dialog box (**Tools > Options**). For more information, see “Setting Folder Testing Options” on page 712.

You can add, delete and prioritize the function libraries associated with your test using the function library control buttons:

Option	Description
	Associates a function library with the test. You can enter the absolute or relative path and filename of the function library, or use the browse button to locate the required file. If the function library contains syntax errors, a message opens stating that your test will fail because of these syntax errors. You can associate files located in Quality Center project folders. For more information, see “Associating Function Libraries in Quality Center Project Folders” on page 770, below.
	Removes an associated function library from the list.
	Assigns a higher priority to the selected function library.
	Assigns a lower priority to the selected function library.

Associating Function Libraries in Quality Center Project Folders

When you are connected to Quality Center and you click the  button, QuickTest adds [QualityCenter], and displays a browse button so that you can locate the Quality Center path.

When not connected to Quality Center, you can add a file located in a Quality Center project folder by holding the SHIFT key and clicking the  button. QuickTest adds [QualityCenter], and you can enter the path. You can also type the entire Quality Center path manually. If you do, you must add a space after [QualityCenter]. For example: [QualityCenter] Subject\Tests.

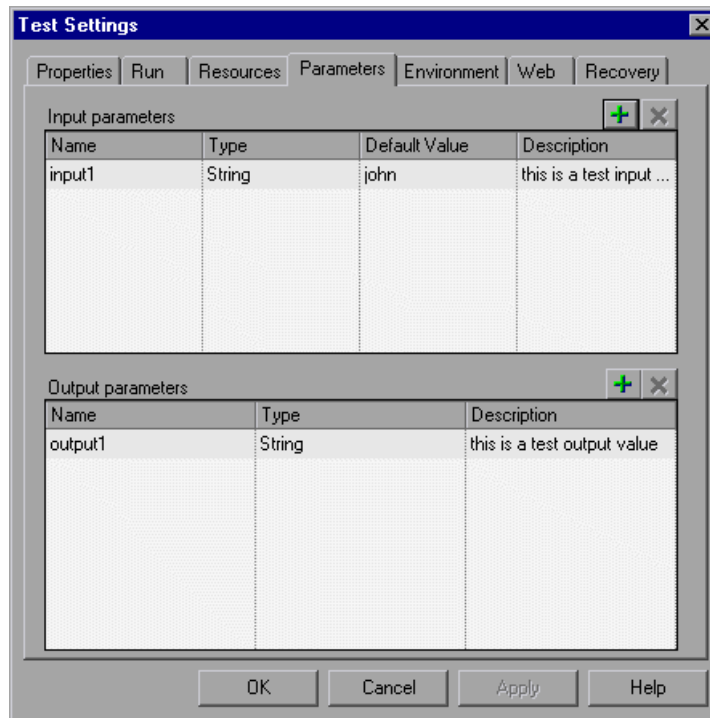
Note: When running a test, QuickTest uses associated function libraries from Quality Center project folders only when you are connected to the corresponding Quality Center project.

For more information on working with Quality Center projects, see Chapter 45, “Working with Quality Center.”

Defining Parameters for Your Test

You use the Parameters tab of the Test Settings dialog box (**File > Settings**) to define input parameters that pass values into your test and output parameters that pass values from your test to external sources. You can also use the Parameters tab to modify or delete existing test parameters.

Test parameters are similar to Action parameters. For information on Action parameters, see “Setting Action Parameters” on page 864.





The Parameters tab contains two parameter lists:

- **Input parameters.** Specifies the parameters that the test can receive values from the source that runs or calls it.
- **Output parameters.** Specifies the parameters that the test can pass to the source that runs or calls it.

You can edit an existing parameter by selecting it in the appropriate list and modifying its details.

You can add and remove input and output parameters for your test using the parameter control buttons:

Option	Description
	<p>Adds a parameter to the appropriate parameter list. Enter a name for the new parameter (case sensitive) and select the parameter type. You can enter a description for the parameter, for example, the purpose of the parameter in the test.</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” on page 772, below.</p> <p>You define test parameters in the same way you define action parameters. For information on defining parameters and parameter types, see “Setting Action Parameters” on page 864.</p>
	Removes the selected parameter from the test.

Defining Default Values for Input Parameters

When a test runs, the actual values used for parameters are generally those sent by the application calling the test (either QuickTest or Quality Center) as described in the table below:

Document Type:	Called From:	Parameter Values Specified In:
Test	QuickTest	Input Parameters tab of the Run dialog box. For more information, see “Running Your Entire Test” on page 609.
Test	Quality Center	Test Run Properties dialog box (Test Lab module). For more information, refer to the <i>Mercury Quality Center User's Guide</i> .

If, when a test runs, a value is not supplied by QuickTest or Quality Center for one or more input parameters, QuickTest uses the default value for the parameter.

When you define a new parameter in the Parameters tab of the Test Settings dialog box, you can specify the default value for the parameter or you can keep the default value that QuickTest assigns for the specified parameter type as follows:

Value Type	QuickTest Default Value
String	Empty string
Boolean	True
Date	The current date
Number	0
Password	Empty string
Any	Empty string

Using Test Parameters in Steps

You can directly access test parameters only when parameterizing the value of a top-level action input parameter or when specifying the storage location for a top-level output parameter. To use values supplied for test parameters in steps within an action, you must pass the test parameter to the action containing the step. For more information, see “Setting Action Parameters” on page 864.

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 Parameters in Steps in the Expert View” on page 377.

Defining Environment Settings for Your Test

The Environment tab of the Test Settings dialog box (**File > Settings**) displays existing built-in and user-defined environment variables. It also enables you to add, modify, or delete internal user-defined environment variables, save the defined variables to an external **.XML** file, and retrieve them from a file.

If you export your user-defined variables to an external **.XML** file, you can then use the exported environment variable file with any other test.

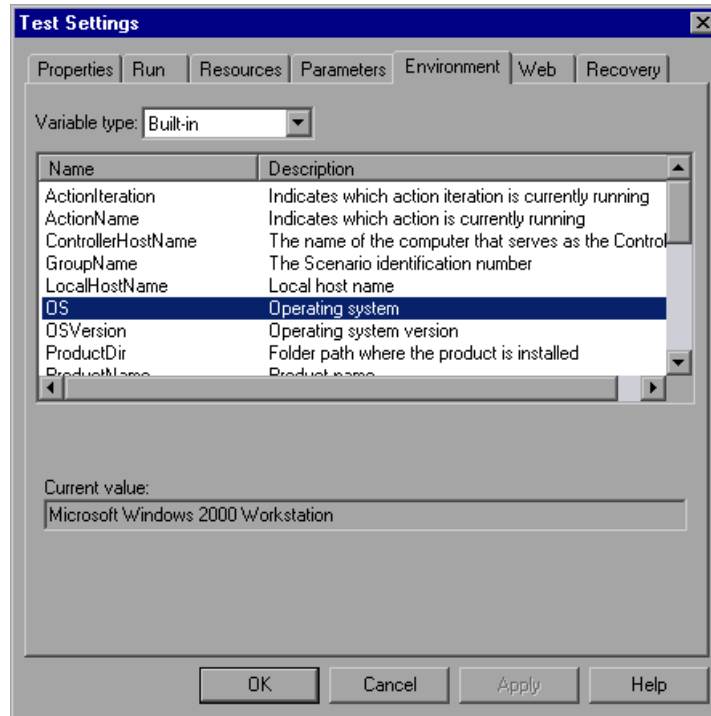
For more information on environment variables and environment parameters, see “Using Environment Variable Parameters” on page 385.

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.

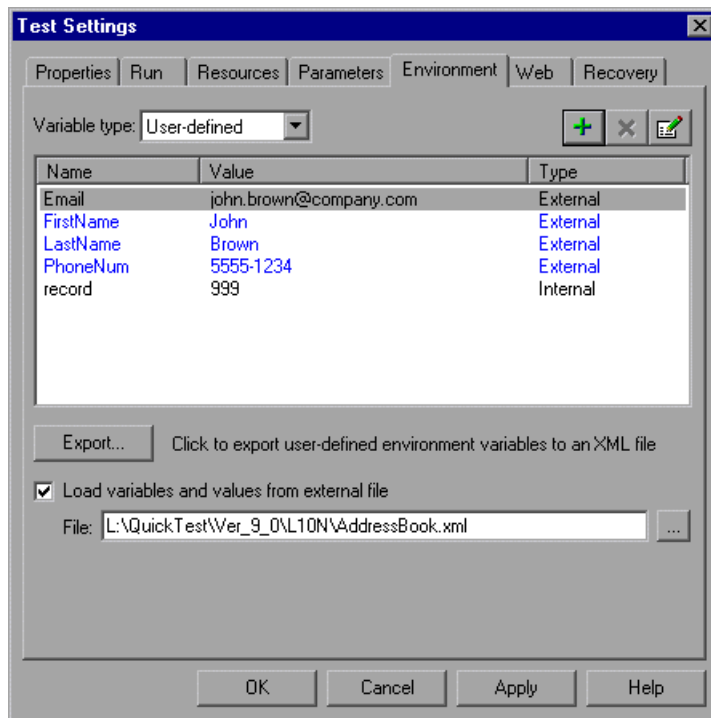


The following information is displayed for built-in environment variables:

- **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.






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 in which they are defined.

The Environment tab provides the following options for user-defined environment variables:

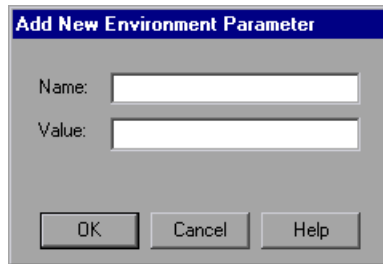
Option	Description
	Enables you to define a new internal environment variable and add it to the list. For more information, see “Adding User-Defined Environment Variables”, below.
	Deletes a selected internal environment variable from the list. Note: After you confirm the deletion of the environment variable, you cannot retrieve it, even if you click Cancel on the Test Settings dialog box.
	Enables you to edit the value of a selected internal environment variable or to view the properties of a selected external environment variable. For more information, see “Viewing and Modifying User-Defined Environment Variables” on page 778.
Export	Exports your user-defined environment variables to an external .XML file for use with other tests. You can then use the exported environment variable file with any test. For more information, see “Exporting and Loading User-Defined Environment Variables” on page 780.
Load variables and values from external file	Loads the variables saved in the .XML file that you specify for use with your test. For more information, see “Exporting and Loading User-Defined Environment Variables” on page 780.

Adding User-Defined Environment Variables

You can add internal user-defined environment variables in the Environment tab of the Test Settings dialog box. Internal environment variables are available only to the test in which they are defined.

To add internal user-defined environment variables:

- 1 In the **Variable type** box of the Environment tab, select **User-defined**.
- 2 Click the **New** button. The Add New Environment Parameter dialog box opens.



- 3 Enter a definition for the variable as follows:
 - **Name.** Enter the name of the variable.
 - **Value.** Enter the value of the variable.
- 4 Click **OK** to save your changes and close the Add New Environment Parameter dialog box. The variable is added to the list (displayed in black) in the Environment tab of the Test Settings dialog box.

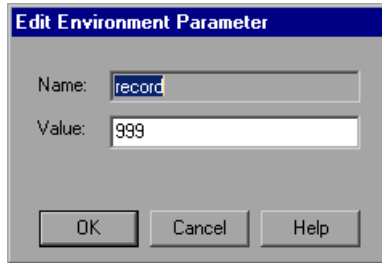
Viewing and Modifying User-Defined Environment Variables

You can edit the values of internal user-defined environment variables in the Environment tab of the Test Settings dialog box. You can also view the properties of external user-defined variables.

You can copy the values of internal and external variables for use in other areas of QuickTest, for example, in the Data Table.

To modify or copy an internal user-defined environment variable:

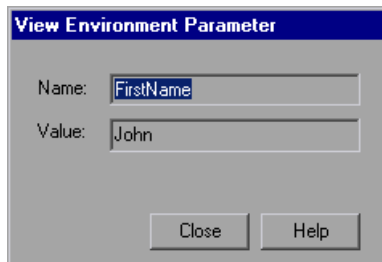
- 1 In the Environment tab of the Test Settings dialog box, double-click the internal variable, or select it and click the **View/Edit Environment Variable** button. The Edit Environment Parameter dialog box opens.



- 2 To modify the value of the variable, enter a different value in the **Value** box.
- 3 To copy the value of the variable to the Clipboard, select the value text, right-click, and 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 dialog box.

To view an external user-defined environment variable:

- 1 In the Environment tab of the Test Settings dialog box, double-click the external variable you want to view, or select it and click the **View/Edit Environment Variable** button. The View Environment Parameter dialog box displays the details of the selected variable.



If the variable has a complex value (a value that cannot be displayed entirely in the **Value** box), you can click the **View/Edit Complex Value** button to view the contents of the value.

- 2 To copy the value of the variable to the Clipboard, select the value text, right-click and 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. You can then use the exported environment variables with any test, by loading them from the file as external user-defined environment variables.

If the file is saved to the file system, its values are loaded each time the test runs. If the file is saved to a Quality Center project, its values are loaded when the test is first loaded. If the values are changed after the test is loaded, the new values will not be used by QuickTest, until the next time the test is loaded.

To export user-defined environment variables:

- 1 In the Environment tab of the Test Settings dialog box, click the **Export** button. The Save Environment Variable File dialog box opens, enabling you to export the current list of user-defined variables and values to an **.XML** file.
- 2 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 dialog box, select **Load variables and values from external file**.
- 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.

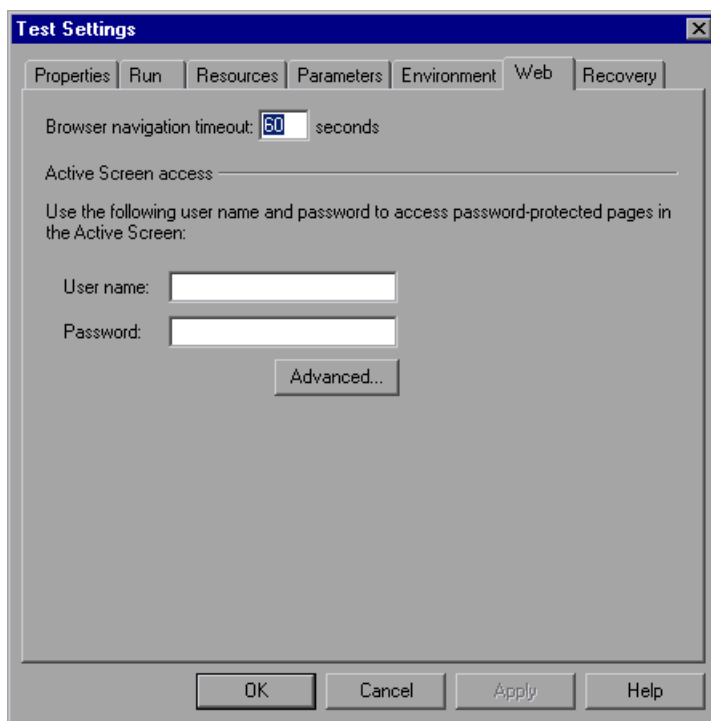
The environment variables loaded from the selected file are displayed in blue in the Environment tab of the Test Settings dialog box.

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 712.

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 385.

Defining Web Settings for Your Test

The Web tab of the Test Settings dialog box (**File > Settings**) provides options for recording and running tests 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 seconds) that QuickTest waits for a Web page to load before running a step in the test.
User name	The user name for password-protected resources that use a standard authentication mechanism. For more information, see “Using the Standard Authentication Mechanism” on page 846.
Password	The password for password-protected resources that use a standard authentication mechanism. For more information, see “Using the Standard Authentication Mechanism” on page 846.
Advanced	Opens the Advanced Authentication dialog box, which enables you to manually log in to your Web site to enable access to password-protected resources that use an advanced authentication mechanism. For more information, see “Using the Standard Authentication Mechanism” on page 846.

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 moving the pointer over an object to open a sub-menu, you may need to modify your Web event configuration to recognize such events. For more information, see Chapter 41, “Configuring Web Event Recording.”

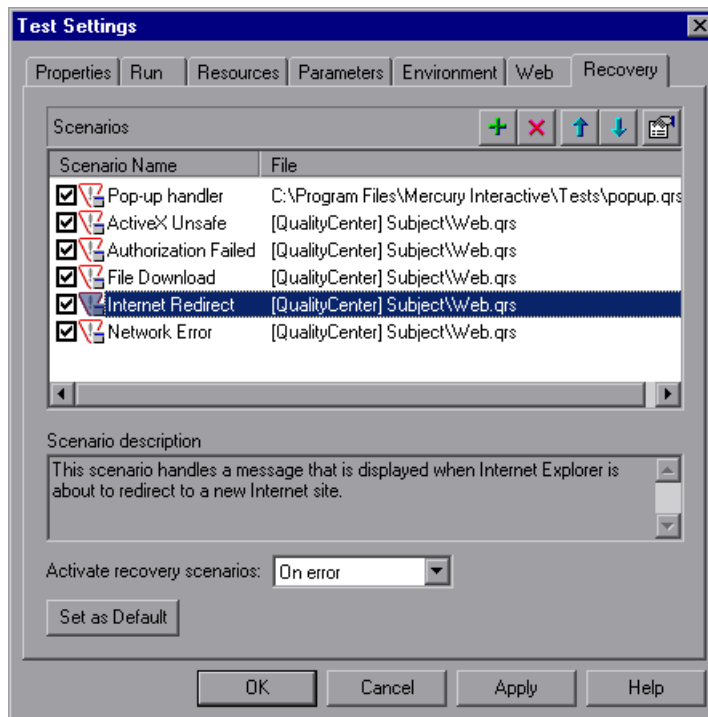
Defining Recovery Scenario Settings for Your Test

The Recovery tab of the Test Settings dialog box (**File > Settings**) displays a list of all recovery scenarios associated with the current test. It also enables you to associate additional recovery scenarios with the test, remove scenarios from the test, change the order in which they are applied to the run session, and view a read-only summary of each scenario.

You can enable or disable specific scenarios or the entire recovery mechanism for the test.

If you are working with tests, you can also specify that the current list of scenarios be used as the default for all new tests.

For more information on recovery scenarios, see Chapter 32, “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. You can add, delete, and prioritize the scenarios in the list, and you can edit the file path for a selected file. For more information, see Chapter 26, “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.</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 in the Folders tab of the Options dialog box (**Tools > Options**). For more information, see “Setting Folder Testing Options” on page 712.

Specifying Associated Recovery Scenarios

You can select or clear the check box next to each scenario to enable or disable it for the current test.






You can also edit the recovery scenario file path by clicking the path once to highlight it, and then clicking it again to enter edit mode. For example, you may want to modify an absolute file path to be a relative file path. If you modify a recovery scenario file path, ensure that the recovery scenario exists in the new path location before running your test.

Scenarios are indicated by the following icons:

Icon	Description
	Indicates that the recovery scenario is triggered by a specific pop-up window in an open application during the run session.
	Indicates that the recovery scenario is triggered when the property values of an object in an application match specified values.
	Indicates that the recovery scenario is triggered when a step in the test does not run successfully.
	Indicates that the recovery scenario is triggered when a specified application fails during the run session.
	Indicates that the recovery scenario is no longer available for the test—possibly because the recovery file has been renamed or moved, or can no longer be accessed by QuickTest. When an associated recovery file is not available during a run session, a message is displayed in the test results.

Note: The default recovery scenarios provided with QuickTest are installed in your QuickTest installation folder. The paths specifying the default recovery scenarios in the Recovery tab use an environment variable (%ProductDir%) in the file path. This enables QuickTest to locate these recovery scenarios when tests associated with them are run on different computers or by different 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 using the recovery scenario file control buttons:

Option	Description
	Opens the Add Recovery Scenario dialog box, which enables you to associate one or more recovery scenarios with the test. For more information, see “Adding Recovery Scenarios to Your Test” on page 935.
	Removes the selected recovery scenario from the test.
	Moves the selected scenario up in the list, giving it a higher priority.
	Moves the selected scenario down in the list, giving it a lower priority.
	Displays summary properties for the selected recovery scenario in read-only format. For more information, see “Viewing Recovery Scenario Properties” on page 938.

27

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.

This chapter describes:	On page:
About Setting Record and Run Options	790
Using the Record and Run Settings Dialog Box	790
Setting Web Record and Run Options	793
Setting Windows Applications Record and Run Options	796
Using Environment Variables to Specify the Record and Run Details for Your Test	802

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 Visual Basic, ActiveX, and terminal emulators, you 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 715.

The Record and Run Settings dialog box can contain the following tabs:

Tab Heading	Subject
Web	Options for testing Web sites and applications. Note: The Web tab is available only when Web support is installed and loaded. For more information, see “Setting Web Record and Run Options” on page 793.
Windows Applications	Options for testing standard Windows applications. Note: This tab is always available and applies to all Standard Windows, Visual Basic, and ActiveX applications. (Appropriate add-ins must also be loaded.) For more information, see “Setting Windows Applications Record and Run Options” on page 796.

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 details, those values override the values in the Record and Run dialog box. For more information, see “Using Environment Variables to Specify the Record and Run Details for Your Test” on page 802.

To set record and run options:



- 1** Click the **Record** button or choose **Automation > 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.

Guidelines for Modifying Record and Run Settings

After you set the record and run settings for a test, the Record and Run settings dialog box will not open the next time you record operations in that test. If needed, you open the Record and Run Settings dialog box by choosing **Automation > Record and Run Settings**.

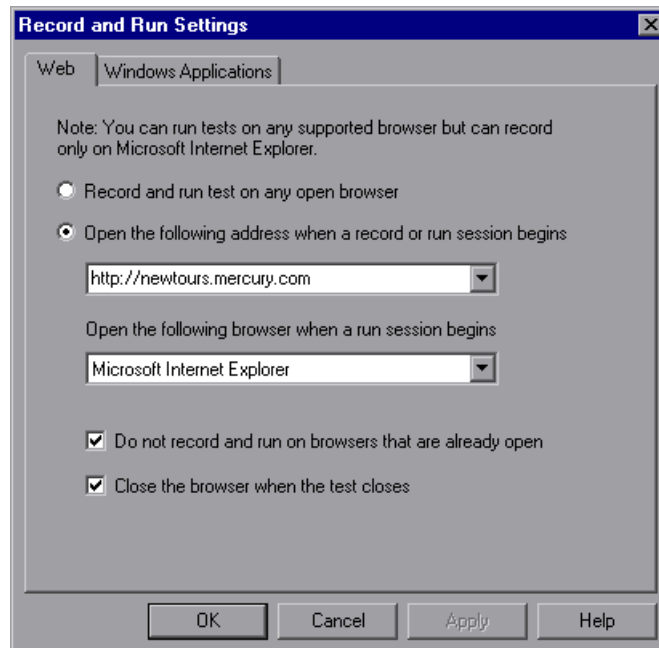
You should set or modify your record and run preferences in the following scenarios:

- 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. (The Web tab is available only when the corresponding Web Add-in is installed and loaded. Some external Web-based QuickTest add-ins may also use this tab. For more information, refer to the relevant QuickTest add-in documentation.)



Note: You can record tests only on Microsoft Internet Explorer browsers. You can run tests on any supported browser (Microsoft Internet Explorer, Netscape Browser, Mozilla Firefox). For information on supported browser versions, refer to the *QuickTest Professional Readme*.

The Web tab includes the following options:

Option	Description
Record and run test on any open browser	<p>Instructs QuickTest to record on any open Microsoft Internet Explorer browser and run on any open supported Web browser (refer to the <i>QuickTest Professional Readme</i> for information on supported browsers).</p> <p>Note: You must open the Web browser after you open QuickTest and select this option.</p> <p>Tip: You can instruct QuickTest to ignore selected browsers that are open during the record and run session. For more information, see “Setting Web Testing Options” on page 739.</p>
Open the following address 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. When recording a test, the address is opened in a Microsoft Internet Explorer browser. When running a test, the address is opened in the browser type specified in the Open the following browser when a run session begins box.</p> <p>Note: If you define a value for the URL_ENV environment variable, that value overrides the value specified here during a run session. For more information, see “Using Environment Variables to Specify the Record and Run Details for Your Test” on page 802.</p>

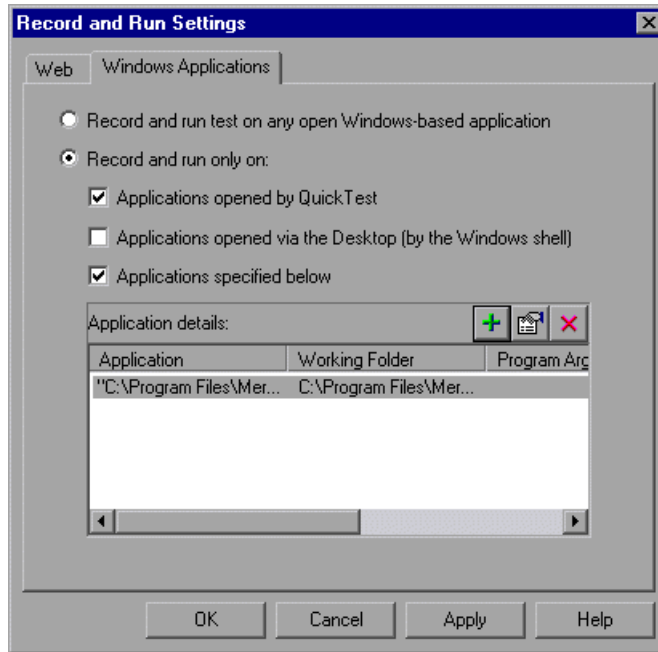
Option	Description
Open the following browser when a run session begins	<p>Instructs QuickTest to open the specified browser type when running a test:</p> <ul style="list-style-type: none"> • Microsoft Internet Explorer • Netscape Browser 8.x • Mozilla Firefox 1.5 <p>Notes: Only those browsers currently installed on your computer are available in the list.</p> <p>If you define a value for the BROWSER_ENV environment variable, that value overrides the value specified here during a run session. For more information, see “Using Environment Variables to Specify the Record and Run Details for Your Test” on page 802.</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 (and prior to opening QuickTest). Selecting this option also prevents you from viewing the properties of these browsers using the Object Spy.</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 users of applications with embedded Web browser controls:

To record and run tests on 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 the application is opened after QuickTest, and start recording. For more information on browser settings, see “Recording a Test” on page 88.

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.



Note: Some external Windows-based QuickTest add-ins may also use this tab. For more information, refer to the relevant QuickTest add-in documentation.

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 opened while recording your test (including e-mail applications, file management applications, and so forth). QuickTest only records and runs on applications that have a user interface, and it does not matter how the applications are opened (as child processes of Windows Explorer, child processes of QuickTest, and so forth).




Selecting **Record and run only on** restricts record and run operations to selected applications. Additionally, you can configure whether QuickTest should open these applications for you at the beginning of a record or run session.

Tip: If you do not want to record on any standard Windows applications, select only the **Applications specified below** check box, and ensure that there are no applications listed in the **Application details** area.

The Windows Applications tab includes the following options:

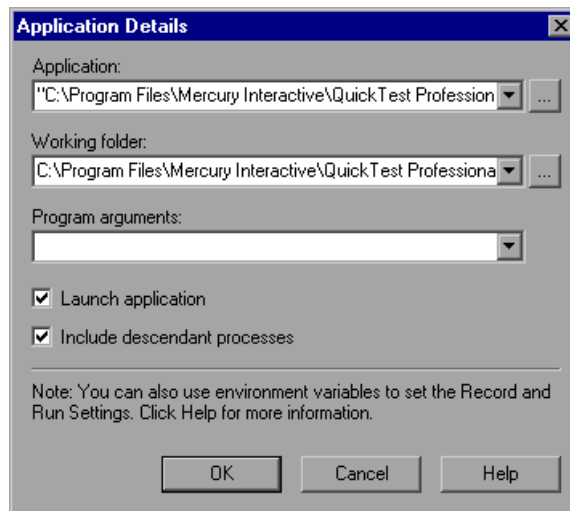
Option	Description
Record and run test on any open Windows-based application	<p>Instructs QuickTest to record and run on any open Windows-based application.</p> <p>Note: Make sure that all the applications on which you want to record are currently closed. QuickTest can record on the applications that you open manually only after 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
Record and run only on	<p>Instructs QuickTest to restrict its record and run operations to one or more of the following options:</p> <ul style="list-style-type: none"> • Applications opened by QuickTest: This option records and runs only on applications invoked by QuickTest (as child processes of QuickTest). For example, applications opened during a record or run session using a SystemUtil.Run statement, or using a statement such as Set shell = createobject("wscript.shell"); shell.run "notepad". • Applications opened via the Desktop (by the Windows shell): This option records and runs only on applications that are opened via the Windows Desktop. For example, applications opened from the Windows Start menu, by double-clicking executable files in the Windows Explorer, by double-clicking a shortcut on the Windows Desktop, or by clicking icons on the Quick Launch bar. • Applications specified below: This option records and runs only on applications listed in the Application details area. This is the recommended option to use. <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 after 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>
Application details	<p>Lists the details of the applications on which to record and run the test. For more information on the details displayed, see “Adding or Editing Application Details” on page 799.</p>

Option	Description
	Opens the Application Details dialog box to enable you to add an application to the application list. You can add up to ten applications. For more information, see “Adding or Editing Application Details”, below.
	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.
	Deletes the selected application from the application list.

Adding or Editing Application Details

When you click the **Add** or **Edit** buttons in the Windows Applications tab of the Record and Run dialog box, the Application Details dialog box opens.



You can add up to ten applications to the application list displayed in the Windows Applications tab, and you can edit an existing application in the list. You can also select whether to launch the selected applications when the session starts, and whether to record and run on the application’s descendant processes.

The details entered in the Application Details dialog box are displayed as a single line for each application in the **Application details** area of the Windows Applications tab.

You can specify the following details for the application in the Application Details dialog box:

Option	Description
Application	<p>Instructs QuickTest to record and run on 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 folder 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 712.</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>The full path name is used to launch an application only when Launch application is selected. QuickTest records and runs on any application with the specified executable file name. For example, if you specify C:\Windows\notepad.exe, QuickTest records on a Notepad application invoked from any folder.</p> <p>Tip: You can specify a document or other file associated in the file system with an application, for example, c:\tmp\1.a.txt. In this case, QuickTest automatically opens the specified file in the associated application (Notepad in this example). If you use this option, QuickTest ignores any defined program arguments.</p>
Working folder	<p>Optional. Specifies the current working folder for the application. The current working folder is used by the application to search for related files. If a working folder is not specified, the executable folder is used as the working folder.</p> <p>Note: This parameter is used only when Launch application is selected. If Launch application is not selected, its value has no effect.</p>

Option	Description
Program arguments	<p>Optional. Instructs QuickTest to open the application using the specified command line arguments.</p> <p>Note: This parameter is used only when Launch application is selected. If Launch application is not selected, its value has no effect.</p>
Launch application	Instructs QuickTest whether to launch the selected application when the record and run session begins. By default, this option is selected.
Include descendant processes	<p>Instructs QuickTest whether to record and run on processes created by the specified application during the record and run session. For example, a process that is used only as a launcher may create another process that actually provides the application functionality. This descendant process must therefore be included when recording or running tests on this application, otherwise the functionality will not be recorded, or the run session will fail.</p> <p>By default, this option is selected.</p>

Using Environment Variables to Specify the Record and Run Details for Your Test

You can use special, predefined environment variables to specify the applications or browsers 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 or browser details, the environment variable values override any values that were added using these areas of the Record and Run Settings dialog box.

Note: If you select the option to Record and Run on any application or browser (the upper radio button in each tab of the Record and Run Settings dialog box), QuickTest ignores any defined record and run 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 or browsers to open when you begin recording—assuming that the option to open an application or browser 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, and the third check box in the Windows Applications tab.)

To use record and run 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 or browser details, and you select the option to open an application or browser when the record session begins (the lower radio button in each tab of the Record and Run Settings dialog box), QuickTest ignores 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 or browser detail you want to set using the appropriate variable name.

For a list of available record and run environment variables, see “Defining Record and Run 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 385.

- 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. If you are running a Windows application, also select the third check box.
- 5 Run the test. QuickTest uses the environment values to determine which application(s) or browsers to open at the beginning of the run session, and on which processes to record.

Defining Record and Run Environment Variables

To use environment variables to specify the applications or browsers 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. For example, Microsoft Internet Explorer, Netscape Browser, or Mozilla Firefox. Possible values: IE, NS8, FF15
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 tests:

Option	Variable Names	Description
Application	EXE_ENV_1 ... EXE_ENV_10	The executable files on which QuickTest records operations when record and run sessions begin. You can specify up to ten executable files.
Working folder	DIR_ENV_1 ... DIR_ENV_10	The folder to which the corresponding executable file refers (for each corresponding application).
Program arguments	ARGS_ENV_1 ... ARGS_ENV_10	The command line arguments to be used for the specified application (for each corresponding application).

Option	Variable Names	Description
Launch application	LNCH_ENV_1 ... LNCH_ENV_10	Whether to open the application when starting the record and run session (for each corresponding application). Possible values: 0 (do not launch the application) 1 (launch the application)
Include descendant processes	CHLD_ENV_1 ... CHLD_ENV_10	Whether to record and run on processes created by the application during the record and run session (for each corresponding application). Possible values: 0 (do not record on descendant processes) 1 (record descendant processes)

Index

A

About QuickTest Professional window 57
access permissions
 required for Quality Center 13
 required to run QuickTest 13
accessibility. *See* Web content accessibility
action calls
 iterations 874
 missing 505
 parameter values 875
 properties 873
 run properties 874
action data sheets 475, 520
action Data Table parameters 384
Action List 477
action parameters 366, 374, 499, 868
 guidelines 871
 setting options 376
 storing output values 416, 426
Action tab, Data Table 475
Action toolbar, Keyword View 45, 477
action values, sharing
 using Dictionary objects 879
 using environment variables 879
 via the global Data Table 878
ActionIteration, environment variable 391
actions 471, 855
 adding to Keyword View 120
 calling using basic syntax 880
 creating 478
 deleting 499
 diagram 472, 856, 857
 external 474
 guidelines for working with 480

 inserting
 call to 861
 copy of 858
 existing 856
 mapping calls to missing 508
 missing calls to 508
 multiple, in tests 473
 nesting 493, 868
 non-reusable 474
 overview 472, 856
 parameterization data, location 493
 parameters. *See* action parameters
 removing 499
 removing calls to missing 512
 renaming 497
 reusable 474
 running from a step 613
 setting parameters 487
 setting properties 482
 sharing values 878
 using Dictionary objects 879
 using environment variables 879
 via the global Data Table 878
 splitting 495
 syntax 880
 syntax for parameters 881
 syntax for storing return values 882
 template 503
 test flow 477
 values. *See* action values, sharing
Active Screen 219
 advanced authentication 848
 defining capture settings 717
 defining Web settings 721
 increasing/decreasing saved
 information 1338
 saving and deleting files 105

Index

- Test Settings dialog box, Web tab 847
 - updating 223
 - Active Screen dialog box 846, 848
 - Active Server Page technology 1334
 - Add Object to Object Repository dialog box 192
 - Add Repository Parameter dialog box 1136
 - Add Schema dialog box, XML checkpoint 341
 - Add Synchronization Point dialog box 581
 - Add/Remove dialog box, object identification 948, 964
 - Add/Remove Properties dialog box 177
 - Add-in Manager dialog box 811
 - add-ins
 - about 809
 - associated and loaded 814
 - associating with a QuickTest test in Quality Center 1277
 - associating with a test 760
 - loading 811
 - tips for working with 814
 - advanced authentication, Active Screen 848
 - Advanced Web Options dialog box 748
 - Advanced Windows Applications Options dialog box 729
 - Agent, Remote 1297
 - Allow other Mercury tools to run tests option 1296
 - analog recording 93, 97
 - analyzing run results. *See* run results
 - API, using Windows 1031
 - application areas
 - recovery scenarios, removing 938
 - Application crash trigger 907
 - Application Details dialog box 799
 - Application Management, integrating with QuickTest 1313
 - application, sample 14
 - applications
 - adding 799
 - closing 1017
 - editing 799
 - running 1017
 - testing localized versions 1340
 - ARGS_ENV_1 variable 804
 - arguments, defining 1060
 - ASCII 522
 - ASP files 1334
 - Assignment column, Keyword View 116
 - assistive properties, configuring 946
 - Associate Repositories dialog box 214
 - associating
 - add-ins with a test 760
 - add-ins with test created in Quality Center 1280
 - add-ins, using the Add-in Manager 814
 - function libraries 1052, 1054
 - object repositories with actions 488
 - shared object repositories 214
 - attribute/<property name> notation 1030
 - authentication
 - connecting to Quality Center 1261
 - for Active Screen 848
 - auto-expand VBScript syntax 1235
 - automation
 - Application object 1110
 - definition 1106
 - development environment 1108
 - generating scripts for a test 760
 - language 1108
 - object model 1105
 - object repository 1148
 - type library 1108
 - Automation Engineer, role in Business Process Testing 1305
 - Automation toolbar, QuickTest window 20, 43
- ## B
- backslash (\) 357
 - Basic event recording configuration level 1209
 - behavior, DHTML 1217
 - Bitmap Checkpoint Properties dialog box 286
 - bitmap checkpoints 285
 - analyzing results for 658
 - creating 286
 - modifying 293

- bookmarks 988
- breakpoints
 - about 596
 - deleting 599
 - disabling/enabling 598
 - setting 597
 - using in Keyword View 148
- Browser Control Utility 852
- Browser Details dialog box 742
- BROWSER_ENV variable 804
- browsers
 - ignoring 741
 - supported 820
- bubbling 1218
- built-in environment variables 390, 774
- business analyst
 - role in Business Process Testing 1304
- business components, overview 12
- Business Process Monitor, integrating with QuickTest 1313
- Business Process Testing 1303
 - roles 1304
 - workflow 1307
- business process tests 1308
 - overview 12
 - running 1311

C

- calculations
 - in function libraries 1021
 - in the Expert View 1021
- Call to WinRunner Function dialog box 1254
- Call to WinRunner Test dialog box 1250
- Cell Identification tab, Database Checkpoint Properties dialog box 310
- CGI scripts 1333
- character set support, Unicode 4
- Check In command 1287, 1290
- Check Out command 1288
- Checkpoint Properties dialog box
 - for checking databases 238
 - for checking objects 277

- checkpoints
 - about 225
 - adding 227
 - bitmaps 285
 - databases 297
 - definition 101, 225
 - images 281
 - in Expert View 978
 - modifying 281, 283
 - objects 274
 - pages 829
 - parameterizing 400
 - standard, for checking text 269
 - supported for Web objects 818
 - tables 233, 234, 238
 - text 255, 257
 - types 228
 - using formulas 535
 - Web content accessibility 749, 841
 - XML 313
- CHLD_ENV_1 variable 804
- Close method 1017
- closing application process 918, 922, 1017
- CMDLINE_ENV variable 804
- collection, properties. *See* programmatic descriptions
- collections, of virtual objects 885
- colors
 - setting in Keyword View 143
 - setting in Object Repository Comparison Tool 1195
 - setting in Object Repository Merge Tool 1164
- columns, displaying in Keyword View 141
- COM 1334
- command line options
 - deleting test results using 691
 - Domain 692
 - FromDate 692
 - Log 692
 - MinSize 693
 - Name 693
 - Password 694
 - Project 694
 - Recursive 694
 - Server 695

Index

- Silent 695
- Test 695
- UntilDate 696
- User 696
- Comment column, Keyword View 116
- comments
 - components 136
 - function libraries 1019
 - the Expert View 1019
 - the Keyword View 578
- compact view, Object Repository window 162
- comparing
 - shared object repositories 1187
- Completing the Recovery Scenario Wizard
 - screen 929
- complex value 349
- component parameters 129
- component resources, missing 505
- components
 - debugging 587
 - pausing runs 595
 - run results. *See* run results
 - steps, adding 120
 - steps, deleting 139
 - steps, managing 137
 - steps, moving 137
- conditional statements 560
 - using in Keyword View 148
- configuration levels, event recording
 - standard 1209
- configuration levels, Web event recording
 - customizing 1211
- Configure Text Selection dialog box 261
- Configure value area 347
- configuring values 345
- conflict resolution
 - in merged object repository 1178
 - settings, Object Repository Merge Tool 1161
- connecting QuickTest to Quality Center 1261
- connection string, specifying for database checkpoints 302
- Constant Value Options button 349
- Constant Value Options dialog box 349
- constant value, defining 345
- content property check, on databases 298
- ControllerHostName, environment variable 391
- conventions, typographical xxiv
- cookies 1333
- creation time identifier. *See* ordinal identifier
- CreationTime property, using to identify an object 955
- currencies, setting custom format 529
- Custom Active Screen Capture Settings
 - dialog box 717
- custom number format, setting 529
- custom objects, mapping 969
- custom Web event recording configuration 1211
 - adding listening events 1215
 - adding objects to the list 1214
 - deleting objects from the list 1215
 - loading configuration files 1226
 - saving configuration files 1226
 - specifying listening criteria 1217
- Custom Web Event Recording Configuration
 - dialog box 1211, 1222

D

- Data Driver 403
- Data menu commands, Data Table 527
- data sheets
 - action 520
 - Global 520
 - global/action, choosing 475
 - local 520
- Data Table 21, 30, 517
 - action data sheets 520
 - Action tab 475
 - Data menu commands 527
 - data sheets 520
 - design-time 518
 - Edit menu commands 526
 - editing tables 522
 - File menu commands 525
 - Format menu commands 529
 - Global tab 475

- importing data, in various formats 522
- iteration options for individual tests 764
- local data sheets 520
- location 521
- menu commands, using 524
- parameters, setting options for 381
- run-time 518
- saving 521
- scripting functions, using 538
- Sheet menu commands 526
- specifications 523
- storing output values 417, 427
- table columns 379
- table rows 379
- using formulas 534
- using with Quality Center 530
- viewing results 678
- worksheet functions 534
- Database Checkpoint Properties dialog box 303
 - Cell Identification tab 310
 - Expected Data tab 307
 - Settings tab 308
- database checkpoints 297
 - about 297
 - analyzing results 656
 - general information 305
 - modifying 312
 - specifying cell identification settings 310
 - specifying cells 305
 - specifying expected data 307
 - specifying value type 308
- Database Output Value Properties dialog box 451
- database output values 414, 451
- Database Query wizard 299
- database values, outputting 450
- databases
 - connection string 302
 - creating a query in ODBC / Microsoft Query 533
 - creating a query with Microsoft Query / SQL statement 301
 - creating checkpoints for 298
 - manually defining an SQL statement 299
 - result set 298
 - Specify SQL statement screen 302
- data-driven test 366, 417
- dates, setting custom format 529
- Debug toolbar, QuickTest window 20, 43
- Debug Viewer 30, 600
- debugging
 - breakpoints
 - deleting 599
 - disabling/enabling 598
 - setting 597
 - components 587
 - function libraries 587, 1049
 - pausing runs 595
 - Run to Step 593
 - Start from Step 593
 - tests 587
 - tests, an example 603
- default object identification settings 958
- default optional steps 627
- default parameter definition 348, 351
- default properties, modifying 61, 149
- defects, reporting 697
 - automatically during test 698
 - from Test Results 697
- Define Object Filter dialog box 197
- deleting
 - actions 499
 - breakpoints 599
 - objects from the object repository 205
 - repository parameters 1139
 - test results 689
- description, test objects 65
 - See also* test objects
- descriptive programming. *See* programmatic descriptions
- design-time Data Table 518
- development environment 1108
- DHTML behavior 1217
- Dictionary object 879
- difference types
 - Object Repository Comparison Tool 1194

- Dim statement, in the Expert View and function libraries 999
- DIR_ENV_1 variable 804
- disconnecting from Quality Center 1268
- disk space, saving 1341
- display area
 - Script Editor 1093
- Do...Loop statement, in the Expert View and function libraries 1023
- docked panes 36
- Documentation Only option 146
- documentation updates xxiii
- documenting a function 1066
- Domain command line option 692
- DOS commands, run within tests 1031
- dynamic Web content 1332
- dynamically generated URLs and Web pages 1333

E

- Edit menu commands, Data Table 526
- Edit Schema dialog box, XML checkpoint 341
- Edit toolbar, QuickTest window 43
- Edit XML dialog box, XML checkpoint 333
- Editor Options dialog box 1232
- embedded Web browser controls 824
- encoding passwords 132
- End Transaction button 44
- End Transaction dialog box 1322
- environment variables 385, 774
 - built-in 390, 774
 - files, with Quality Center 389
 - record and run 804
 - predefined variable names 804
 - understanding 802
 - storing output values 417, 428
 - types 386
- environment variables, user-defined 778
 - exporting 780
 - external 387
 - internal 386
 - modifying 778
 - viewing 778
- error behavior options for tests 764

- errors in VBScript syntax 1003
- event recording configuration *See* Web event recording configuration
- Excel formulas
 - for parameterizing values 535
 - in checkpoints 535
 - in the Data Table 534
- Excel. *See* Microsoft Excel
- EXE_ENV_1 variable 804
- ExecuteFile function 1080
- ExecuteFile statement 1053
- EXEPATH_ENV variable 804
- Exist property 1333
- Exist statement 583
- existing actions, inserting 856
- Expert View 973, 1331
 - about 975
 - basic action syntax 880
 - checkpoints 978
 - closing applications 1017
 - customizing appearance of 1231
 - finding text 990
 - general customization options 1232
 - highlighting elements 1236
 - replacing text 992
 - running applications 1017
 - syntax for action parameters 881
 - syntax for action return values 882
 - understanding parameters 979
- Export to HTML File dialog box 651
- exporting
 - local objects to object repository file 217
 - object repository to XML file 1146
 - tests to zip files 107
- expressions, using in the Expert View and function libraries 995
- eXtensible Markup Language (XML) 1335
- external action
 - data location 493
 - definition 474
- external functions, executing from script 1080
- external user-defined environment variables 387

F

- FAQs 1329
- File menu commands, Data Table 525
- File toolbar, QuickTest window 21
- filter
 - defining for objects 197
- Filter dialog box
 - Object Repository Comparison Tool 1200
 - Object Repository Merge Tool 1180
- Filter Images Check dialog box 835, 839
- Filter Links Check dialog box 835, 837
- filter properties (Smart Identification) 959
- filtering
 - hypertext links 837
 - image sources 839
 - objects in Object Repository window 162
 - repositories in Object Repository Comparison Tool 1200
 - target repository 1180
- Find & Replace dialog box, object repositories 206
- Find dialog box
 - Expert View 990
 - Object Repository Comparison Tool 1202
 - Object Repository Merge Tool 1181
- floating panes 37
- Flow pane 1088
- fonts, setting in Keyword View 143
- For...Each statement, in the Expert View and function libraries 1023
- For...Next statement, in the Expert View and function libraries 1022
- Format menu commands, Data Table 529
- formulas
 - for parameterizing values 535
 - in checkpoints 535
 - in the Data Table 534
- FromDate command line option 692
- full view, Object Repository window 162
- function arguments, passing parameters from QuickTest to WinRunner 1257

- Function Definition Generator 1060
 - about 1056
 - defining a function 1060
 - documenting a function 1066
 - opening 1058
 - previewing function code 1068
 - registering a function 1061
- function libraries 1037
 - about 10
 - associating current 1054
 - associating with tests in the Script Editor 1101
 - closing in the Script Editor 1104
 - creating 1041
 - creating in the Script Editor 1101
 - customizing appearance of 1231
 - customizing general options 1232
 - debugging 587, 1049
 - description 26
 - editing 1047
 - editing in the Script Editor 1102
 - finding text 990
 - general options 1232
 - highlighting elements 1236
 - managing 1040
 - modifying associated 1054
 - navigating 1046
 - opening 1041, 1051
 - opening in the Script Editor 1099
 - pausing runs 595
 - properties 1091
 - read-only, editing 1049
 - replacing text 992
 - saving 1044
 - saving in the Script Editor 1103
 - specifying for a test 767
 - working with 1099
 - working with associated 1052
- functions
 - code, finalizing 1069
 - code, inserting 1069
 - user-defined 1037

G

- general options 1232
- Generate Script option 1111
- GetROProperty method 1028
- Global data sheet 475, 520
- global Data Table parameter 384
- global testing options 707
- global/action data sheets, choosing 475
- Go To dialog box 986
- GroupName, environment variable 391
- guidelines
 - user-defined functions 1077

H

- handler 1217
- High event recording configuration level 1209
- HTML Source dialog box 833
- HTML Tags dialog box 833
- HTML verification 833
- hypertext links, filtering 837

I

- If...Then...Else statement, in Expert View and function libraries 1025
- ignore browsers list 741
 - adding browser 742
 - modifying browser 743
 - removing browser 744
- Image Checkpoint Properties dialog box 281
- image checkpoints
 - comparing image contents 282
 - editing the property value 282
- image sources, filtering, for page checkpoints 839
- importing
 - object repository from XML file 1145
 - tests from zip files 108
- index identifier. *See* ordinal identifier
- Index property
 - programmatic descriptions 1014
 - using to identify an object 953
- Information Pane 20, 28
- initialization scripts 1107

- Insert New Action dialog box 479
- Insert Report dialog box 575
- Insert toolbar, QuickTest window 44
- IntelliSense 980, 1234
- internal user-defined environment variables 386
- Item cell 122
- Item column, Keyword View 115
- Item list 123
- item, selecting
 - from Item list 123
 - from shared object repository 123
 - from your application 126
- iterations 378, 874
 - options for individual tests 764

J

- JavaScript 1108

K

- key assignments
 - in Expert View 1238
 - in function libraries 1238
- key column 248, 311
- keyboard shortcuts
 - in Expert View 1238
 - in function libraries 1238
 - in Keyword View 140
- Keyword View 23, 111, 113
 - columns, description of 115
 - columns, displaying 141
 - display options 141
 - fonts and colors 143
 - keyboard keys 140
 - steps, adding 120
 - steps, adding after block 135
 - steps, deleting 139
 - steps, modifying 136
- Keyword View tab 23
- Knowledge Base xxii

L

- language 1108
- language support, Unicode 4
- layout
 - customizing QuickTest window 31
 - moving panes 31
 - moving tabs 31
 - restoring default 39
- learning objects 1131
- license information 14
- LNCH_ENV_1 variable 804
- loaded and associated add-ins 814
- loading QuickTest add-ins 811
- LoadRunner, integrating with QuickTest 1313
- local data sheet. *See* action data sheets
- local object repositories 151, 154
 - copying objects to 164
 - merging 1168
- local objects, exporting to object repository file 217
- local parameter 129
- local test 474
- LocalHostName, environment variable 391
- localization 385, 521
- localized applications, testing 1340
- Locate Missing Actions dialog box 508, 512
- location identifier. *See* ordinal identifier
- Location property, using to identify an object 954
- Log command line option 692
- loop statements 566
 - using in Keyword View 148
- low-level recording 93, 100, 1330

M

- Manage Repository Parameters dialog box 1134
- mandatory properties, configuring 946
- manual steps 136
- manual tests 146
- Map Shared Object Repository Parameters dialog box 185

- mapping
 - calls to missing actions 508
 - custom objects 969
 - missing actions 508
 - repository parameters 185
 - unmapped object repositories 514
 - unmapped repository parameters 515
- mathematical formulas, in the Data Table 534
- Medium event recording configuration level 1209
- menu bar, QuickTest window 20
- Mercury Application Management, integrating with QuickTest 1313
- Mercury Best Practices xxii
- Mercury Customer Support Web site xxii
- Mercury Home Page xxii
- Mercury Quality Center. *See* Quality Center
- Mercury Tours, sample application 14
- merging
 - local object repositories 1168
 - shared object repositories 1151
- messages
 - displaying during the run session 577
 - generating 575
 - sending to test results 575
- meta tags 1334
- methods
 - adding new or changing behavior of 1072
 - run-time objects 1029
 - user-defined 1072
 - viewing test objects 61
- Microsoft Excel 522, 534
- Microsoft Internet Explorer, working with 821
- Microsoft Query
 - choosing a database for a database checkpoint 301, 533
- Microsoft Visual Basic scripting language 10
- MinSize command line option 693
- missing resources 505

Index

- Missing Resources pane 29
 - about 506
 - filtering 507
 - unmapped repository parameters 515
 - unmapped shared object repositories 514
- Modify Row Range dialog box 447
- modifying
 - your license 14
- moving a step 137
- Mozilla Firefox, working with 821
- multiple actions in tests 473
- multiple documents, working with 39

N

- Name and Description screen 928
- Name command line option 693
- names
 - modifying for test objects 174
- Navigate and Learn option 1131
- nesting actions 493, 868
- Netscape, working with 821
- New Merge dialog box 1165
- non-reusable action 474

O

- object identification
 - generating automation scripts 958
 - restoring defaults 958
- Object Identification dialog box 945
- object identification options, specifying 730
- Object Mapping dialog box 969
- object model
 - automation 1105
 - definition 1106
- object property values
 - restoring default 172, 174
 - specifying or modifying 170
 - viewing 166
- Object property, run-time methods 1030
- object repositories
 - adding objects 189
 - associating with actions 488
 - closing 1128

- converting from earlier version 1124
- copying, pasting, and moving objects 202
- creating 1124
- deleting objects 205
- exporting local objects 217
- exporting to XML 1146
- importing from XML 1145
- local 154
- locating objects 210
- managing 1116
- managing associations 214
- manipulating using automation 1148
- missing 505
- modifying 1130
- opening 1124
- saving 1126
- shared 155
- unmapped 514
- Object Repository Comparison Tool 1187
 - color settings 1195
 - difference types 1194
 - filtering the repositories 1200
 - repository panes 1190
 - statistics 1199
 - synchronizing repositories 1201
 - window 1189
- Object Repository Manager 1118
- Object Repository Merge Tool 1151
 - changing the view 1155
 - color settings 1164
 - conflict resolution settings 1161
 - conflicts 1175
 - filtering the target repository 1180
 - primary repository pane 1157
 - resolution options pane 1157
 - resolving conflicts 1178
 - secondary repository pane 1157
 - target repository pane 1155
 - window 1153
- object repository mode
 - choosing 153
 - setting for tests 767
- object repository types 151

- Object Repository window 156
 - compact view and full view 162
 - filtering objects 162
 - test object details 163
 - Object Selection dialog box 126
 - Object Spy 73
 - Object state trigger 907
 - objects
 - adding using navigate and learn 1131
 - deleting from object repository 205
 - identification 943
 - identifying 61
 - methods, run-time 1029
 - properties, run-time 1029
 - viewing methods 61
 - See also* test objects
 - ODBC, choosing a database for a database
 - checkpoint 533
 - online documentation xx
 - online resources xxii
 - Open QuickTest Test dialog box 104
 - Open Test from Quality Center Project dialog
 - box 1274, 1276
 - operation
 - arguments 129
 - selecting for step 128
 - selecting from Item list 122, 123
 - Operation cell 128
 - Operation column, Keyword View 116
 - Option Explicit statement 1078
 - optional steps 625
 - default 627
 - setting 626
 - Options dialog box 708
 - Active Screen tab 715
 - Folders tab 712
 - General tab 710
 - Generate Script option 710, 1111
 - Run tab 723
 - Web tab 739
 - Windows Applications tab 726
 - ordinal identifiers 952
 - specifying for test objects 183
 - OS, environment variable 391
 - OSVersion, environment variable 391
 - output types 425
 - action parameters 426
 - Data Table 427
 - environment variables 428
 - test parameters 426
 - output value categories
 - database output values 414
 - standard output values 413
 - text output values 413
 - XML output values 414
 - Output Value Properties dialog box 421
 - output values
 - database 450, 451, 453
 - definition 411
 - editing 418
 - object properties 421
 - standard 419
 - storing in action or test parameters 416
 - storing in Data Table 417
 - storing in environment variables 417
 - tables 436, 440, 444
 - text 430, 431
 - viewing 418
 - viewing results 677
 - XML 454, 462
 - output.txt log file 1325
 - outputting
 - database values 450
 - property values 419
 - text values 430
 - values 411
 - XML values 454
- P**
- Page and Frame Options dialog box 745
 - Page Checkpoint Properties dialog box 829, 831
 - page checkpoints 829
 - editing page property values 833
 - filtering hypertext links 837
 - filtering image sources 839
 - HTML verification 833

Index

panes

- auto-hiding 36
- customizing layout 31
- Debug Viewer 30
- docked 36
- floating 37
- Information 28
- Missing Resources 29
- moving 31

parameter definition, default 348, 351

Parameter Options button 348

Parameter Options dialog box 376

Parameter reserved object 773

parameter types

- action parameters 366
- Data Table parameters 378
- environment variable parameters 385
- random number parameters 396
- test parameters 366

parameter values

- action calls 875
- defining 345

parameterization example 398

parameterization icon 350, 370, 372

parameterized values, viewing in test results 675

parameterizing

- methods 367
- property values using repository parameters 1141
- tests, example 398
- using the Data Driver 403
- values 365

parameters

- action 377, 499, 868
- action guidelines 871
- environment variables, user-defined 776, 778
- handling unmapped object repository 515
- in the Expert View 979
- output from previous action call 377
- parent action 377
- passing to a WinRunner function 1257
- passing to a WinRunner test 1252

repository 1133

- adding 1136
- deleting 1139
- managing 1134
- mapping 185
- missing in 505
- modifying 1138
- setting for actions 487
- specifying for tests 771
- syntax for calling action 881
- test 377

passing data between actions 475

Password command line option 694

Password Encoder dialog box 132

passwords, encoding 132

PathFinder.Locate, statement 714

pausing run sessions 595

percentages, setting custom format 529

performance testing products, integrating with QuickTest 1313

performance, improving 1341

permissions

required for Quality Center 13

required to run QuickTest 13

Pop-up window trigger 907

post-recovery test run options 898

Post-Recovery Test Run Options screen 926

previewing function code 1068

primary repository 1152

primary repository pane 1157

Print dialog box, Test Results window 648

Print Preview dialog box 649

Print, utility statement 577

printing

function libraries 1050

tests 108

priority

setting for recovery scenarios 938

Product Information button 57

Product Information window 57

ProductDir, environment variable 391

ProductName, environment variable 391

ProductVer, environment variable 391

programmatic descriptions 213, 1005

description objects 1010

Index property 1014

- statement 1007
- variables 1007
- WebElement objects 1014
- With statement 1009
- programming 1331
 - comments 578
 - conditional statements 560
 - displaying messages during the run session 577
 - Expert View and function libraries 973
 - function libraries 973
 - generating messages 575
 - loop statements 566
 - sending messages to test results 575
 - Step Generator 540, 541
 - VBScript 996
- project (Quality Center)
 - connecting to 1261
 - disconnecting from 1268
 - opening tests in 1273
 - saving tests to 1272
- Project command line option 694
- properties 1089, 1091
 - adding for test object descriptions 177
 - CreationTime 955
 - default 61, 149
 - defining new for test object 180
 - deleting from a test object description 182
 - Index 953
 - Location 954
 - modifying for test objects 168
 - run-time objects 1029
 - setting for action calls 873
 - setting for actions 482
 - viewing for recovery scenarios 932, 938
 - viewing for steps in Keyword View 147
- Properties tab
 - Table Checkpoint Properties dialog box 249
 - Table Output Value Properties dialog box 445

- property collection. *See* programmatic descriptions
- property values
 - specifying in the test object description 1141
 - synchronization points 580

Q

- QA engineer. *See* Automation Engineer
- QCUtil object 1271
- Quality Center 1259
 - associated function libraries 1052
 - connecting QuickTest to 1261
 - Connectivity Add-in 1271
 - Data Table 530
 - disconnecting from 1268
 - environment variable files 389
 - integrating with QuickTest 1271
 - managing the testing process 11
 - opening tests in 1273
 - reporting defects
 - automatically 698
 - manually 697
 - running QuickTest tests remotely 1296
 - saving tests to a project 1272
 - using QuickTest with 11
 - version control for 1287
- Quality Center Connection - Project Connection dialog box 1263
- Quality Center Connection - Server Connection dialog box 1262
- Quality Center Connection dialog box 1266
- Quality Center OTA 1271
- query file, for a database checkpoint
 - creating 301, 533
 - working with ODBC / Microsoft Query 533
- QuickTest
 - about 3
 - access permissions, required 13
 - automation object model 1105
 - getting started 17

- integrating with Mercury application
 - management and performance
 - testing products 1313
- layout, customizing 31
- product information 57
- starting 18
- updating software 15
- window. *See* QuickTest window
- QuickTest Automation Reference 1112
- QuickTest Print Log window 577
- QuickTest window
 - Action toolbar 20, 45
 - auto-hiding panes 36
 - Automation toolbar 20, 43
 - customizing layout 31
 - Data Table 21
 - Debug toolbar 20
 - Edit toolbar 43
 - File toolbar 21
 - Information Pane 20, 28
 - Insert toolbar 44
 - look and feel 23
 - menu bar 20
 - Missing Resources 29
 - moving panes 31
 - moving tabs 31
 - multiple documents 39
 - restoring default layout 39
 - Standard toolbar 42
 - status bar 21
 - theme 23
 - title bar 21
 - Tools toolbar 44
 - View toolbar 44

R

- random number parameters 396
- Readme xx
- Record and Run Settings dialog box 790
 - environment variables 802
 - Web tab 793
 - Windows Applications tab 796
- record settings options 730

- recording
 - analog 93
 - custom Web event recording
 - configuration 1211
 - low-level 93, 1330
 - on Web sites 818
 - right mouse button clicks 1221
 - standard Web event recording
 - configuration 1209
 - status, options 1218
 - tests 88, 789
 - time, improving 1341
 - Web event recording configuration 1208
- recovery operations 898
 - Close application process 918
 - Function call 918
 - Keyboard or mouse operation 918
 - Restart Microsoft Windows 918
- Recovery Scenario Manager Dialog Box 901
- Recovery Scenario Wizard 905
 - Click Button or Press Key screen 920
 - Close Processes screen 922
 - Completing the Recovery Scenario Wizard screen 929
 - Function screen 923
 - Name and Description screen 928
 - Post-Recovery Test Run Options screen 926
 - Recovery Operation - Click Button or Press Key screen 920
 - Recovery Operation - Close Processes screen 922
 - Recovery Operation - Function Call screen 923
 - Recovery Operation screen 918
 - Recovery Operations screen 917
 - Select Object screen 911
 - Select Processes screen 915
 - Select Test Run Error screen 914
 - Select Trigger Event screen 907
 - Set Object Properties and Values screen 913
 - Specify Pop-up Window Conditions screen 909

- recovery scenarios 897
 - associating with tests 935
 - copying 934
 - deleting 934
 - disabling 939
 - files 901
 - modifying 933
 - removing from tests 938
 - saving 930
 - setting priority 938
 - viewing properties 932, 938
- Recursive command line option 694
- redirection of server 1334
- Register Browser Control Utility dialog box 852
- registering browser controls 852
- registering functions 1061
- registering methods 1072
- RegisterUserFunc statement 1061, 1072, 1074
- regular expressions 352
 - backslash (\) 357
 - defining 355
 - for constants 347
 - for property values 353
 - in checkpoints 354
 - using in function libraries 995
 - using in the Expert View and function libraries 995
- remote access to QuickTest 1296
- Remote Agent 1297
- Replace dialog box
 - Expert View 992
 - function libraries 992
- report. *See* Test Results window
- reporting defects
 - automatically 697
 - manually 697
- reports, filter 1035
- repositories. *See* object repositories
- Repository Parameter dialog box 1141
- repository parameters 1133
 - adding 1136
 - deleting 1139
 - managing 1134
 - mapping 185
 - modifying 1138
 - parameterizing values 1141
- repository types 151
- reserved objects 1052
- Resolution Options pane, Object Repository
 - Merge Tool 1157
- resolving conflicts, Object Repository Merge Tool 1178
- Resources pane 1090
- resources, missing in component 505
- resources, missing in test 505
- restoring QuickTest default layout 710
- result set 298
- ResultDir, environment variable 391
- Results Remover Utility, running from the
 - command line 691
- results. *See* run results
- reusable actions 474
- right mouse button
 - configuring QuickTest to record 1221
 - recording clicks 1221
- roles in Business Process Testing 1304
- Run dialog box 609
- run options, in the Options dialog box 723
- run properties, setting for action calls 874
- run results 631
 - checkpoints 653
 - customizing display 702
 - deleting with command line options 691
 - deleting with Test Results Deletion Tool 689
 - enabling and filtering 1035
 - exporting to HTML 651
 - filtering 641
 - finding 642
 - output values 677
 - parameterized values 675
 - previewing before printing 649
 - printing 648
 - reporting defects automatically 698
 - reporting defects manually 697
 - run-time Data Table 678
 - schema 702
 - sending messages to 575
 - Test Results window 634

Index

- viewing for a selected run 644
- viewing WinRunner steps 699
- run sessions
 - creating test objects programmatically 213
 - disabling recovery scenarios 939
 - modifying test object properties 213
 - pausing 595
 - printing results 648
 - working with test objects 213
- run settings options 736
- Run to Step 593
- running components
 - from a step 613
 - Run dialog box 609
 - to update expected results 616
 - Update Run dialog box 621
- running tests 607, 789
 - advanced issues 1330
 - from a Quality Center project 1285
 - from a step 613
 - on Web sites 818
 - Run dialog box 609
 - running WinRunner tests 1250
 - to update expected results 616
 - Update Run dialog box 621
 - using optional steps 625
 - using Silent Test Runner 1324
 - viewing results 638
- run-time
 - Data Table 518, 678
 - objects 1029
 - settings, adding and removing 1245

S

- sample application, Mercury Tours 14
- Save QuickTest Test dialog box 105
- Save Shared Object Repository dialog box 1183, 1184
- Save Test to Quality Center Project dialog box 1272
- ScenarioId, environment variable 391
- scenarios. *See* recovery scenarios
- Schema Validation dialog box, XML checkpoint 338

- schema, for run results 702
- Script Editor 1083
 - customizing the window 1086
 - display area 1093
 - Flow pane 1088
 - function libraries 1099
 - main window 1085
 - Resources pane 1090
 - tests 1095
- scripts, test. *See* tests
- secondary object repository 1152
- secondary repository pane 1157
- Section 508, Web Content Accessibility Guidelines 5, 841
- Select Action dialog box 859, 862
- Select Object for Step dialog box 123
- Select Object screen 911
- Select Processes screen 915
- Select Test Run Error screen 914
- Select Trigger Event screen 907
- selecting a test object
 - from Item list 123
 - from shared object repository 123
 - from your application 126
- server
 - Quality Center, disconnecting from 1268
 - redirections 1334
 - server-side connections 1333
- Server command line option 695
- session IDs 1334
- Set Object Properties and Values screen 913
- Set statement, in the Expert View and function libraries 999
- Setting object 1242
- Settings tab, Database Checkpoint Properties dialog box 308
- SetTOProperty method 213
- SGML 1335
- shared object repositories 151, 155
 - associating with actions 488
 - comparing 1187
 - managing associations 214
 - merging 1151
 - unmapped 514
 - Update from Local Repository 1168

- shared object repository window 1122
- Sheet menu commands, Data Table 526
- shortcut keys
 - in Keyword View 140
 - in QuickTest 45
- shortcuts
 - for menu items 45
 - in Expert View 1238
 - in function libraries 1238
 - in Object Repository Comparison Tool 1193
 - in Object Repository Merge Tool 1160
 - in QuickTest 45
- Silent command line option 695
- Silent Test Runner 1323
 - opening 1324
 - running tests from 1324
- Silent Test Runner dialog box 1324
- Smart Identification
 - analyzing information 685
 - configuring 959
 - disabling during test runs 765
 - enabling from the Object Identification dialog box 957, 958
- Smart Identification Properties dialog box 964
- snapshots
 - Active Screen capture settings 717
 - Test Results window 632
- software updates 15
- Specifications for Data Table 523
- Specify Pop-up Window Conditions screen 909
- Specify SQL statement screen, for creating database checkpoints 302
- Split Action dialog box 496
- splitting actions 495
- Spy. *See* Object Spy
- standard checkpoints
 - analyzing results 654
 - specifying timeout 280, 836
- standard output values 413
 - creating 419
 - specifying 421
- Standard toolbar, QuickTest window 42
- standard Web event recording configuration 1209
- Start from Step 593
- Start Transaction dialog box 1321
- starting QuickTest 18
- statement completion 980, 1234
- statements, using in Keyword View 136
- Statistics dialog box 1174
 - Comparison Tool 1199
- status bar
 - Object Repository Comparison Tool 1192
 - Object Repository Merge Tool 1158
 - QuickTest window 21
- Step commands 590
- Step Generator 540, 541
- Step Generator dialog box 544
- steps
 - adding 120
 - adding after block 135
 - adding to Keyword View 120
 - deleting 139
 - deleting from Keyword View 139
 - inserting 541
 - managing for component 137
 - manual 136
 - modifying in Keyword View 136
 - moving 137
 - optional 625
 - viewing properties in Keyword View 147
- Subject Matter Expert, role in Business Process Testing 1304
- Summary column, Keyword View 117
- synchronization points
 - creating 580
 - inserting 581
- synchronization timeout
 - setting 764
- synchronizing repositories
 - Object Repository Comparison Tool 1201
- synchronizing tests 579
 - modifying timeout values 584
 - synchronization point 580

Index

- waiting for objects to appear 583
- waiting for specified property values 580
- syntax
 - actions 880
 - for action parameters 881
 - for action return values 882
- syntax errors, VBScript 1003
- SystemTempDir, environment variable 391
- SystemUtil.Run method 1017

T

- Table Checkpoint Properties dialog box 238
 - Expected Data tab 245
 - Properties tab 249
 - Table Content tab 239
- table checkpoints
 - about 233
 - analyzing results 656
 - creating 234
 - general options 240
 - modifying 252
 - specifying cell identification settings 247
 - specifying cells 243
 - specifying expected data 245
 - specifying value type 246
 - Table Content tab 241
 - Table Properties tab 241
- Table Content tab
 - Table Checkpoint Properties dialog box 239
 - Table Output Value Properties dialog box 440
- Table Output Value Properties dialog box 440
 - Properties tab 445
 - Table Content tab 440
- table output values 440
 - modifying output options 447
 - modifying row range 447
 - Table Content tab 442
 - Table Properties tab 442

- table properties
 - specifying which to check 250
 - specifying which to output 446
- target repository 1152
 - saving 1183
- target repository pane 1155
- template tests 1277, 1280
- templates, actions 503
- test batches, running 627
- Test command line option 695
- test database, maintaining 1107
- test flow (actions) 477
- test object properties 61
- test objects
 - adding
 - description properties 177
 - to object repository 189
 - copying to local repository 164
 - copying, pasting, and moving in object repository 202
 - creating in run sessions 213
 - creating using programmatic descriptions 213
 - defining new 200
 - defining new properties 180
 - deleting description properties 182
 - finding 206
 - highlighting in an application 209
 - identifying 61
 - in run sessions 213
 - locating in object repository 206, 210
 - managing 149
 - modifying
 - in run sessions 213
 - names 174
 - properties 163, 168
 - properties during run sessions 213
 - property values, replacing 206
 - property values, retrieving and setting 1027
 - renaming 174
 - selecting
 - from application 126
 - from Item list 123
 - from shared object repository 123

- specifying ordinal identifiers 183
- viewing properties 166
- test parameters 366, 374
 - setting options 376
 - storing output values 416, 426
 - using in steps 773
- test resources, missing 505
- Test Results Deletion Tool 689
- Test Results toolbar, Test Results window 637
- Test Results tree 635
- Test Results window 634
 - look and feel 638
 - Test Results toolbar 637
 - test results tree 635
 - theme 638
- test results. *See* run results
- Test run error trigger 907
- Test Run Log 1325
- test run time, improving 1341
- test set 1286
- Test Settings dialog box 757
 - Environment tab 774
 - Generate Script option 1111
 - Parameters tab 771
 - Properties tab 759
 - Recovery tab 784
 - Resources tab 767
 - Run tab 763
 - Web tab 782
- test versions in QuickTest 1287
- TestDir, environment variable 391
- TestDirector. *See* Quality Center
- testing options
 - during a test run 1241
 - restoring 1244
 - retrieving 1244
 - run-time 1245
 - setting 1242
 - setting for all tests 707
 - setting for an individual test 755
- testing process 6
 - analyzing test results 9
 - creating tests 6, 7
 - running tests 9
- TestIteration, environment variable 391
- TestName, environment variable 391
- tests
 - about test steps 92
 - adding to version control 1287
 - and components, a comparison 1312
 - associating recovery scenarios with 935
 - checking in to version control 1290
 - checking out of version control 1288
 - checkpoints. *See* checkpoints
 - closing in the Script Editor 1098
 - creating 79, 103
 - creating in Quality Center using a template test 1281
 - debugging 587
 - diagram 472, 856, 857
 - disabling recovery scenarios 939
 - editing in the Script Editor 1097
 - enhancing 101
 - local 474
 - managing 103
 - managing in Quality Center 11
 - opening in a Quality Center project 1273
 - opening in QuickTest 103
 - opening in the Script Editor 1095
 - parameterizing, example 398
 - pausing runs 595
 - planning 81
 - printing 108
 - properties 1089, 1091
 - recording 88, 789
 - removing recovery scenarios from 938
 - running 607, 789
 - running from a step 613
 - running using optional steps 625
 - running using Silent Test Runner 1324
 - saving 105
 - saving in the Script Editor 1098
 - saving to a Quality Center project 1272
 - unzipping 108
 - updating 616
 - working with 1095
 - zipping 107
 - See also* run results

- Text Checkpoint Properties dialog box 259
- text checkpoints 255, 257
 - analyzing results 660
 - configuring the text selection 261
 - modifying 269
 - setting options 261
 - specifying the checked text 264
 - specifying the text after 266
 - specifying the text before 265
 - specifying timeout 268
 - standard checkpoints 269
 - types 256
- TEXT function in Data Table worksheet 534
- Text Output Value Properties dialog box 431
- text output values 413
 - creating 430
 - specifying 431
- text values, outputting 430
- timeout
 - setting 764
 - specifying for standard checkpoint 280, 836
 - specifying for text checkpoints 268
- times, setting custom format 529
- title bar, QuickTest window 21
- toolbars
 - Object Repository Comparison Tool 1193
 - Object Repository Merge Tool 1159
 - QuickTest window
 - Action 45
 - Automation 43
 - Debug 20, 43
 - Edit 43
 - File 21
 - Insert 44
 - Standard 42
 - Testing 20
 - Tools 44
 - View 44
- Tools toolbar, QuickTest window 44
- transactions 1319
 - defining 1319
 - ending 1322
 - inserting 1321
 - measuring 1319

- Tree View. *See* Keyword View
- trigger
 - Application crash 907
 - events 898
 - Object state 907
 - Pop-up window 907
 - test run error 907
- TSL functions, calling from QuickTest 1254
- type library 1108
- typographical conventions xxiv

U

- Unicode 4
- unregistering methods, using the
 - UnregisterUserFunc statement 1076
- UnregisterUserFunc statement 1072
- UntilDate command line option 696
- unzipping tests 108
- Update Run dialog box 621
- updates, documentation xxiii
- UpdatingActiveScreen, environment variable 392
- UpdatingCheckpoints, environment variable 392
- URL_ENV variable 804
- User command line option 696
- user-defined
 - functions. *See* user-defined functions
 - methods 1072
 - properties, accessing 1030
 - test objects, mapping 969
- user-defined functions 1037
 - adding a tooltip to 1066
 - documenting 1066
 - finalizing 1069
 - Function Definition Generator 1056
 - generating additional 1069
 - guidelines for 1077
 - previewing code in Function Definition Generator 1068
 - registering 1061
- UserName, environment variable 392

V

- Value cell 129
- Value column, Keyword View 116
- Value Configuration Options dialog box 350, 370
- VALUE function in Data Table worksheet 534
- values
 - configuring 345
 - input 129
 - outputting 411
 - parameterizing 365
 - restoring default for object properties 172, 174
 - specifying for object properties 170
 - viewing for object properties 166
- variables
 - environment 774
 - unique in global scope 1078
 - See also* environment variables, user-defined
- VBScript 1108
 - associated function libraries
 - with Quality Center 1052
 - auto-expand syntax 1235
 - documentation 1019
 - formatting text 1001
 - syntax 996
 - syntax errors 1003
- version control 1287
 - adding tests to 1287
 - checking tests in to 1290
 - checking tests out of 1288
- version manager 1287
- View toolbar 44
- Viewlink objects 818
- Virtual Object Manager 894
- Virtual Object wizard 890
- virtual objects 885
 - defining 889
 - removing 894
- Visual Basic 1108
- Visual C++ 1108
- Visual Studio.NET 1108
- VuserId, environment variable 392

W

- W3C Web Content Accessibility Guidelines 5, 841
- Wait statement 583
- WaitProperty statement 580
- Web browsers, supported 818
- Web content accessibility checkpoints 841
 - automatically adding 841
 - in test results 670, 844
 - manually adding 842
 - setting preferences 841
- Web content, dynamic 1332
- Web event recording configuration 1207
 - custom 1211
 - resetting 1228
 - standard 1209
- Web Event Recording Configuration dialog box 1210, 1222
- Web Page Appearance dialog box 721
- Web settings
 - Advanced Web Options dialog box 748
 - Browser Details dialog box 742
 - ignore browsers 741
 - Options dialog box 739
 - Page and Frame Options dialog box 745
- Web sites, recording and running tests 818
- Web tab, Record and Run Settings dialog box 793
- WebElement objects, programmatic descriptions 1014
- While statement, in the Expert View and function libraries 1024
- Windows API 1031
- Windows applications
 - adding 799
 - editing 799
 - settings 726
- Windows Applications settings, Advanced Web Options dialog box 729
- Windows Applications tab, Record and Run Settings dialog box 796
- Windows command line options 691
- Windows dialog box 39

Index

WinRunner

- calling tests from QuickTest 1250
- calling TSL functions from QuickTest 1254
- function arguments, passing
 - parameters from QuickTest 1257
- tests, passing parameters from QuickTest 1252
- viewing WinRunner steps in test results 699
- working with 1249

With statements

- entering manually 1026
- generating automatically, while recording 570
- generating for existing actions 572
- in the Expert View 569
- removing 574
- With Generation Results window 573

WORKDIR_ENV variable 804

workflow in Business Process Testing 1307

worksheet functions in the Data Table 534

wscript.exe 1109

analyzing 679

attribute details 682

XML Checkpoint from File dialog box 321

XML Checkpoint Properties dialog box 328

XML Checkpoint Results window 662

XML Output Properties dialog box 462

XML Output Value Results window 680

XML output values 414

XML structure

importing 334, 466

updating 334, 466

updating using Update Run mode 334, 466

XML values, outputting 454

Z

zip files

exporting tests to 107

importing tests from 108

zipping tests 107

X

XML

checkpoint results

attribute details 664

checkpoint summary 663

checkpoints 313

Add Schema dialog box 341

analyzing results 342, 661

Edit Schema dialog box 341

for files 321

for test objects 324

for web page/frame 317

modifying 342

namespace 315, 343, 454

Schema Validation dialog box 338

XPath 343

Edit XML dialog box 333

exporting from object repository 1146

importing as object repository 1145

objects and methods 343

output value results 679