

HP QuickTest Professional

Software Version: 10.00

User Guide Volume 1

Manufacturing Part Number: T6513-90039

Document Release Date: January 2009

Software Release Date: January 2009



Legal Notices

Warranty

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

The information contained herein is subject to change without notice.

Restricted Rights Legend

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

Third-Party Web Sites

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

Copyright Notices

© 1992 - 2009 Mercury Interactive (Israel) Ltd.

Trademark Notices

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

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

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

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

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

Unix® is a registered trademark of The Open Group.

SlickEdit® is a registered trademark of SlickEdit Inc.

Documentation Updates

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

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

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

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

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

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

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

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

Support

You can visit the HP Software Support web site at:

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

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

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

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

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

To find more information about access levels, go to:

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

To register for an HP Passport ID, go to:

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

Table of Contents

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

Welcome to This Guide xxi
How This Guide Is Organized xxii
Who Should Read This Guidexxiv
QuickTest Professional Online Documentationxxiv
Additional Online Resources.....xxvii

PART I: INTRODUCING QUICKTEST PROFESSIONAL (VOL. 1)

Chapter 1: Introduction3
Testing with QuickTest.....5
Understanding the Testing Process7
Programming in the Expert View.....13
Understanding Functions and Function Libraries14
Managing the Testing Process Using Quality Center14
Understanding Business Process Testing.....15
Setting Required Access Permissions16
Using the Sample Site.....17
Modifying License Information17
Updating QuickTest Software.....18

Chapter 2: QuickTest at a Glance 19
Starting QuickTest 20
The QuickTest Window 23
Keyword View 28
Expert View 29
Function Library 30
Start Page 31
Active Screen 33
Available Keywords Pane 34
Data Table 35
Debug Viewer Pane 36
Information Pane 37
Missing Resources Pane 38
Process Guidance Panes 39
Resources Pane 40
Test Flow Pane 41
To Do Pane 42
Using QuickTest Commands 43
Browsing the QuickTest Professional Program Folder 69
Viewing Product Information 73

PART II: WORKING WITH TEST OBJECTS (VOL. 1)

Chapter 3: Understanding the Test Object Model 79
About Understanding the Test Object Model 79
Applying the Test Object Model Concept 83
Understanding Object Repository Types 89
Viewing Object Properties and Operations Using the Object Spy 97
The Object Spy Dialog Box 100
Chapter 4: Configuring Object Identification 105
About Configuring Object Identification 106
Understanding the Object Identification Dialog Box 107
Configuring Smart Identification 121
Mapping User-Defined Test Object Classes 131
Chapter 5: Managing Test Objects in Object Repositories 135
Adding Test Objects to a Local or Shared Object Repository 136
Copying, Pasting, and Moving Objects in the Object Repository 150
Deleting Objects from the Object Repository 153
Locating Objects 154
Maintaining Identification Properties 162

Chapter 6: Using Object Repositories in Your Test	181
Understanding the Object Repository Window.....	182
The Object Properties Dialog Box	197
Managing Shared Object Repository Associations	199
Mapping Repository Parameter Values	202
Working with Test Objects During a Run Session	206
Chapter 7: Managing Object Repositories	207
About Managing Object Repositories.....	208
The Object Repository Manager	210
Working with Object Repositories	217
Managing Objects in Shared Object Repositories	222
Working with Repository Parameters	228
Modifying Object Details	234
Locating Test Objects	239
Performing Merge Operations.....	240
Performing Import and Export Operations.....	241
Managing Object Repositories Using Automation	244
Chapter 8: Merging Shared Object Repositories.....	247
About Merging Shared Object Repositories	248
Understanding the Object Repository Merge Tool	250
Using Object Repository Merge Tool Commands.....	257
Defining Default Settings	262
Merging Two Object Repositories	267
Updating a Shared Object Repository from Local Object Repositories.....	269
Viewing Merge Statistics.....	276
Understanding Object Conflicts	277
Resolving Object Conflicts	280
Filtering the Target Repository Pane	282
Finding Specific Objects	284
Saving the Target Object Repository	285

Chapter 9: Comparing Shared Object Repositories287
About Comparing Shared Object Repositories288
Understanding the Object Repository Comparison Tool289
Using Object Repository Comparison Tool Commands293
Understanding Object Differences297
Changing Color Settings298
Comparing Object Repositories299
Viewing Comparison Statistics.....301
Filtering the Repository Panes.....302
Synchronizing Object Repository Views.....303
Finding Specific Objects304

PART III: DESIGNING TESTS (VOL. 1)

Chapter 10: Creating Tests — Overview.....309
About Creating Tests309
Deciding Which Methodology to Use -
 Keyword-Driven or Recording311
Understanding Your Test313
Enhancing Your Test315
Using Relative Paths in QuickTest316

Chapter 11: Managing Your Test.....321
Creating a New Test321
Opening an Existing Test322
Saving a Test324
Creating Portable Copies of Your Tests.....326
Zipping a Test331
Unzipping a Test331
Printing a Test332

**Chapter 12: Creating Tests Using the Keyword-Driven
 Methodology335**
Understanding the Keyword-Driven Methodology336
Using the Keyword-Driven Methodology341
Sample Implementation of the Keyword-Driven Methodology351

Chapter 13: Creating Tests Using the Recording Mechanism361
About Recording Tests.....362
Recording a Test364
Choosing the Recording Mode368
Working with the Active Screen376

Chapter 14: Working with the Keyword View.....	383
About Working with the Keyword View	384
The Keyword View.....	385
Understanding the QuickTest Object Hierarchy.....	391
Adding a Standard Step to Your Test	392
Adding Other Types of Steps to Your Test	407
Modifying the Parts of a Step	410
Working with Comments	410
Managing Action Steps.....	412
Using Keyboard Commands in the Keyword View	415
Defining Keyword View Display Options	416
Viewing Properties of Step Elements in the Keyword View.....	422
Working with Breakpoints in the Keyword View	423
Chapter 15: Working with Actions	425
About Working with Actions	426
Using Global and Action Data Sheets	429
Using the Test Flow Pane	431
Using the Action Toolbar in the Keyword View	435
Creating New Actions.....	436
Guidelines for Working with Actions	439
Setting Action Properties.....	441
Nesting Actions	453
Splitting Actions	455
Renaming Actions	457
Removing Actions from a Test	460
Creating an Action Template	462
Chapter 16: Working with Advanced Action Features	463
About Working with Advanced Action Features	464
Inserting Calls to Existing Actions	464
Setting Action Parameters	472
Using Action Parameters	476
Setting Action Call Properties	481
Sharing Action Information	486
Understanding Action Syntax in the Expert View.....	488
Exiting an Action.....	491

PART IV: ENHANCING TESTS (VOL. 1)

Chapter 17: Understanding Checkpoints495
About Understanding Checkpoints495
Adding New Checkpoints to a Test.....496
Adding Existing Checkpoints to a Test.....498
Understanding Types of Checkpoints.....501

**Chapter 18: Checking Object Property Values Using Standard
Checkpoints505**
About Checking Object Property Values505
Creating Standard Checkpoints506
Understanding the Checkpoint Properties Dialog Box508
Understanding the Image Checkpoint Properties Dialog Box.....512
Modifying Checkpoints.....514

Chapter 19: Checking Bitmaps515
About Checking Bitmaps515
Fine-Tuning the Bitmap Comparison516
Creating and Modifying Bitmap Checkpoints.....518
The Bitmap Checkpoint Properties Dialog Box522

Chapter 20: Checking Tables529
About Checking Tables529
Creating a Table Checkpoint530
Understanding the Table Checkpoint Properties Dialog Box.....535
Checking Table Content536
Checking Table Properties.....546
Modifying a Table Checkpoint548

Chapter 21: Checking Text551
About Checking Text551
Creating a Text Checkpoint552
Creating a Text Area Checkpoint.....554
The Text / Text Area Checkpoint Properties Dialog Box557
Modifying a Text or Text Area Checkpoint570
Creating a Standard Checkpoint for Checking Text.....570

Chapter 22: Checking Databases575
About Checking Databases.....575
Creating a Check on a Database576
Understanding the Database Checkpoint Properties Dialog Box581
Modifying a Database Checkpoint.....590

Chapter 23: Checking XML	591
About Checking XML.....	592
Creating XML Checkpoints.....	594
Updating the XML Hierarchy for XML Test Object Operation	
Checkpoints (for WebService Test Objects Only).....	614
Modifying XML Checkpoints.....	622
Reviewing XML Checkpoint Results	622
Using XML Objects and Methods to Enhance Your Test	623
Chapter 24: Parameterizing Values	625
About Parameterizing Values	626
Parameterizing Values in Steps and Checkpoints.....	628
Using Test and Action Input Parameters	635
Using Data Table Parameters.....	639
Using Environment Variable Parameters	645
Using Random Number Parameters.....	655
Example of a Parameterized Test.....	657
Using the Data Driver to Parameterize Your Test	662
Chapter 25: Outputting Values	669
About Outputting Values	669
Creating Output Values.....	670
Outputting Property Values	676
Specifying the Output Type and Settings	683
Outputting Text Values.....	688
Outputting Table Values	698
Outputting Database Values.....	713
Outputting XML Values	718
Updating the XML Hierarchy for XML Test Object Operation	
Output Value Steps (For WebService Test Objects Only)	732
Adding Existing Output Values to a Test	736
Chapter 26: Working with Text Recognition for	
Windows-Based Objects	741
About Working with Text Recognition for Windows-Based	
Objects	742
The Options Dialog Box: General > Text Recognition Pane.....	742
Guidelines for Text Recognition	746
Text Recognition and Development Environments	748
Use-Case Scenario: Checking Text in an Image	750

Chapter 27: Configuring Values.....	755
About Configuring Values.....	755
Configuring Constant and Parameter Values	756
Understanding and Using Regular Expressions	762
Defining Regular Expressions.....	765
Chapter 28: Adding Steps Containing Programming Logic	775
About Adding Steps Containing Programming Logic	776
Inserting Steps Using the Step Generator	777
Using Conditional Statements	797
Using Loop Statements.....	803
Generating With Statements for Your Test.....	806
Generating Messages	812
Adding Comments	815
Synchronizing Your Test	816

**PART V: DEFINING FUNCTIONS AND OTHER PROGRAMMING TASKS
(VOL. 2)**

Chapter 29: Working in the Expert View and Function Library	
Windows	825
About Working in the Expert View and Function Library	
Windows	826
Understanding and Using the Expert View	827
Navigating in the Expert View and Function Libraries	843
Understanding Basic VBScript Syntax.....	853
Using Programmatic Descriptions.....	863
Running and Closing Applications Programmatically	875
Using Comments, Control-Flow, and Other VBScript Statements...	876
Retrieving and Setting Identification Property Values	886
Accessing Native Properties and Operations.....	887
Running DOS Commands.....	889
Enhancing Your Tests and Function Libraries Using the	
Windows API.....	889
Choosing Which Steps to Report During the Run Session	893
Chapter 30: Customizing the Expert View and Function Library	
Windows	895
About Customizing the Expert View and Function Library	
Windows	896
Customizing Editor Behavior	897
Customizing Element Appearance	900
Personalizing Editing Commands.....	902

Chapter 31: Working with User-Defined Functions and Function Libraries.....	905
About Working with User-Defined Functions and Function Libraries.....	906
Managing Function Libraries	908
Working with Associated Function Libraries	919
Using the Function Definition Generator.....	923
Registering User-Defined Functions as Test Object Methods	939
Additional Tips for Working with User-Defined Functions	945
Executing Externally-Defined Functions from Your Test	948

PART VI: RUNNING AND ANALYZING TESTS (VOL. 2)

Chapter 32: Running Tests.....	953
About Running Tests	954
Running Your Entire Test.....	955
Running Part of Your Test.....	956
The Run Dialog Box: Results Location Tab	960
The Run Dialog Box: Input Parameters Tab.....	962
Using Optional Steps.....	963
Running a Test Batch	966
Chapter 33: Viewing Run Session Results.....	969
About Viewing Run Session Results	970
The Test Results Window	971
Viewing the Results of a Run Session.....	980
Deleting Run Results	1004
Submitting Defects Detected During a Run Session	1013
Viewing WinRunner Test Steps in the Test Results	1017
Customizing the Test Results Display	1019
Chapter 34: Analyzing Run Session Results.....	1023
Analyzing Smart Identification Information in the Test Results....	1024
Viewing Checkpoint Results	1028
Viewing Parameterized Values and Output Value Results.....	1053
Viewing System Monitor Results.....	1063

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

Chapter 35: Debugging Tests and Function Libraries.....	1069
About Debugging Tests and Function Libraries	1070
Slowing a Debug Session	1072
Using the Single Step Commands.....	1072
Using the Run to Step and Debug from Step Commands	1076
Pausing a Run Session	1078
Using Breakpoints	1078
The Debug Viewer Pane	1082
Handling Run Errors.....	1094
Practicing Debugging an Action or a Function.....	1096
Chapter 36: Maintaining Tests.....	1101
Why Tests Fail	1102
Running Tests with the Maintenance Run Wizard.....	1104
Updating a Test Using the Update Run Mode Option	1125

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

Chapter 37: QuickTest Window Layout	1135
Modifying the QuickTest Window Layout	1135
Customizing Toolbars and Menus	1146
Working with Multiple Documents.....	1159
Chapter 38: Managing Resources	1161
The Resources Pane	1161
Chapter 39: Adding Keywords to Your Test.....	1165
Understanding the Available Keywords Pane	1165
Chapter 40: Managing QuickTest Tasks and Comments	1169
Working with Tasks and TODO Comments.....	1169
The To Do Pane	1170
The Task Editor Dialog Box	1177
Chapter 41: Handling Missing Resources	1179
About Handling Missing Resources.....	1180
Handling Missing Actions	1183
Handling Missing Environment Variables Files.....	1188
Handling Missing Function Libraries.....	1189
Handling Missing Shared Object Repositories	1191
Handling Missing Recovery Scenarios	1192
Handling Unmapped Shared Object Repository Parameter Values	1194

Chapter 42: Working with Data Tables	1197
About Working with Data Tables.....	1197
Working with Global and Action Sheets	1199
Saving the Data Table.....	1201
Editing the Data Table.....	1202
Using Data Table Files with Quality Center.....	1212
Importing Data from a Database.....	1213
Using Formulas in the Data Table.....	1216
Using Data Table Scripting Methods.....	1220
Chapter 43: Working with Process Guidance	1221
Process Guidance Panes.....	1222
Opening Process Guidance.....	1224
Managing the List of Available Processes.....	1225
The Process Guidance Management Dialog Box	1226

PART IX: CONFIGURING QUICKTEST SETTINGS (VOL. 2)

Chapter 44: Setting Global Testing Options	1231
About Setting Global Testing Options	1231
Using the Options Dialog Box	1232
Setting General Testing Options	1234
Setting Folder Testing Options.....	1237
Setting Active Screen Options	1240
Setting Run Testing Options	1253
Chapter 45: Setting Options for Individual Tests.....	1261
Using the Test Settings Dialog Box	1262
Defining Properties for Your Test.....	1265
Defining Run Settings for Your Test	1270
Defining Resource Settings for Your Test.....	1274
Defining Parameters for Your Test	1280
Defining Environment Settings for Your Test	1283
Defining Recovery Scenario Settings for Your Test.....	1291
Enabling System Monitoring for Your Test	1296
Chapter 46: Using the Setting Object to Set Testing Options	
During the Run Session	1301
About Setting Testing Options During the Run Session.....	1301
Setting Testing Options.....	1302
Retrieving Testing Options.....	1304
Controlling the Test Run.....	1305
Adding and Removing Run-Time Settings.....	1305

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

Chapter 47: Learning Virtual Objects1309
About Learning Virtual Objects1310
Understanding Virtual Objects1311
Understanding the Virtual Object Manager1312
Defining a Virtual Object1314
Removing or Disabling Virtual Object Definitions.....1327

Chapter 48: Defining and Using Recovery Scenarios1329
About Defining and Using Recovery Scenarios1330
Deciding When to Use Recovery Scenarios1332
Defining Recovery Scenarios1333
Understanding the Recovery Scenario Wizard1338
Managing Recovery Scenarios1367
Associating Recovery Scenarios with Your Tests.....1372
Programmatically Controlling the Recovery Mechanism1379

Chapter 49: Working with the QuickTest Script Editor.....1381
About the QuickTest Script Editor1382
Understanding the QuickTest Script Editor Window1383
Customizing the QuickTest Script Editor Window.....1384
Understanding the Flow Pane1386
Understanding the Resources Pane1388
Understanding the Display Area1391
Working with Tests1393
Working with Function Libraries.....1397

Chapter 50: Automating QuickTest Operations1403
About Automating QuickTest Operations1404
Deciding When to Use QuickTest Automation Scripts.....1405
Choosing a Language and Development Environment for
 Designing and Running Automation Scripts1407
Learning the Basic Elements of a QuickTest Automation Script1409
Generating Automation Scripts1410
Using the QuickTest Automation Reference.....1411

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

Chapter 51: Integrating with Quality Center	1415
About Working with Quality Center	1416
Connecting to and Disconnecting from Quality Center	1418
Integrating QuickTest with Quality Center	1424
Saving Tests to a Quality Center Project	1425
Opening Tests from a Quality Center Project	1426
Working with Template Tests	1430
Running a Test Stored in a Quality Center Project from QuickTest	1437
Setting Preferences for Quality Center Test Runs	1439
Chapter 52: Using the Resources and Dependencies Model	1447
Resources and Dependencies Model Terminology	1448
About the Resources and Dependencies Model	1449
Advantages of Working with Asset Dependencies	1451
Working With the Resources and Dependencies Model in Quality Center	1452
Chapter 53: Viewing and Comparing Versions of QuickTest Assets	1461
Working with the Asset Comparison Tool and Asset Viewer	1462
The QuickTest Asset Comparison Tool	1465
The QuickTest Asset Viewer	1474
Chapter 54: Managing Assets Using Version Control	1479
Managing Versions of Assets in Quality Center	1480
Viewing Version History for an Asset	1488
Viewing Baseline History	1490
Version History Versus Baseline History	1494
Chapter 55: Working with Version Control in Quality Center 9.x	1495
Opening Tests from a Quality Center 9.x Project with Version Control Support	1496
Managing Test Versions in QuickTest	1496

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

Chapter 56: Working with Business Process Testing	1507
About Working with Business Process Testing	1507
Understanding Business Process Testing Roles	1508
Understanding Business Process Testing Methodology	1512

Chapter 57: Working with WinRunner1517
About Working with WinRunner1517
Calling WinRunner Tests1518
Calling WinRunner Functions1522

**Chapter 58: Working with HP Performance Testing and
Business Availability Center Products.....1527**
About Working with HP Performance Testing and
Business Availability Center Products1528
Using QuickTest Performance Testing and
Business Availability Center Features1529
Designing QuickTest Tests for Use with
Performance Testing Products or Business Process Monitor1530
Inserting and Running Tests in a Performance Test or in
Business Process Monitor.....1531
Measuring Transactions1534
Using Silent Test Runner1538

PART XIII: APPENDIXES (VOL. 2)

Appendix A: Supported Checkpoints and Output Values
Per Add-In1545
Supported Checkpoints1546
Supported Output Values1548

Appendix B: Frequently Asked Questions.....1551
Creating Tests1552
Programming in the Expert View.....1553
Working with Dynamic Content1555
Advanced Web Issues1557
Standard Windows Environment.....1560
Test Maintenance1561
Testing Localized Applications.....1563
Improving QuickTest Performance1564

Appendix C: Creating Custom Process Guidance Packages1569
About Process Guidance Packages.....1569
Understanding the Package Configuration File.....1570
Creating Data Files1573
Installing Custom Process Guidance Packages in QuickTest.....1574

Appendix D: Bitmap Checkpoint Customization	1575
About Bitmap Checkpoint Customization	1576
Developing a Custom Bitmap Comparer	1579
Tutorial: Creating a Custom Comparer	1589
Using the Bitmap Checkpoint Customization Samples	1600
Index	I-1

Welcome to This Guide

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

This chapter includes:

- How This Guide Is Organized on page xxii
- Who Should Read This Guide on page xxiv
- QuickTest Professional Online Documentation on page xxiv
- Additional Online Resources on page xxvii

How This Guide Is Organized

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

This guide contains the following parts:

Volume 1

Part I Introducing QuickTest Professional

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

Part II Working with Test Objects

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

Part III Designing Tests

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

Part IV Enhancing Tests

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

Volume 2

Part V Defining Functions and Other Programming Tasks

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

Part VI Running and Analyzing Tests

Describes how to run tests and analyze the results.

Part VII Maintaining and Debugging Tests

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

Part VIII Working with the QuickTest IDE

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

Part IX Configuring QuickTest Settings

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

Part X Working with Advanced Testing Features

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

Part XI Working with Quality Center

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

Part XII Working with Other HP Products

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

Part XIII Appendixes

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

Who Should Read This Guide

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

QuickTest Professional Online Documentation

QuickTest Professional includes the following online documentation:

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

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

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

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

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

QuickTest Professional Help includes:

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

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

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

Additional Online Resources

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

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

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

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

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

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

To find more information about access levels, go to:

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

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

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

Part I

Introducing QuickTest Professional

1

Introduction

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

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

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

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



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

This chapter includes:

- Testing with QuickTest on page 5
- Understanding the Testing Process on page 7
- Programming in the Expert View on page 13
- Understanding Functions and Function Libraries on page 14
- Managing the Testing Process Using Quality Center on page 14
- Understanding Business Process Testing on page 15
- Setting Required Access Permissions on page 16
- Using the Sample Site on page 17
- Modifying License Information on page 17
- Updating QuickTest Software on page 18

Testing with QuickTest

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

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

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

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

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

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

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

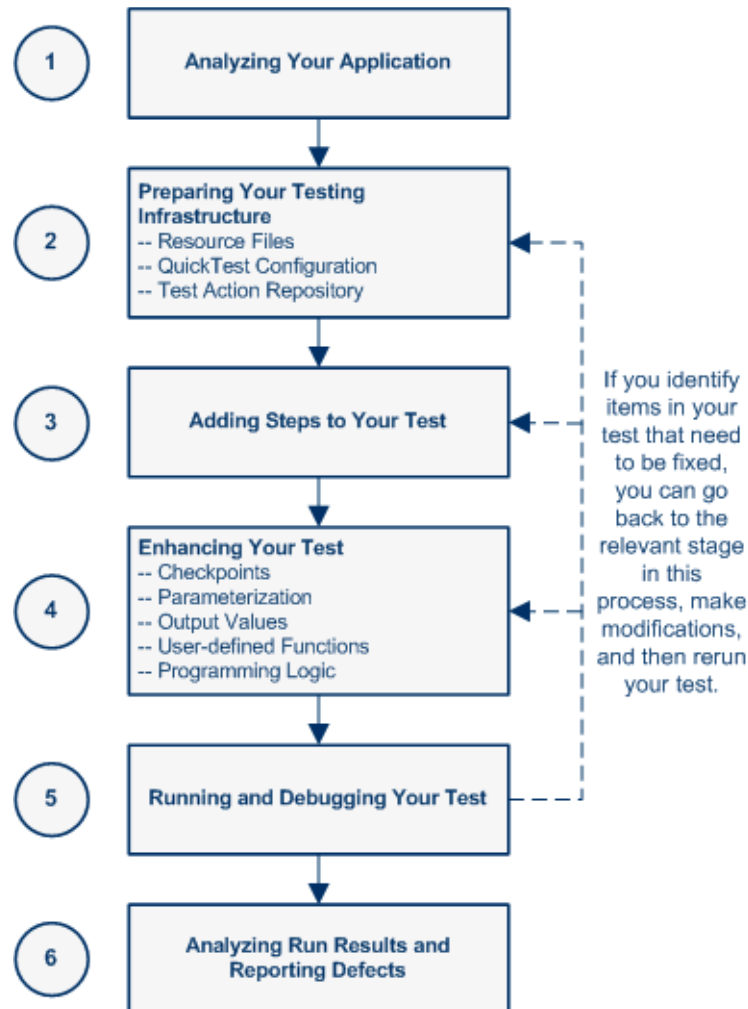
When you perform a run session, QuickTest performs each step in your test. After the run session ends, you can view a report detailing which steps were performed, and which ones succeeded or failed.

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

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

Understanding the Testing Process

Testing with QuickTest involves the following main stages:



Stage 1: Analyzing Your Application

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

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

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

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

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

Stage 2: Preparing the Testing Infrastructure

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

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

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

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

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

Stage 3: Adding Steps to Your Actions

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

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

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

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

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

- Record on your application.

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

Stage 4: Enhancing Your Test

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

- Insert checkpoints and output values into your test.

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

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

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

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

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

- Add user-defined functions by creating function libraries and calling their functions from your test. For more information, see Chapter 31, “Working with User-Defined Functions and Function Libraries.”
- Use the many functional testing features included in QuickTest to enhance your test and/or add programming statements to achieve more complex testing goals. For more information, see Chapter 28, “Adding Steps Containing Programming Logic.”

Stage 5: Running and Debugging Your Test

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

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

Stage 6: Analyzing Test Results and Reporting Defects

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

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

Programming in the Expert View

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

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

Understanding Functions and Function Libraries

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

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

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

Managing the Testing Process Using Quality Center

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

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

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

Understanding Business Process Testing

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

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

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

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

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

Setting Required Access Permissions

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

Permissions Required to Run QuickTest Professional

You must have the following file system permissions:

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

You must have the following registry key permissions:

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

Permissions Required When Working with Quality Center

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

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

Using the Sample Site

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

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

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

Modifying License Information

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

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

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

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

Updating QuickTest Software

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

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

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

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

2

QuickTest at a Glance

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

This chapter includes:

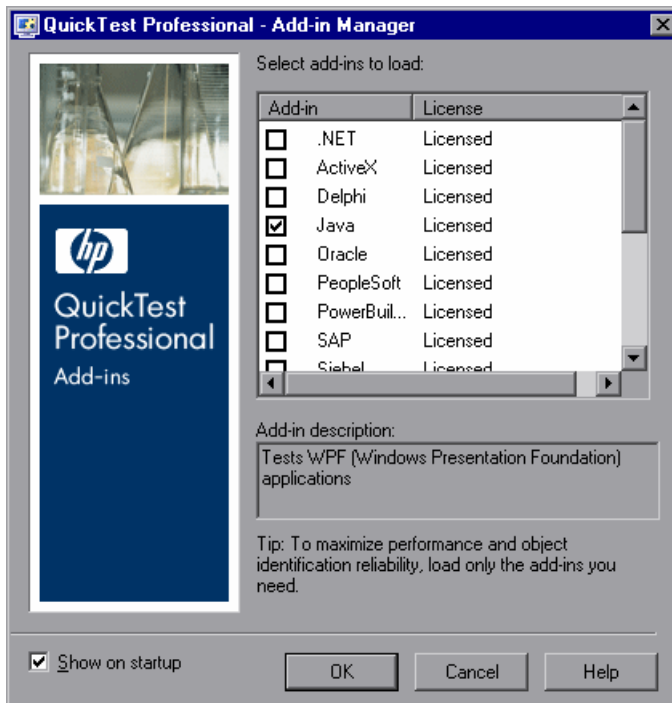
- Starting QuickTest on page 20
- The QuickTest Window on page 23
- Keyword View on page 28
- Expert View on page 29
- Function Library on page 30
- Start Page on page 31
- Active Screen on page 33
- Available Keywords Pane on page 34
- Data Table on page 35
- Debug Viewer Pane on page 36
- Information Pane on page 37
- Missing Resources Pane on page 38
- Process Guidance Panes on page 39
- Resources Pane on page 40
- Test Flow Pane on page 41
- To Do Pane on page 42
- Using QuickTest Commands on page 43
- Browsing the QuickTest Professional Program Folder on page 69
- Viewing Product Information on page 73

Starting QuickTest



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

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

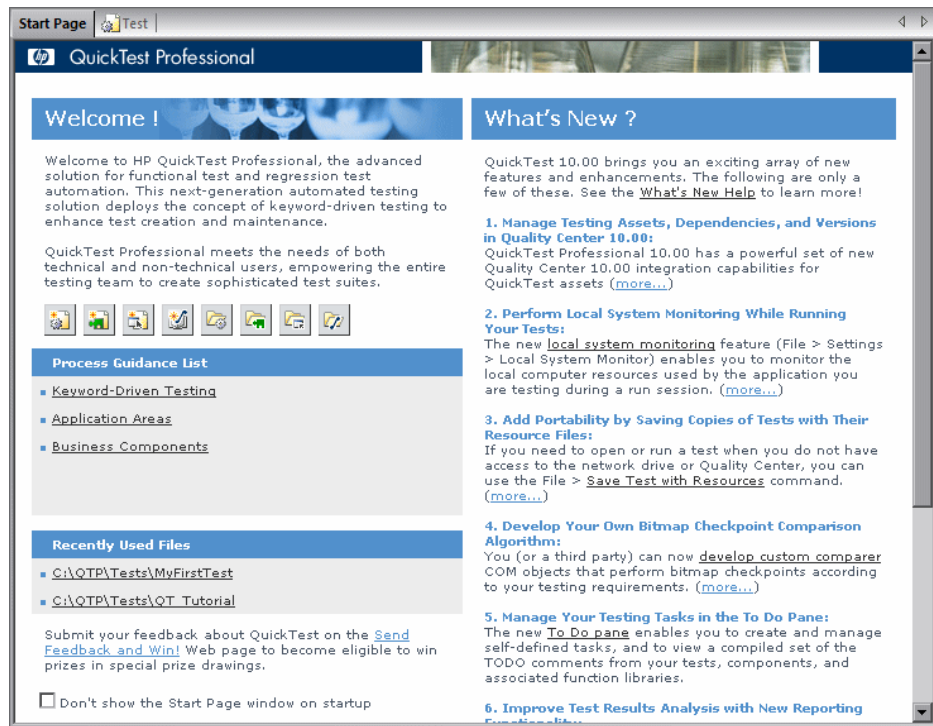


QuickTest remembers the add-ins you load so that the next time you open QuickTest, the add-ins you selected in the previous session are selected by default. For best performance, it is recommended to clear any add-ins that are not needed for a particular session.

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

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

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



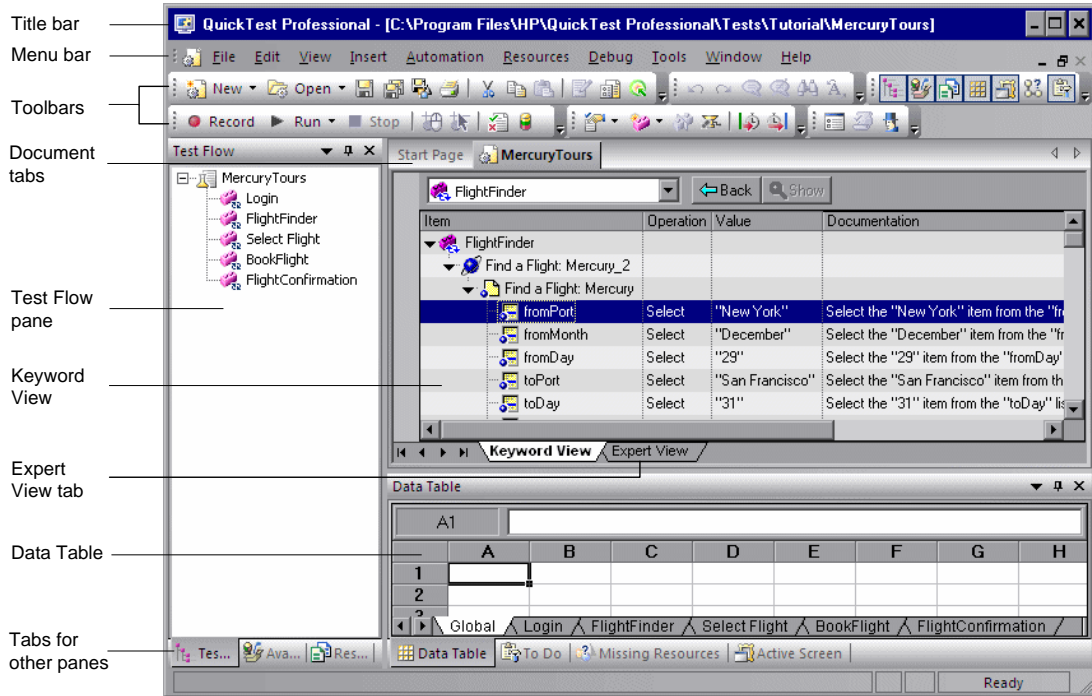
In the Start page, you can:

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

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

The QuickTest Window

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



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

Document Area

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

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

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

Key Elements in the QuickTest Window

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

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

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

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

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

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

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

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

Changing the Appearance of the QuickTest Window

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

To change the appearance of the main QuickTest window:

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

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

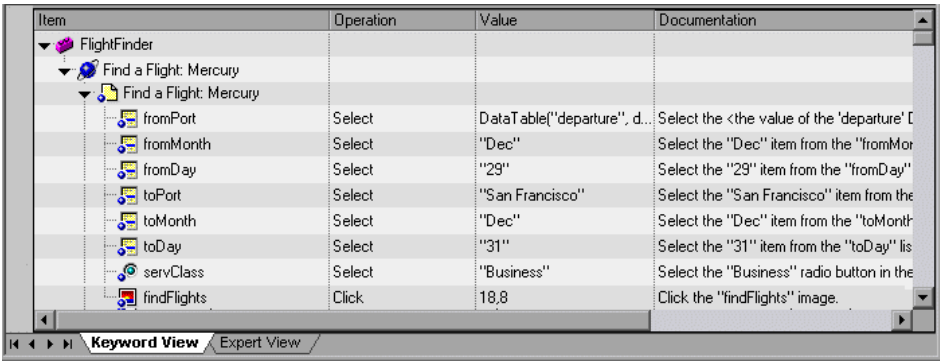
Tip: You can also change the theme used for the Test Results window. For more information, see “Changing the Appearance of the Test Results Window” on page 979.

Keyword View

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

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

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

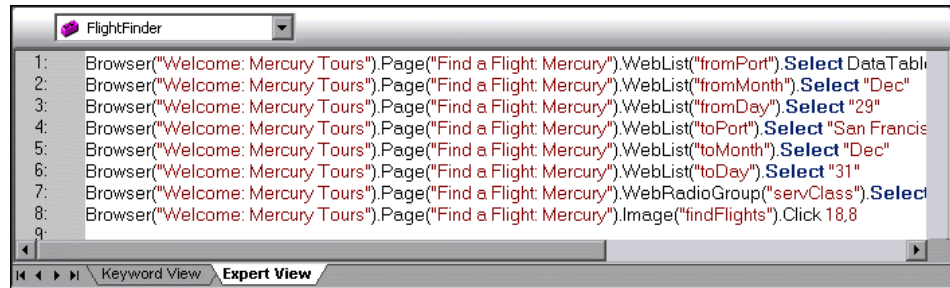


Item	Operation	Value	Documentation
FlightFinder			
Find a Flight: Mercury			
Find a Flight: Mercury			
fromPort	Select	DataTable("departure", d...	Select the <the value of the 'departure' [
fromMonth	Select	"Dec"	Select the "Dec" item from the "fromMor
fromDay	Select	"29"	Select the "29" item from the "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.

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

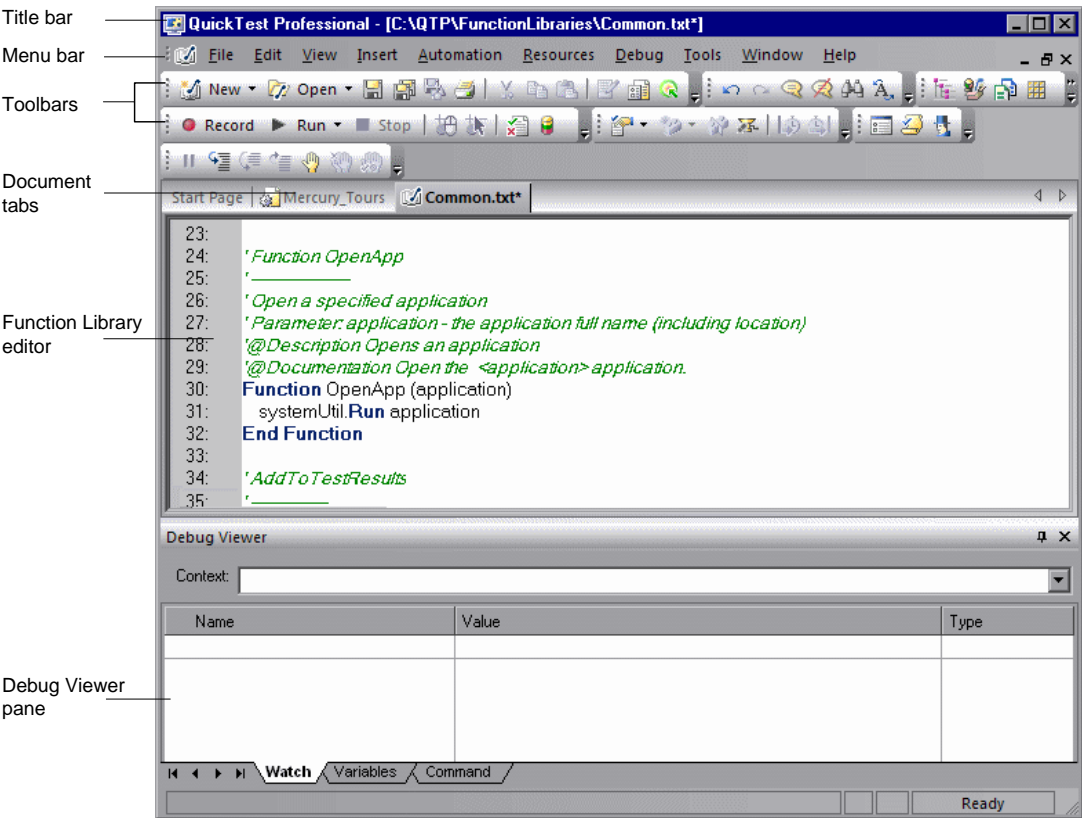
Expert View

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



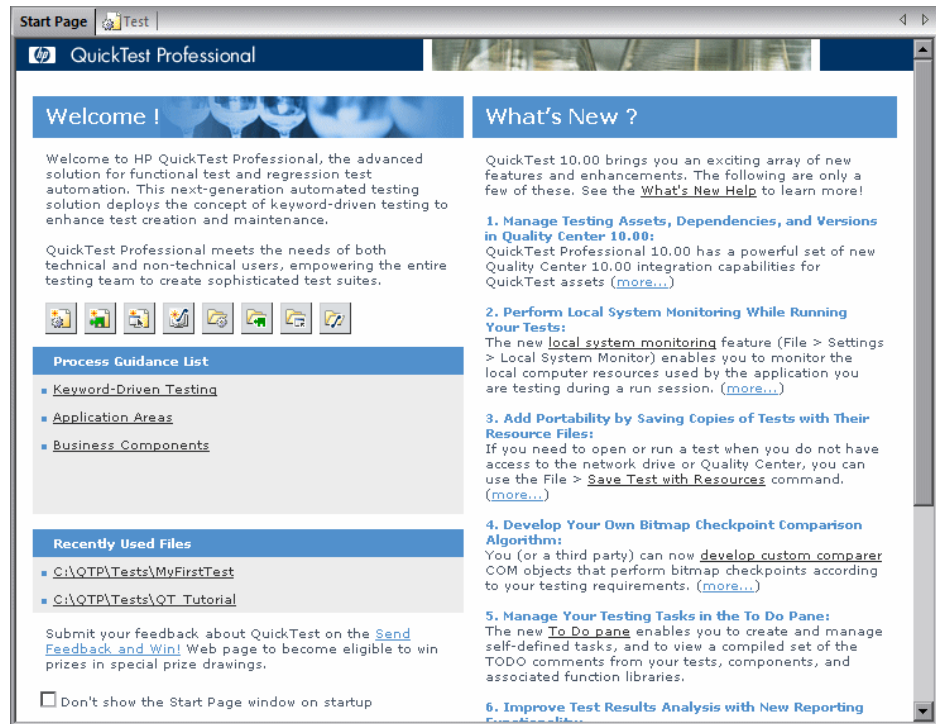
Function Library

QuickTest provides a built-in editor that enables you to create and debug function libraries using the same editing features that are available in the Expert View. Each function library is a separate QuickTest document containing VBScript functions, subroutines, classes, modules, and so forth. Each function library opens in its own window, in addition to the test that is already open. You can work on one or several function libraries at the same time. After you finish editing a function library, you can close it, leaving your QuickTest session open. You can also close all open function libraries simultaneously. For more information, see Chapter 31, “Working with User-Defined Functions and Function Libraries.”











Start Page

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



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

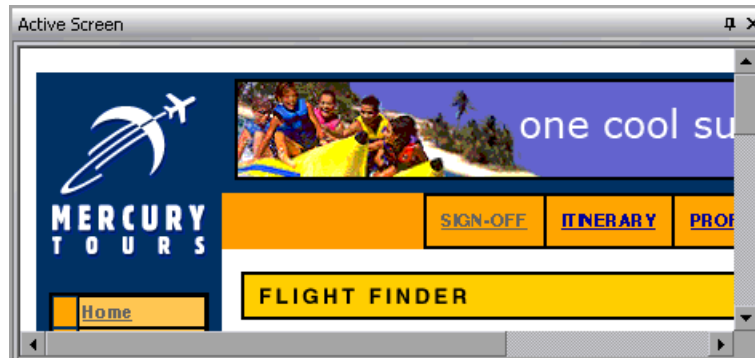
Click	to...
	Open a new test.
	Open a new business component.
	Open a new application area.
	Open a new function library.
	Open an existing test.
	Open an existing business component.
	Open an existing application area.
	Open an existing function library.

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

Active Screen



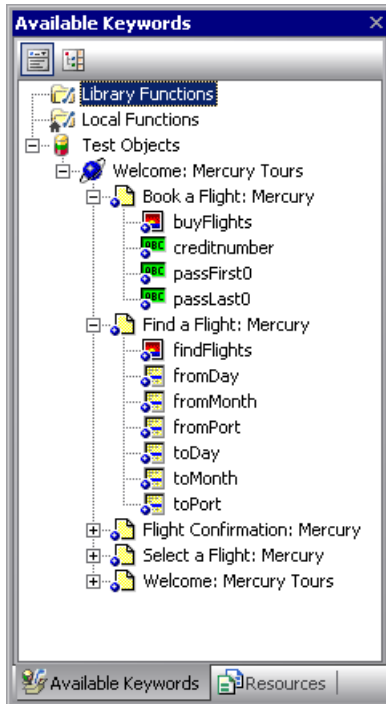
The Active Screen provides a snapshot of your application as it appeared when you performed a certain step during a recording session. Additionally, depending on the Active Screen capture options that you used while recording, the page displayed in the Active Screen can contain detailed property information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



Available Keywords Pane



The Available Keywords pane enables you to drag and drop objects or calls to functions into your test. When you drag and drop an object into your test, QuickTest inserts a step with the default operation for that object. When you drag and drop a function into your test, QuickTest inserts a call to that function. To view the Available Keywords pane, click the **Available Keywords Pane** button or select **View > Available Keywords**.



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

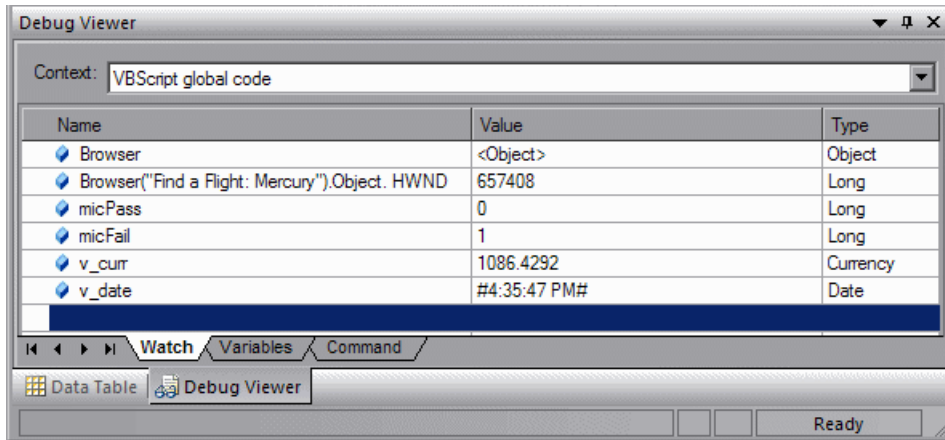
Data Table



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

Debug Viewer Pane

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



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

Watch

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

Variables

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

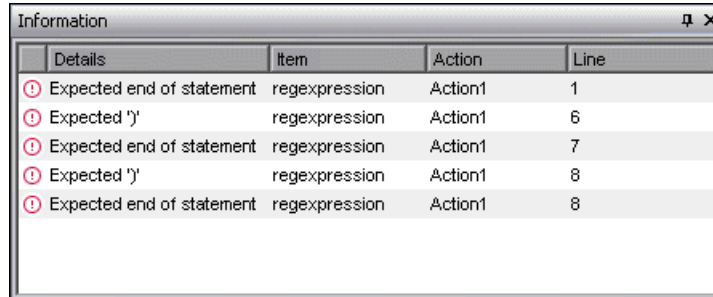
Command

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

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

Information Pane

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



The screenshot shows a window titled "Information" with a table of syntax errors. The table has four columns: Details, Item, Action, and Line. There are five rows of error messages, each preceded by a red circle with an exclamation mark icon.

	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

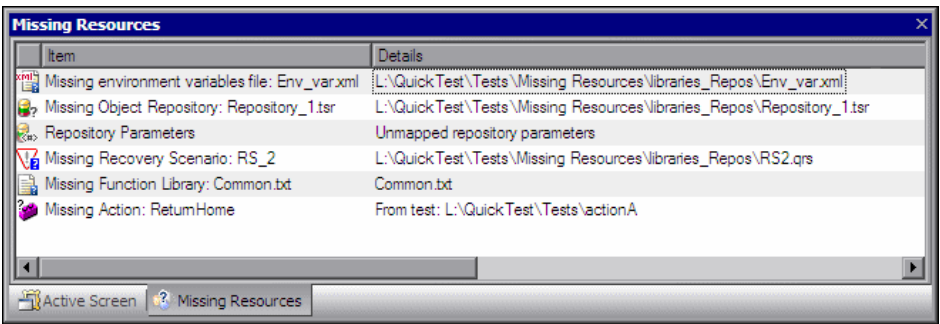
When you switch from the Expert View to the Keyword View, QuickTest automatically checks for syntax errors in your script, and shows them in the Information pane. If the Information pane is not currently displayed, QuickTest automatically opens it when a syntax error is detected.

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 860.

Missing Resources Pane

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

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



The Missing Resources pane contains the following columns.

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

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

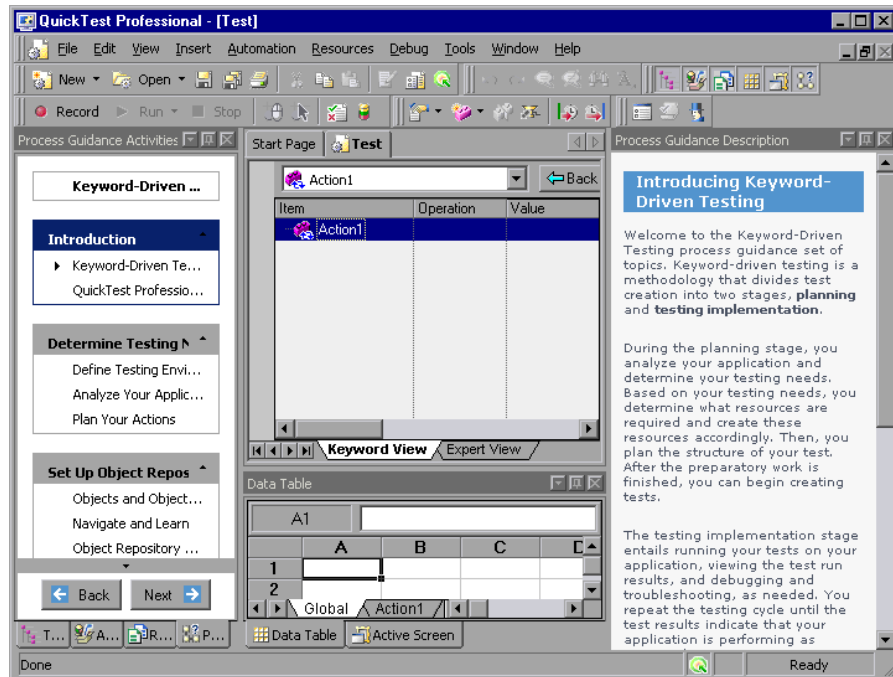
For more information, see “Handling Missing Resources” on page 1179.

Process Guidance Panes

Process guidance is a tool that provides procedures and descriptions on how to best perform specific processes. You use process guidance to learn about new processes and to learn the preferred methodology for performing processes with which you are already familiar.



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

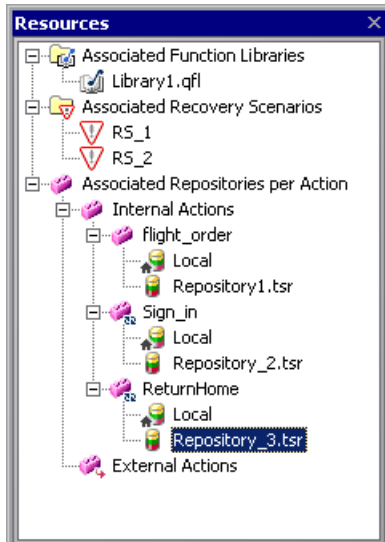


The **Process Guidance Activities** pane (shown on the left) lists the activities that are part of the selected process. The **Process Guidance Description** pane (shown on the right) displays the topic (description), for the selected activity. For more information, see Chapter 43, “Working with Process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

Resources Pane



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

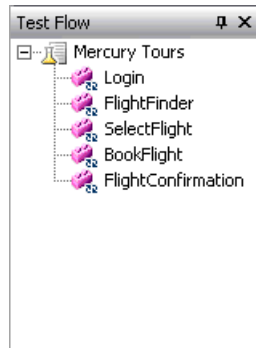


For more information, see “The Resources Pane” on page 1161.

Test Flow Pane

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

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



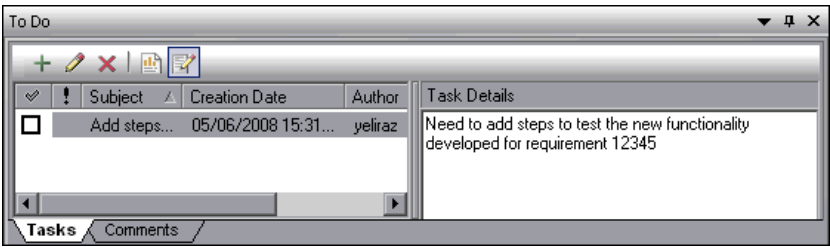
For more information, see “Using the Test Flow Pane” on page 431.

To Do Pane

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



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



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

Using QuickTest Commands

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

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

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

QuickTest Toolbars

This section describes the QuickTest built-in toolbars.

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

Standard Toolbar

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



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

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

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

Automation Toolbar

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



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

Debug Toolbar

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



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

Edit Toolbar

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



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

Insert Toolbar

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



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

Tools Toolbar

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



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

View Toolbar

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



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

Action Toolbar

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



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

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










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







Performing QuickTest Commands



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

File Menu Commands

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

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









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

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








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

Edit Menu Commands

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

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

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










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

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

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

View Menu Commands



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








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



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

Insert Menu Commands

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

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









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


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

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

Automation Menu Commands

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





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

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

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

Resources Menu Commands









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

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

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

Debug Menu Commands




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

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

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

Tools Menu Commands

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

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

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

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

Window Menu Commands

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

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

Help Menu Commands

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

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

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

Command	Shortcut Key	Function
HP QuickTest Professional Software Web Page		<p>Uses your default Web browser to access the HP QuickTest Professional software Web page within the HP corporate Web site. This site provides you with overview information, data sheets, demos and white papers about QuickTest as well as access to other technical resources.</p> <p>The URL is: https://h10078.www1.hp.com/cda/hpms/display/main/hpms_content.jsp?zn=bto&cp=1-11-127-24^1352_4000_100</p>
About QuickTest Professional		Displays information about the installed version of QuickTest Professional.

Data Table Menu Commands


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

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

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

Other QuickTest Commands

You can perform the following special options using shortcut keys:

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

Browsing the QuickTest Professional Program Folder

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

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

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

- **QuickTest Professional Help.** Opens a comprehensive help file containing the *HP QuickTest Professional User Guide*, the *HP QuickTest Professional for Business Process Testing User Guide*, the *HP QuickTest Professional Add-ins Guide*, the *HP QuickTest Professional Object Model Reference* (including the relevant sections for any installed add-ins), *QuickTest Advanced References* (Automation API and XML Schema references), and the *Microsoft VBScript Reference*.
- **Tutorial.** Opens the QuickTest Professional tutorial, which teaches you basic QuickTest skills and shows you how to start testing your applications.
- **Extensibility.** Provides links to the Help for the add-in Extensibility SDKs available with QuickTest Professional 10.00. If you install an extensibility SDK, this program folder may also contain additional items.
- **Sample Applications.** Contains the following links to sample applications that you can use to practice testing with QuickTest:
 - **Flight.** Opens a sample flight reservation Windows application. To access the application, enter any username and the password **mercury**.
 - **Mercury Tours Web site.** Opens a sample flight reservation Web application. This Web application is used as a basis for the QuickTest tutorial. For more information, see the *HP QuickTest Professional Tutorial*.
- **Tools.** Contains the following utilities and tools that assist you with the testing process:

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

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

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

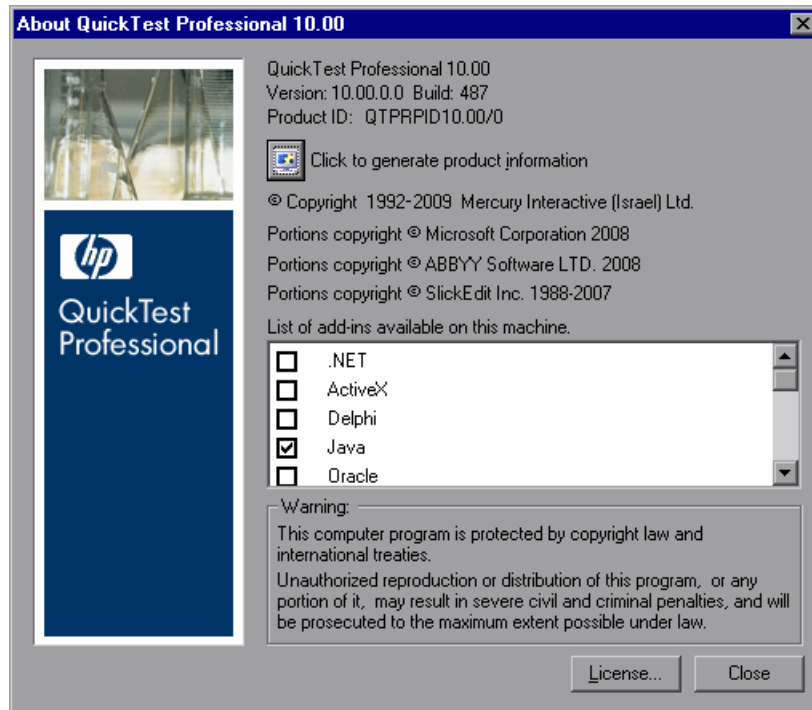
- **Test Results Deletion Tool.** Opens the Test Results Deletion Tool dialog box, which enables you to delete unwanted or obsolete results from your system according to specific criteria that you define. For more information, see “Deleting Results Using the Test Results Deletion Tool” on page 1004.
- **Check for Updates.** Checks online for any available updates to QuickTest Professional. You can choose which updates you want to download and (optionally) install.
- **QuickTest Professional.** Opens the QuickTest Professional application.
- **Readme.** Opens the *HP QuickTest Professional Readme*, which provides the latest news and information on QuickTest Professional and the QuickTest Professional add-ins.
- **Test Results Viewer.** Opens the Test Results window, which enables you to select a test and view information about the steps performed during the run session. For more information, see “The Test Results Window” on page 971.

Viewing Product Information

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

To view the product information:

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



The About QuickTest Professional window displays the following information:

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

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



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

Product Information

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

Add-in Information:


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

Hotfix and Patch Information:

Name
QTP_00557 for HP QuickTest Professional 10.00 QFE

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

© Copyright 1992–2009 Mercury Interactive (Israel) Ltd.



The Product Information window displays the following information:

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

Part II

Working with Test Objects

3

Understanding the Test Object Model

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

This chapter includes:

- About Understanding the Test Object Model on page 79
- Applying the Test Object Model Concept on page 83
- Understanding Object Repository Types on page 89
- Viewing Object Properties and Operations Using the Object Spy on page 97
- The Object Spy Dialog Box on page 100

About Understanding the Test Object Model

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

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

Understanding How QuickTest Learns Objects

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

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

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

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

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

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

QuickTest uses a very similar method when it learns objects.

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

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

Understanding How QuickTest Identifies Objects During the Run Session

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

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

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

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

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

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

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

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

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

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

Applying the Test Object Model Concept

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

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

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

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

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

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

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

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

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

Name	Value
Description properties	
type	submit
name	Search
html tag	INPUT

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

QuickTest displays your step in the Keyword View like this:

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

QuickTest displays your step in the Expert View as follows:

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

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

Understanding Test Object Descriptions

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

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

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	"tutorial"	Enter "tutorial" in the "userName" ed
password	SetSecure	"477cdb71935682eda"	Enter the encrypted string "477cdb7
Sign-In	Click	30,12	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 learn the descriptions of the objects in your application, and you can enable and configure the Smart Identification mechanism. For more information, see Chapter 4, “Configuring Object Identification.”

Understanding Test Object and Native Properties and Operations

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

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

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

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

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

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

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

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

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

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

Understanding Object Repository Types

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

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

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

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

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

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

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

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

Note for users of previous QuickTest versions:

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

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

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

Deciding Whether to Use Local or Shared Object Repositories

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

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

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

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

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

Understanding the Local Object Repository

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

When working with a local object repository:

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

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

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

Understanding the Shared Object Repository

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

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

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

When working with a shared object repository:

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

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

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

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

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

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


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

Viewing Object Properties and Operations Using the Object Spy

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

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

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

- 1 Open your application to the page containing the object on which you want to spy.
- 2  Select **Tools > Object Spy** or click the **Object Spy** toolbar button to open the Object Spy dialog box.
- 3 Select the details you want to view for the object. For more information, see “The Object Spy Dialog Box” on page 100.
- 4 If the objects on which you want to spy have a deep hierarchy, or long property names and values, resize the Object Spy dialog box to view all the information without scrolling.



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

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

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

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

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

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

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


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

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

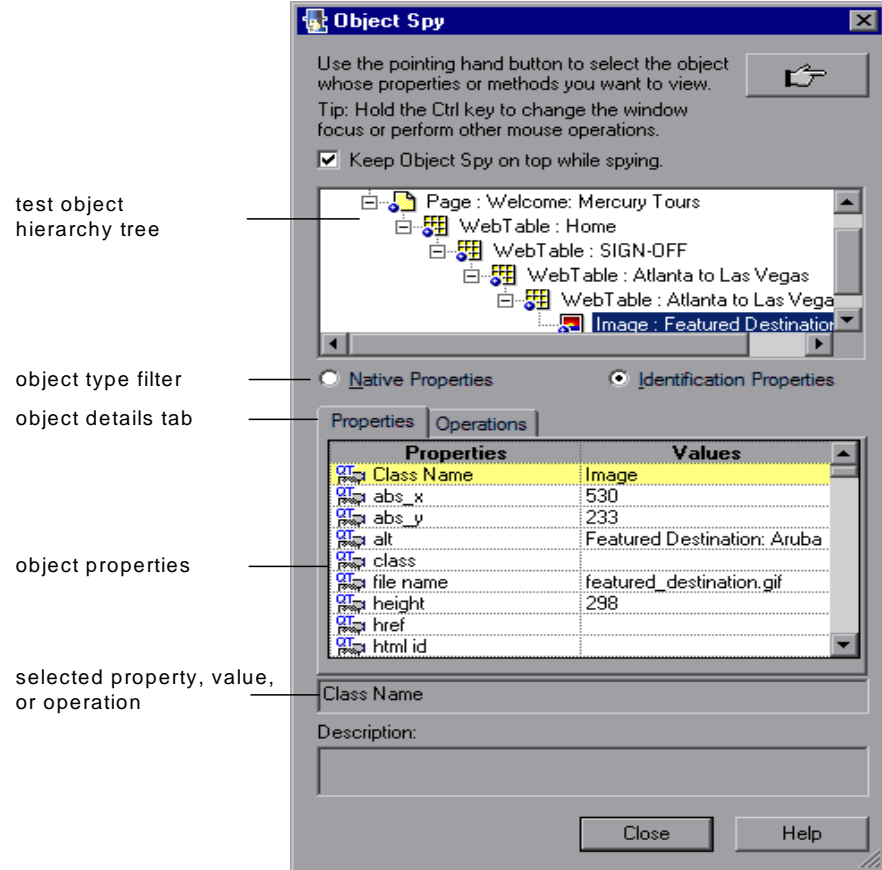
Tips for Working with the Pointing Hand

- You can hold the left CTRL key to change the pointing hand to a standard pointer. You can then change the window focus or perform operations in QuickTest or in your application, such as right-clicking, using the scroll bars, or moving the pointer over an object to display a context menu.
- If the window containing the object you want to select is partially hidden by another window, hold the pointing hand over the partially hidden window for a few seconds until it comes to the foreground. Then point to and click the required object. You can configure the length of time required to bring a window into the foreground using the General pane of the Options dialog box.
- If the window containing the object you want to select is fully hidden by another window, or if a dialog box is hidden behind a window, press the left CTRL key and arrange the windows as needed.
- If the window containing the object you want to select is minimized, you can display it by holding the left CTRL key, right-clicking the application in the Windows task bar, and choosing **Restore** from the context menu.
- If the object you want to select can be displayed only by performing an event (such as right-clicking or moving the pointer over an object to display a context menu), hold the left CTRL key. The pointing hand temporarily turns into a standard pointer and you can perform the event. When the object you want to select is displayed, release the left CTRL key. The pointer becomes a pointing hand again.


The Object Spy Dialog Box

Description	Enables you to view the native properties and operations of any object in an open application, as well as the test object hierarchy, identification properties, and operations of the test object that QuickTest uses to represent that object.
How to Access	<ul style="list-style-type: none">➤ Select the Tools > Object Spy menu command➤ Click the Object Spy toolbar button ➤ Press ALT+T+S <p>You can access the Object Spy dialog box using the methods described above, from any of the following locations:</p> <ul style="list-style-type: none">➤ The QuickTest Window (described on page 23)➤ The Object Repository Window (described on page 183)➤ The Object Repository Manager (described on page 210)
Learn More	<p>Conceptual overview: “Understanding Test Object and Native Properties and Operations” on page 87</p> <p>Primary task: “Viewing Object Properties and Operations Using the Object Spy” on page 97</p> <p>Additional related topics: “Additional References” on page 104</p>

Below is an image of the Object Spy dialog box:



Object Spy Dialog Box Options

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

Option	Description
Native Properties / Native Operations	Select this option to display the native properties or operations of the run-time object associated with the test object selected in the Object Spy test object hierarchy tree. Note that the label changes depending on whether the Properties or Operations tab is selected.
Identification Properties / Test Object Operations	Select this option to display the identification properties or the test object operations of the test object selected in the Object Spy test object hierarchy tree. Note that the label changes depending on whether the Properties or Operations tab is selected.
Properties tab	<p>Displays the native properties or the identification properties of the selected object and the values of the properties.</p> <ul style="list-style-type: none"> ➤ Properties. Displays the property names for the test object that is currently selected in the Object Spy test object hierarchy tree, or the run-time object associated with it. ➤ Values. Displays the property values for the properties listed in the Properties column.
Operations tab	Displays the native operations or test object operations, and their corresponding syntax, for the test object that is currently selected in the Object Spy test object hierarchy tree, or the run-time object associated with it.

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

Additional References

Related Concepts	<ul style="list-style-type: none">➤ “Accessing Native Properties and Operations” on page 887➤ “Retrieving and Setting Identification Property Values” on page 886
------------------	--

4

Configuring Object Identification

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

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

This chapter includes:

- About Configuring Object Identification on page 106
- Understanding the Object Identification Dialog Box on page 107
- Configuring Smart Identification on page 121
- Mapping User-Defined Test Object Classes on page 131

About Configuring Object Identification

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

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

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

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

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

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

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

Understanding the Object Identification Dialog Box

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

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

Notes:

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

For more information, see:

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

Configuring Mandatory and Assistive Properties

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

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

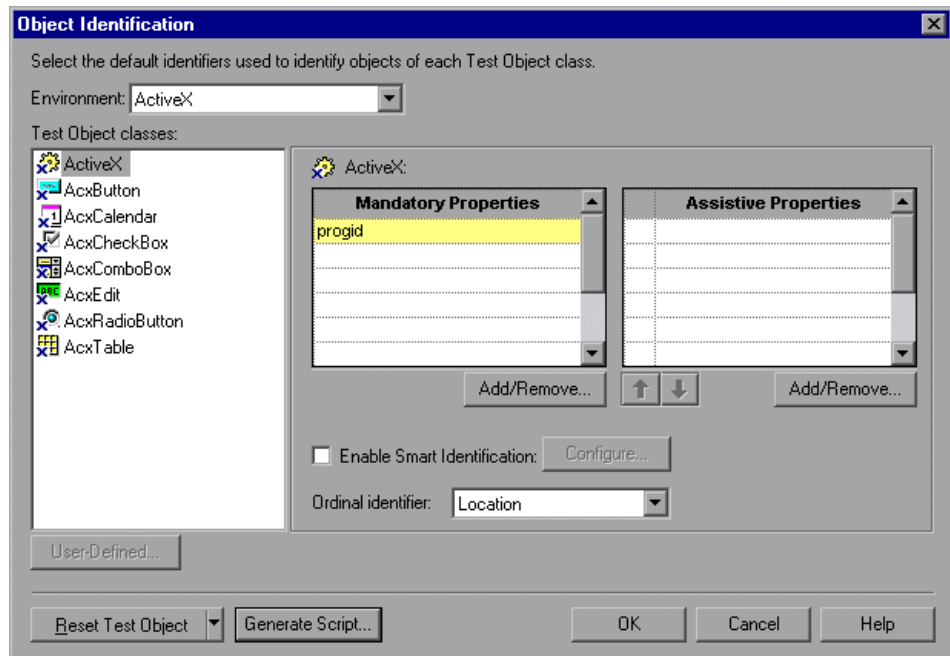
For example, the default mandatory properties for a Web Image object are the **alt**, **html tag**, and **image type** properties. There are no default assistive properties defined. Suppose your Web site contains several space holders for different collections of rotating advertisements. You want to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

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

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

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

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

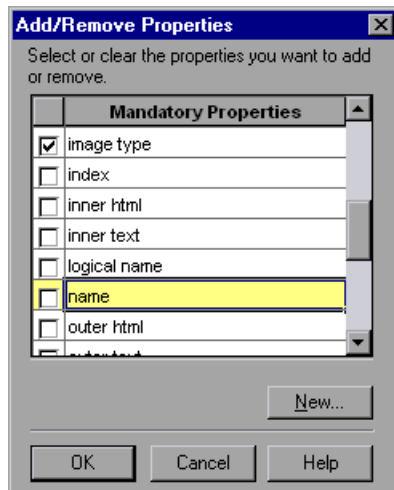


- 2 Select the appropriate environment in the **Environment** list. The test object classes associated with the selected environment are displayed alphabetically in the **Test Object classes** list. (In Standard Windows, the user-defined objects are displayed at the bottom of the list.)

Notes:

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

- 3 In the **Test Object classes** list, select the test object class you want to configure.
- 4 In the **Mandatory Properties** list, click **Add/Remove**. The Add/Remove Properties dialog box for mandatory properties opens.



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

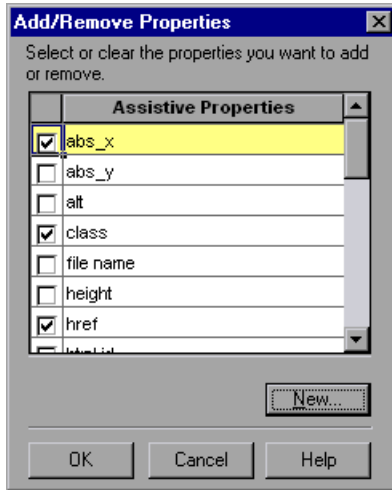
Note: You cannot include the same property in both the mandatory and assistive property lists.

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

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

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

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



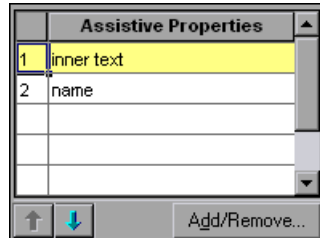
- 8 Select the properties you want to include in the assistive properties list and/or clear the properties you want to remove from the list.

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

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

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

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



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

Selecting an Ordinal Identifier

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

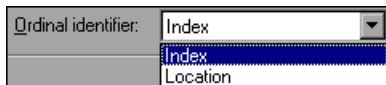
The assigned ordinal property value is a relative value and is accurate only in relation to the other objects displayed when QuickTest learns an object. Therefore, changes in the layout or composition of your application page or screen can cause this value to change, even though the object itself has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

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

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

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

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



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

Identifying an Object Using the Index Property

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

Index property values are object-specific. Therefore, if you use `Index:=3` to describe a `WebEdit` test object, QuickTest searches for the fourth `WebEdit` object in the page. However, if you use `Index:=3` to describe a `WebElement` object, QuickTest searches for the fourth `Web` object on the page—regardless of the type—because the `WebElement` object applies to all `Web` objects.

For example, suppose a page contains the following objects:

- an image with the name `Apple`
- an image with the name `UserName`
- a `WebEdit` object with the name `UserName`
- an image with the name `Password`
- a `WebEdit` object with the name `Password`

The following statement refers to the third item in the list, as this is the first `WebEdit` object on the page with the name `UserName`:

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

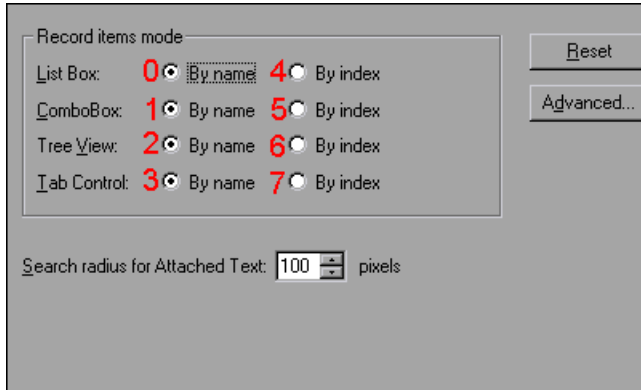
In contrast, the following statement refers to the second item in the list, as that is the first object of any type (`WebElement`) with the name `UserName`:

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

Identifying an Object Using the Location Property

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

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



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

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

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

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

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

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

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

Identifying an Object Using the CreationTime Property

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

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

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

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

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

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

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

Enabling and Disabling Smart Identification

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

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

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

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

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

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

Restoring Default Object Identification Settings for Test Objects

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

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

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

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

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

Generating Automation Scripts for Your Object Identification Settings

You can click the **Generate Script** button to generate an automation script containing the current object identification settings. For more information, see “Automating QuickTest Operations” on page 1403 or the *QuickTest Professional Automation Object Model Reference* (**Help > QuickTest Professional Help > HP QuickTest Professional Advanced References > HP QuickTest Professional Automation Object Model**).

Configuring Smart Identification

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

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

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

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

The Smart Identification mechanism uses two types of properties:

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

Understanding the Smart Identification Process

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

- 1** QuickTest "forgets" the learned test object description and creates a new **object candidate** list containing the objects (within the object's parent object) that match all of the properties defined in the Base Filter Properties list.
- 2** QuickTest filters out any object in the object candidate list that does not match the first property listed in the Optional Filter Properties list. The remaining objects become the new object candidate list.
- 3** QuickTest evaluates the new object candidate list:
 - If the new object candidate list still has more than one object, QuickTest uses the new (smaller) object candidate list to repeat step 2 for the next optional filter property in the list.
 - If the new object candidate list is empty, QuickTest ignores this optional filter property, returns to the previous object candidate list, and repeats step 2 for the next optional filter property in the list.
 - If the object candidate list contains exactly one object, then QuickTest concludes that it has identified the object and performs the statement containing the object.
- 4** QuickTest continues the process described in steps 2 and 3 until it either identifies one object, or runs out of optional filter properties to use.

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

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

Reviewing Smart Identification Information in the Test Results

If the learned description does not enable QuickTest to identify a specified object in a step, and a Smart Identification definition is defined (and enabled) for the object, then QuickTest tries to identify the object using the Smart Identification mechanism.

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

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

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

Walking Through a Smart Identification Example

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

Suppose you have the following statement in your test:

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

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

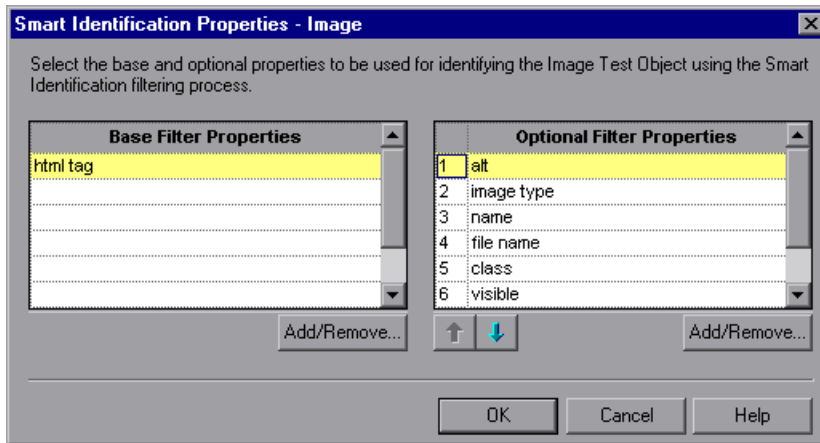
Name	Value
[-] Description properties	
image type	Image Button
html tag	INPUT
alt	Login

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

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

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

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



The learned values are as follows:

Base Filter Properties:

Property	Value
html tag	INPUT

Optional Filter Properties:

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

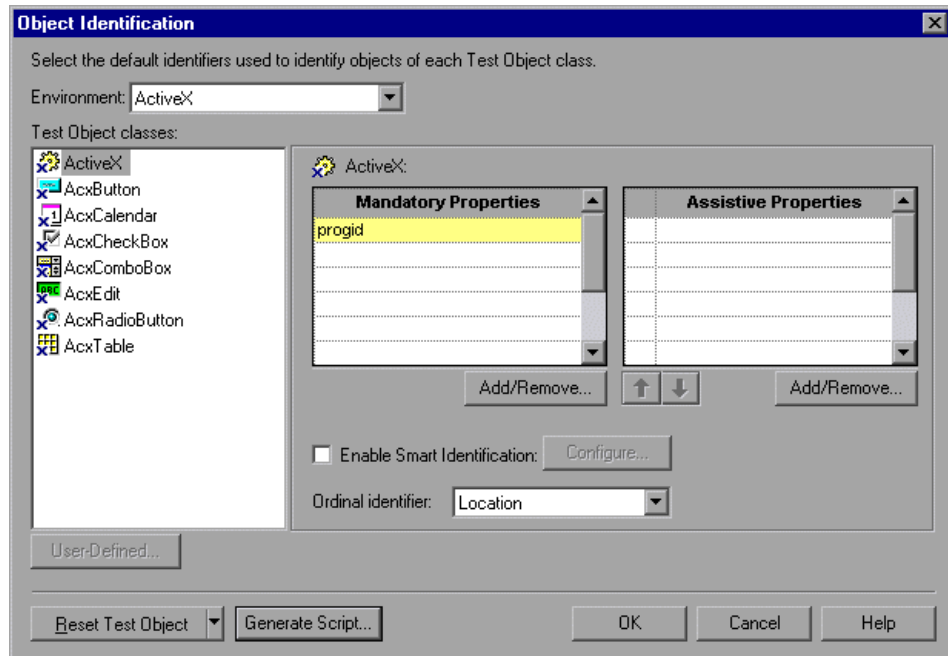
- 2** QuickTest begins the Smart Identification process by identifying the five objects on the Mercury Tours page that match the base filter properties definition (**html tag** = INPUT). QuickTest considers these to be the object candidates and begins checking the object candidates against the **Optional Filter Properties** list.
- 3** QuickTest checks the **alt** property of each of the object candidates, but none have the **alt** value: Login, so QuickTest ignores this property and moves on to the next one.
- 4** QuickTest checks the **image type** property of the each of the object candidates, but none have the **image type** value: Image Button, so QuickTest ignores this property and moves on to the next one.
- 5** QuickTest checks the **name** property of each of the object candidates, and finds that two of the objects (both the basic and VIP Login buttons) have the name: login. QuickTest filters out the other three objects from the list, and these two login buttons become the new object candidates.
- 6** QuickTest checks the **file name** property of the two remaining object candidates. Only one of them has the file name login.gif, so QuickTest correctly concludes that it has found the Login button and clicks it.

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

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

To configure Smart Identification properties:

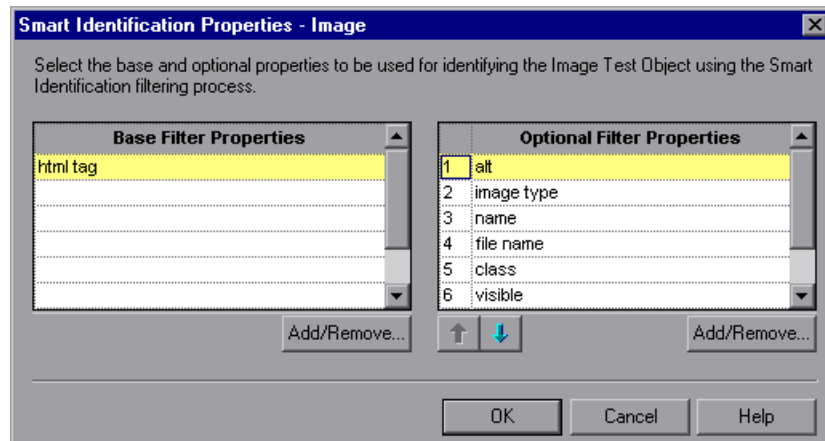
- 1 Select **Tools > Object Identification**. The Object Identification dialog box opens.



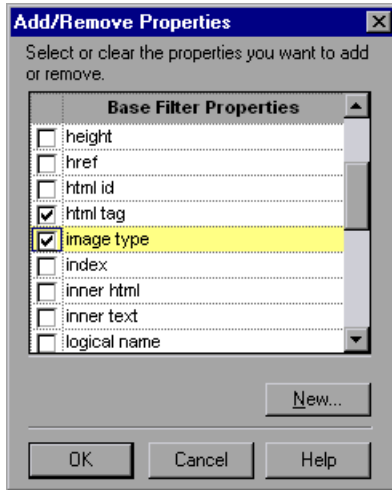
- 2 Select the appropriate environment in the **Environment** list. The test object classes associated with the selected environment are displayed in the **Test Object classes** list.

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

- 3 Select the test object class you want to configure.
- 4 Click the **Configure** button next to the **Enable Smart Identification** check box. The **Configure** button is enabled only when the **Enable Smart Identification** option is selected. The Smart Identification Properties dialog box opens.



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



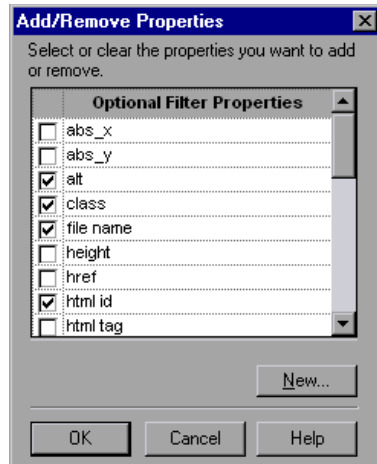
- 6 Select the properties you want to include in the **Base Filter Properties** list and/or clear the properties you want to remove from the list.

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

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

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

- 7 Click **OK** to close the Add/Remove Properties dialog box. The updated set of base filter properties is displayed in the **Base Filter Properties** list.
- 8 In the **Optional Filter Properties** list, click **Add/Remove**. The Add/Remove Properties dialog box for optional filter properties opens.



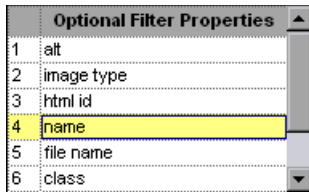
- 9 Select the properties you want to include in the **Optional Filter Properties** list and/or clear the properties you want to remove from the list.

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

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

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

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



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

Mapping User-Defined Test Object Classes

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

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

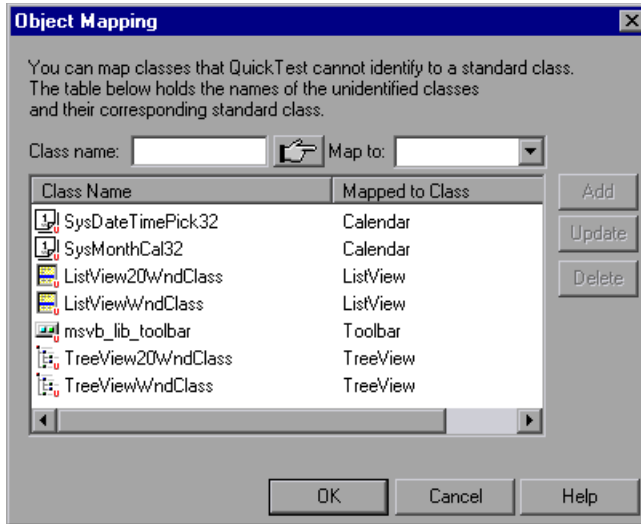
Notes:

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

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

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

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



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

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

- 5 In the **Map to** box, select the standard object class to which you want to map your user-defined object class and click **Add**. The class name and mapping is added to the object mapping list.
- 6 If you want to map additional objects to standard classes, repeat steps 4 to 5 for each object.

- 7** Click **OK**. The Object Mapping dialog box closes and your object is added to the list of Standard Windows test object classes as a user-defined test object. Note that your object has an icon with a red U in the lower-right corner, identifying it as a user-defined class.
- 8** Configure the object identification settings for your user defined object class just as you would any other object class. For more information, see “Configuring Mandatory and Assistive Properties” on page 108, and “Configuring Smart Identification” on page 121.

To modify an existing mapping:

- 1** In the Object Mapping dialog box, select the class you want to modify from the object mapping list. The class name and current mapping are displayed in the Class name and Map to boxes.
- 2** Select the standard object class to which you want to map the selected user-defined class and click **Update**. The class name and mapping is updated in the object mapping list.
- 3** Click **OK** to close the Object Mapping dialog box.

To delete an existing mapping:

- 1** In the Object Mapping dialog box, select the class you want to delete from the object mapping list.
- 2** Click **Delete**. The class name and mapping is deleted from the object mapping list in the Object Mapping dialog box.
- 3** Click **OK**. The Object Mapping dialog box closes and the class name is deleted from the Standard Windows test object classes list in the Object Identification dialog box.

Tips for Using the Pointing Hand

- You can hold the left CTRL key to change the pointing hand to a standard pointer. You can then change the window focus or perform operations in QuickTest or in your application, such as right-clicking, using the scroll bars, or moving the pointer over an object to display a context menu.
- If the window containing the object you want to select is partially hidden by another window, hold the pointing hand over the partially hidden window for a few seconds until it comes to the foreground. Then point to and click the required object. You can configure the length of time required to bring a window into the foreground using the General pane of the Options dialog box.
- If the window containing the object you want to select is fully hidden by another window, or if a dialog box is hidden behind a window, press the left CTRL key and arrange the windows as needed.
- If the window containing the object you want to select is minimized, you can display it by holding the left CTRL key, right-clicking the application in the Windows task bar, and choosing **Restore** from the context menu.
- If the object you want to select can be displayed only by performing an event (such as right-clicking or moving the pointer over an object to display a context menu), hold the left CTRL key. The pointing hand temporarily turns into a standard pointer and you can perform the event. When the object you want to select is displayed, release the left CTRL key. The pointer becomes a pointing hand again.

5

Managing Test Objects in Object Repositories

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

This chapter includes:

- Adding Test Objects to a Local or Shared Object Repository on page 136
- Copying, Pasting, and Moving Objects in the Object Repository on page 150
- Deleting Objects from the Object Repository on page 153
- Locating Objects on page 154
- Maintaining Identification Properties on page 162

Adding Test Objects to a Local or Shared Object Repository

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

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

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

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

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

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

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

Tips:

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

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

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

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

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

1 Perform one of the following:



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



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

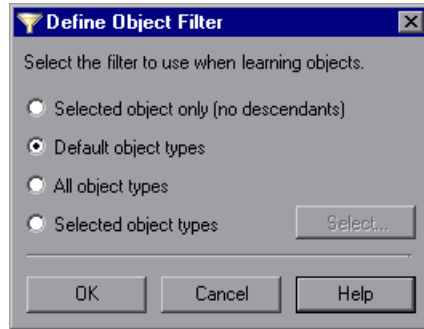
QuickTest and the Object Repository window or Object Repository Manager are hidden, and the pointer changes into a pointing hand. For more information on using the pointing hand, see “Tips for Using the Pointing Hand” on page 141.

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

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

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

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

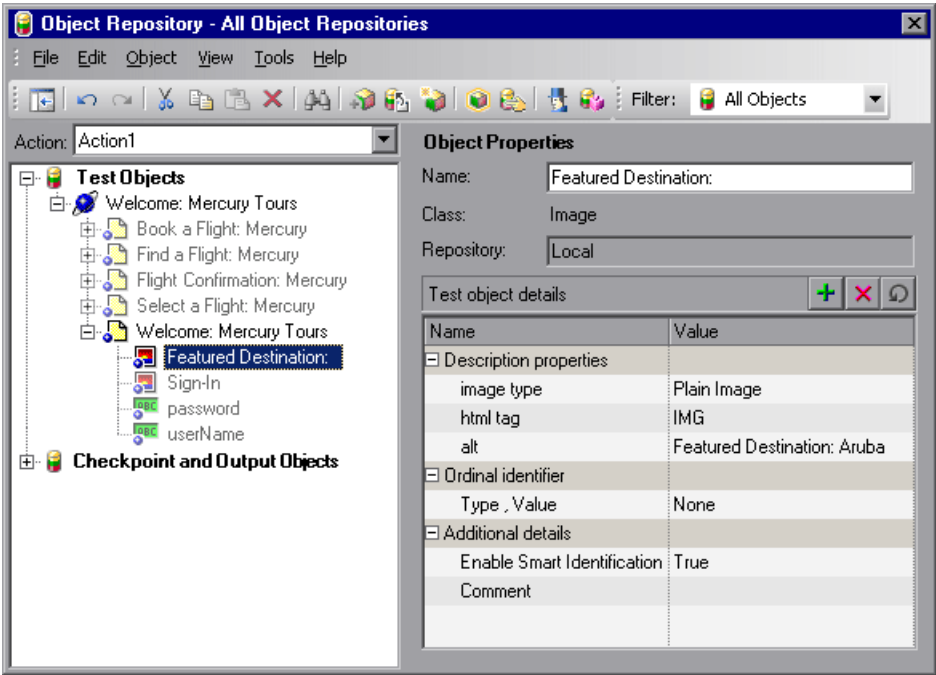


You can choose from the following options:

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

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

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



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

Tips for Using the Pointing Hand

- You can hold the left CTRL key to change the pointing hand to a standard pointer. You can then change the window focus or perform operations in QuickTest or in your application, such as right-clicking, using the scroll bars, or moving the pointer over an object to display a context menu.
- If the window containing the object you want to select is partially hidden by another window, hold the pointing hand over the partially hidden window for a few seconds until it comes to the foreground. Then point to and click the required object. You can configure the length of time required to bring a window into the foreground using the General pane of the Options dialog box.
- If the window containing the object you want to select is fully hidden by another window, or if a dialog box is hidden behind a window, press the left CTRL key and arrange the windows as needed.
- If the window containing the object you want to select is minimized, you can display it by holding the left CTRL key, right-clicking the application in the Windows task bar, and choosing **Restore** from the context menu.
- If the object you want to select can be displayed only by performing an event (such as right-clicking or moving the pointer over an object to display a context menu), hold the left CTRL key. The pointing hand temporarily turns into a standard pointer and you can perform the event. When the object you want to select is displayed, release the left CTRL key. The pointer becomes a pointing hand again.

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

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

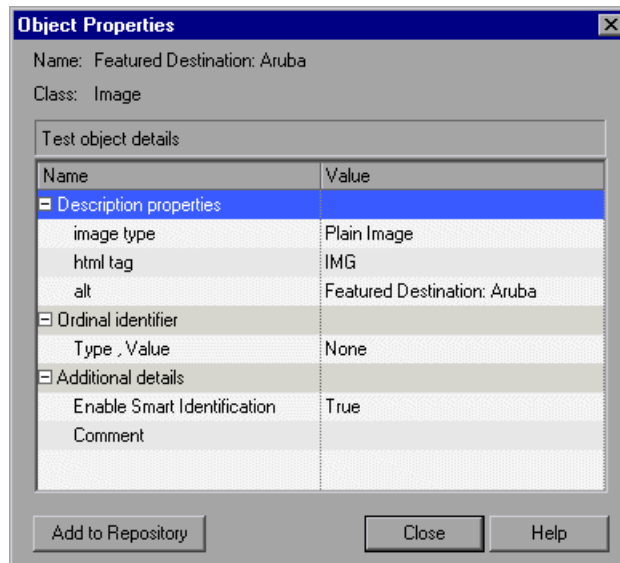
To add test objects to the object repository using the Active Screen, the Active Screen must contain information for the object you want to add. You can control how much information is captured in the Active Screen in the Active Screen node of the Options dialog box. For more information, see “Setting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

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

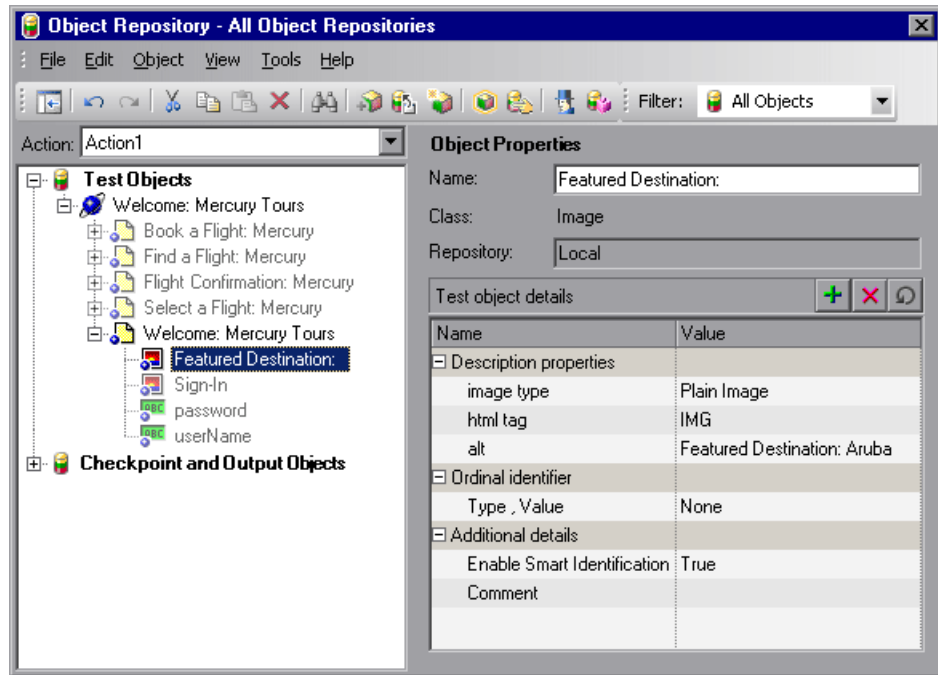
To add a test object to the object repository using the View/Add Object option from the Active Screen:



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



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



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

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



- 1 If the Active Screen is not displayed, select **View > Active Screen** or click the Active Screen toolbar button to display it.
- 2 Select a step in your test whose Active Screen contains the object for which you want to add a step.

- 3 In the Active Screen, right-click the object for which you want to add a step and select the type of step you want to insert (checkpoint, output value, Step Generator, and so forth).
- 4 If the location you clicked is associated with more than one object, the Object Selection dialog box opens. Select the object for which you want to add a step, and click **OK**.

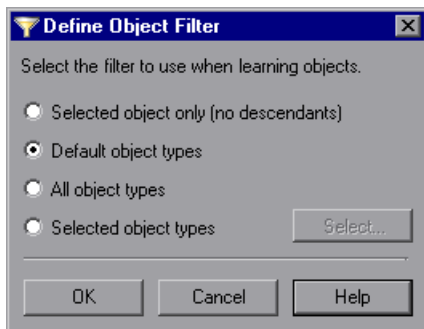
The appropriate dialog box opens, enabling you to configure your preferences for the step you want to insert.

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

Understanding the Define Object Filter Dialog Box

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

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



You can choose from the following options:

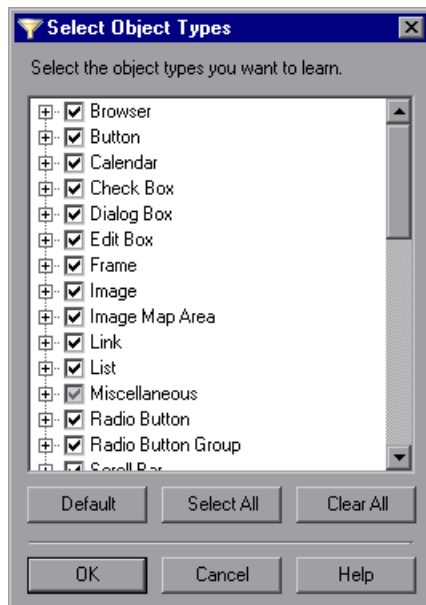
- **Selected object only (no descendants).** Adds to the object repository the previously selected object's properties and values, without its descendant objects.
- **Default object types.** Adds to the object repository the previously selected object's properties and values, with the properties and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

Understanding the Select Object Types Dialog Box

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

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

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



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

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

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

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

Defining New Test Objects

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

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

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

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

To define a new test object:

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



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

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

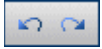
Notes:

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

- 4 In the **Class** box, select the class of the test object you want to define.
- 5 In the **Name** box, enter a name for the new test object. After you enter a name, the **Test object details** area is enabled.
- 6 In the **Test object details** area, define the properties and values for your test object. The **Test object details** area automatically contains the mandatory properties defined for the object class in the Object Identification dialog box. You can add or remove properties as required, and define values for the properties. For more information, see “Maintaining Identification Properties” on page 162.
- 7 Click **Add**. The new test object is added to the local object repository in the selected location.
- 8 Repeat step 3 to step 7 to define additional test objects, or click **Close** to close the Define New Test Object dialog box.

Copying, Pasting, and Moving Objects in the Object Repository

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



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

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

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

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

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

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

To move or copy an object without its child objects:

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

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



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

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

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

To copy objects from one shared object repository to another:

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

To move objects from one shared object repository to another:

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

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

Guidelines for Copying, Pasting, and Moving Objects

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

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

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

Deleting Objects from the Object Repository

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

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

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

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

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

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

To delete an object from the object repository:

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



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

Locating Objects

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

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

Finding Objects in an Object Repository

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

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

Notes:

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

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

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

A screenshot of the 'Find & Replace' dialog box. The dialog has a title bar with a magnifying glass icon and the text 'Find & Replace'. It contains several input fields and buttons. The 'Find' section has a text input field and a 'Find Next' button. The 'Object name' section has a text input field. The 'Object type' section has a dropdown menu with 'All' selected. The 'Object class' section has a dropdown menu with 'All' selected. The 'Property name' section has a text input field. The 'Property value' section has a text input field. The 'Replace' section has a text input field and a 'Replace' button. The 'New property value' section has a text input field and a 'Replace All' button. The 'Options' section has two checkboxes: 'Match case' and 'Match whole word', both of which are unchecked. There are also two radio buttons for 'Direction': 'Up' and 'Down', with 'Down' selected. At the bottom, there are 'Close' and 'Help' buttons.

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

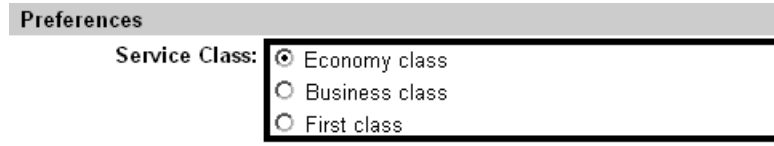
You can select a test object in your object repository and highlight it in the application you are testing. When you choose to highlight a test object, QuickTest indicates the selected object's location in your application by temporarily showing a frame around the object and causing it to flash briefly.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

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

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

To highlight an object in your application:

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



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

Locating a Test Object in the Object Repository

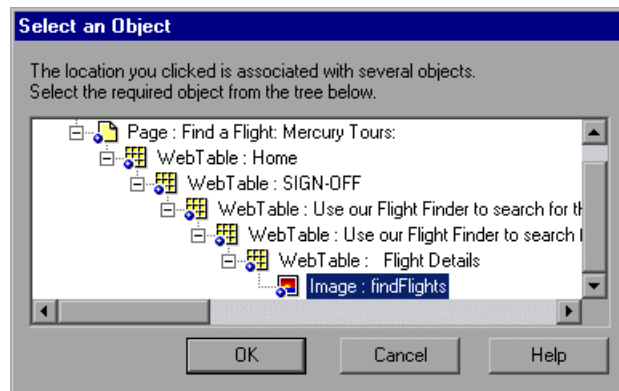
You can select an object in the application you are testing and highlight the test object in the object repository.

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

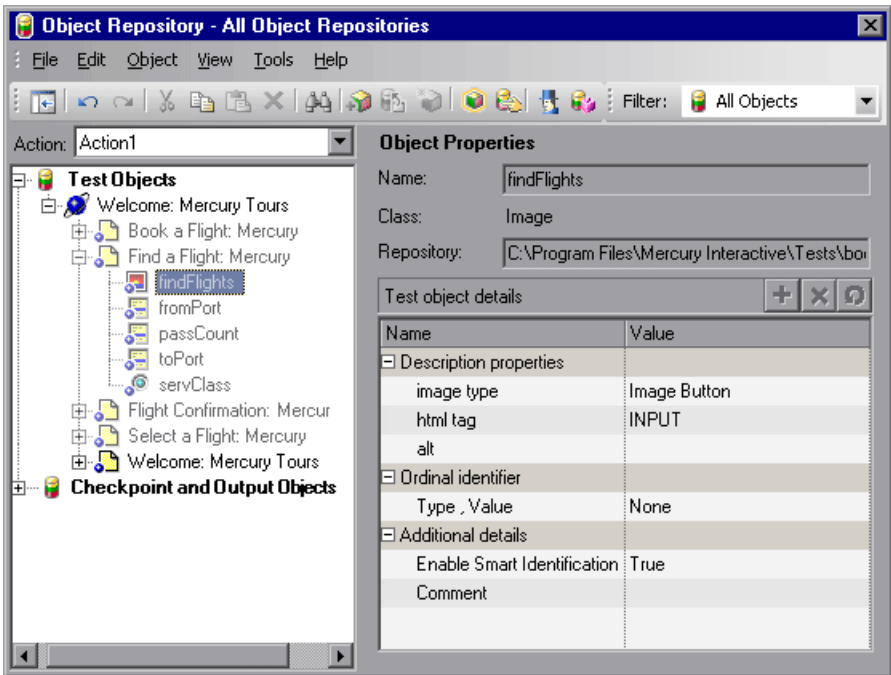
To locate an object in the object repository:

- 1 Make sure your application is open to the correct window or page.
- 2 Click the **Locate in Repository** button or select **View > Locate in Repository**. QuickTest is hidden, and the pointer changes into a pointing hand.
- 3 Use the pointing hand to click on the required object in your application. For more information on using the pointing hand, see “Tips for Using the Pointing Hand” on page 161.

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



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



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

Tips for Using the Pointing Hand

- You can hold the left CTRL key to change the pointing hand to a standard pointer. You can then change the window focus or perform operations in QuickTest or in your application, such as right-clicking, using the scroll bars, or moving the pointer over an object to display a context menu.
- If the window containing the object you want to select is partially hidden by another window, hold the pointing hand over the partially hidden window for a few seconds until it comes to the foreground. Then point to and click the required object. You can configure the length of time required to bring a window into the foreground using the General pane of the Options dialog box.
- If the window containing the object you want to select is fully hidden by another window, or if a dialog box is hidden behind a window, press the left CTRL key and arrange the windows as needed.
- If the window containing the object you want to select is minimized, you can display it by holding the left CTRL key, right-clicking the application in the Windows task bar, and choosing **Restore** from the context menu.
- If the object you want to select can be displayed only by performing an event (such as right-clicking or moving the pointer over an object to display a context menu), hold the left CTRL key. The pointing hand temporarily turns into a standard pointer and you can perform the event. When the object you want to select is displayed, release the left CTRL key. The pointer becomes a pointing hand again.

Maintaining Identification Properties

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

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

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

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

Specifying or Modifying Property Values

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

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

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

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

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



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

To specify an identification property value:

- 1** In the Object Repository window or Manager, select the test object whose property value you want to specify.
- 2** In the **Test object details** area, click in the value cell for the required property.

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

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




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

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






- If you want to specify a constant value, enter it in the value cell.



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

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

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

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

Updating Identification Properties from an Object in Your Application

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


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

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

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

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

To update identification properties from an object in your application:

- 1** In the object repository tree, select the test object whose description you want to update.
-  **2** Select **Object > Update from Application** or click the **Update from Application** button. QuickTest is hidden, and the pointer changes into a pointing hand. For more information on using the pointing hand, see “Tips for Using the Pointing Hand” on page 167.
- 3** Find the object in your application whose properties you want to update in the object repository and click it. You must choose an object of the same object class as the test object you selected in the object repository tree.

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

Tips for Using the Pointing Hand

- You can hold the left CTRL key to change the pointing hand to a standard pointer. You can then change the window focus or perform operations in QuickTest or in your application, such as right-clicking, using the scroll bars, or moving the pointer over an object to display a context menu.
- If the window containing the object you want to select is partially hidden by another window, hold the pointing hand over the partially hidden window for a few seconds until it comes to the foreground. Then point to and click the required object. You can configure the length of time required to bring a window into the foreground using the General pane of the Options dialog box.
- If the window containing the object you want to select is fully hidden by another window, or if a dialog box is hidden behind a window, press the left CTRL key and arrange the windows as needed.
- If the window containing the object you want to select is minimized, you can display it by holding the left CTRL key, right-clicking the application in the Windows task bar, and choosing **Restore** from the context menu.
- If the object you want to select can be displayed only by performing an event (such as right-clicking or moving the pointer over an object to display a context menu), hold the left CTRL key. The pointing hand temporarily turns into a standard pointer and you can perform the event. When the object you want to select is displayed, release the left CTRL key. The pointer becomes a pointing hand again.


Restoring Default Mandatory Properties for a Test Object

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

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

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

To restore the mandatory property set:

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

Renaming Test Objects

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

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

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

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

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

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

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

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

To rename a test object:

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

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

Adding Properties to a Test Object Description

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

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

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

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

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

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

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



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

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

To add properties to a test object description:

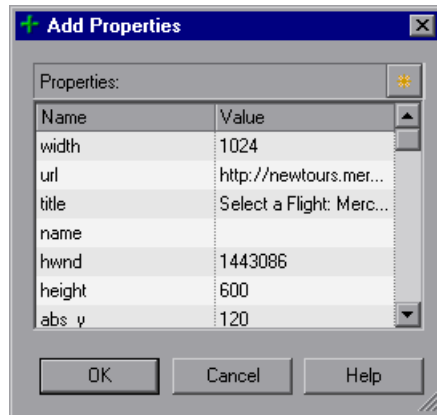
- 1 In the object repository tree of the Object Repository window or Manager, select the test object whose description you want to modify.
- 2 In the **Test object details** area, click the **Add description properties** button.




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

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

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



Notes:

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

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

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

Defining New Identification Properties

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

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

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



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

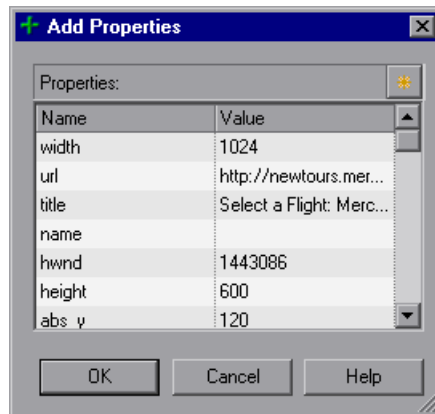
To define a new identification property:

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



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

The Add Properties dialog box opens.





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

The image shows a standard Windows-style dialog box titled "New Property". It contains two text input fields, one labeled "Property name:" and the other "Property value:". Below these fields are three buttons: "OK", "Cancel", and "Help". The dialog box has a blue title bar and a gray border.

- 4 Specify a valid identification property:
 - **Property name.** Enter the property name.
 - **Property value.** Enter the value for the property.

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

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

Removing Properties from a Test Object Description

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

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

To remove a property from a test object description:

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

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



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

Specifying Ordinal Identifiers

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

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

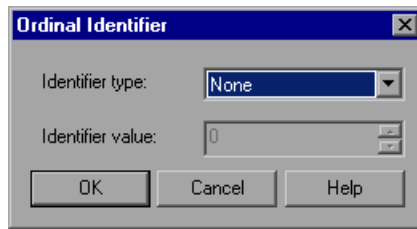
For more information on ordinal identifiers, see “Selecting an Ordinal Identifier” on page 113.

To specify an ordinal identifier:

- 1** In the object repository tree of the Object Repository window or Manager, select the test object whose ordinal identifier you want to specify.
- 2** In the **Test object details** area, click in the cell to the right of the **Type, Value** cell under the **Ordinal identifier** row.

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

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



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

- 5** In the **Identifier value** box, enter the numeric value of the ordinal identifier.
- 6** Click **OK**. The ordinal identifier appears in the relevant row of the **Test object details** area for the selected object.

6

Using Object Repositories in Your Test

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

This chapter includes:

- Understanding the Object Repository Window on page 182
- The Object Properties Dialog Box on page 197
- Managing Shared Object Repository Associations on page 199
- Mapping Repository Parameter Values on page 202
- Working with Test Objects During a Run Session on page 206

Understanding the Object Repository Window

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

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


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

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

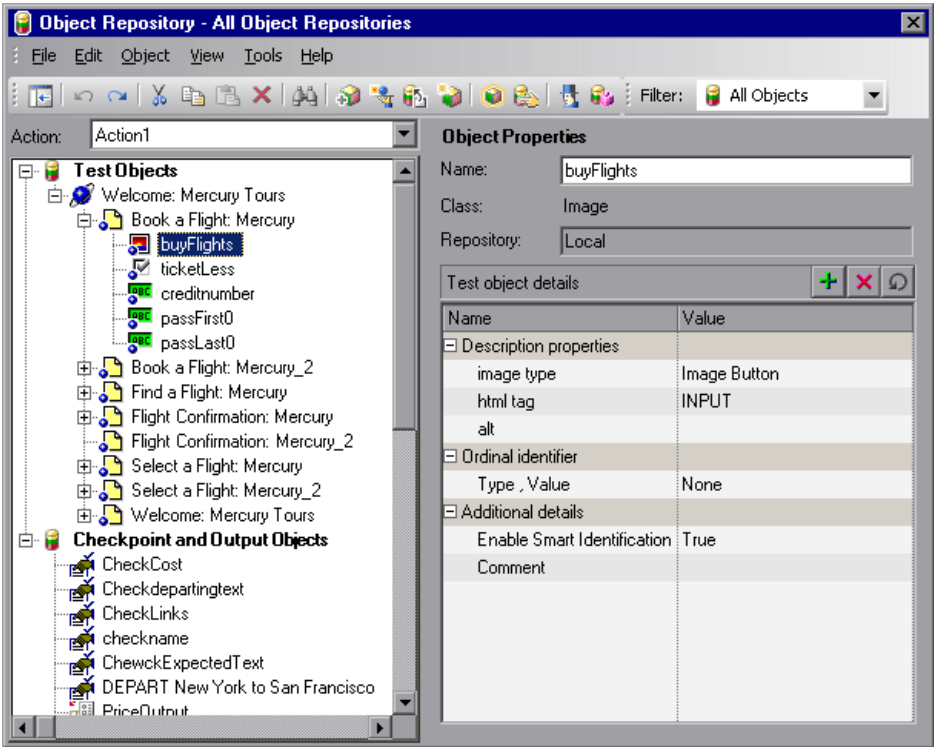
You can use the Object Repository window to view the object description of any object in the repository (in local and shared object repositories), to modify local objects and their properties, and to add test objects to your local object repository. You can also drag and drop test objects from the Object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

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

The Object Repository Window










Description	Enables you to manage identification properties and object repository associations for your action.
How to Access	<ul style="list-style-type: none"> ➤ Click the Object Repository button  ➤ Double-click the repository in the Resources pane, or right-click it and choose Open Repository ➤ Right-click an action in the Test Flow pane and choose Object Repository ➤ Right-click an object in the repository in the Available Keywords pane and choose Open Resource ➤ Choose Resources > Object Repository
Learn More	<p>Conceptual overview: “Understanding the Object Repository Window” on page 182</p> <p>Primary tasks:</p> <ul style="list-style-type: none"> ➤ “Adding Test Objects to a Local or Shared Object Repository” on page 136 ➤ “Copying, Pasting, and Moving Objects in the Object Repository” on page 150 ➤ “Deleting Objects from the Object Repository” on page 153 ➤ “Locating Objects” on page 154 ➤ “Maintaining Identification Properties” on page 162 <p>Additional related topics: “Additional References” on page 189</p>








Below is an image of the Object Repository window:



The Object Repository Window - Edit Toolbar

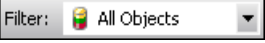
The Object Repository window Edit toolbar contains the following buttons:

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

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

The Object Repository Window - Filter Toolbar

The Filter toolbar contains the following options:

Option	Description
	<p>You can use the Filter toolbar to filter the objects shown in the Object Repository window.</p> <p>You can choose to show objects that meet one of the following criteria:</p> <ul style="list-style-type: none"> ➤ 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 <p>To filter the Object Repository window:</p> <p>In the Filter toolbar list, select one of the following options:</p> <ul style="list-style-type: none"> ➤ All Objects ➤ Local Objects ➤ The name of a specific shared object repository associated with the current action <p>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.</p>

Object Repository Window Options

The Object Repository window contains the following options:

Option	Description
Action	Enables you to select the action whose objects you want to view.
Test Objects tree	<p>Contains all test objects in the selected action (all local test objects and all test objects in any shared object repositories associated with the selected action).</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 446.</p> <p>You can filter the objects shown in the object repository tree. For more information, see “The Object Repository Window - Filter Toolbar” on page 187.</p>
Checkpoint and Output Objects tree	Contains all the checkpoint and output objects in the selected action (all local checkpoint and output objects and all checkpoint and output objects in any shared object repositories associated with the selected action).
Name	The name that QuickTest assigns to the object. You can change the name of a object in the local object repository. For more information, see “Renaming Test Objects” on page 169.
Class	The class of the object.

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

Additional References

Related Tasks	<ul style="list-style-type: none"> ➤ “Mapping Repository Parameter Values” on page 202 ➤ “Exporting Local Objects to a Shared Object Repository” on page 193 ➤ “Copying an Object to the Local Object Repository” on page 195 ➤ “Renaming Test Objects” on page 169 ➤ You can drag and drop test objects from other locations. For more information, see “Understanding the Available Keywords Pane” on page 1165 and “Adding Test Objects to Your Test Using the Object Repository Manager” on page 225. ➤ You can modify the properties of a test object during a test run. For more information, see “Working with Test Objects During a Run Session” on page 206. ➤ You can view and modify object properties from other locations. For more information, see “Maintaining Identification Properties” on page 162.
----------------------	---

Understanding the Object Details Area

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

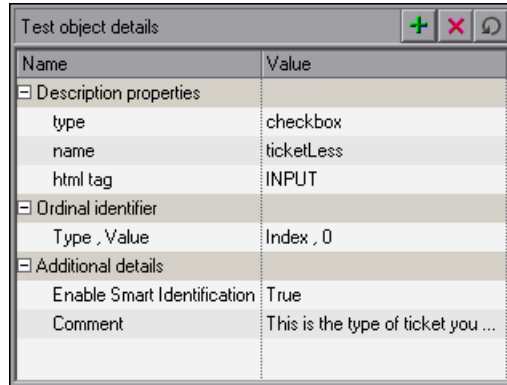
In the Object Repository window, objects in a shared object repository are displayed in the Object Properties pane (including the object details area) in read-only format. To modify objects in a shared object repository, open the shared object repository using the Object Repository Manager. For more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

Tips:

- You can view object properties and property values using the Object Properties dialog box. For more information, see “The Object Properties Dialog Box” on page 197.
- You can use the Object Spy at any time to view native or identification properties and values of the objects in the application you are testing. You open the Object Spy by choosing **Tools > Object Spy** or clicking the **Object Spy** toolbar button. For more information, see “Viewing Object Properties and Operations Using the Object Spy” on page 97.



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



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

Item	Description
Description properties	<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 171.</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 163.</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 177.</p>

Item	Description
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 121.</p> <ul style="list-style-type: none"> ➤ Comment. Enables you to add textual information about the test object.

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

Tips:

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

For more information, see:

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

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

Exporting Local Objects to a Shared Object Repository

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

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

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

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

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

To export local objects to a new shared object repository:



- 1** Open the test that has the local objects you want to export.
- 2** Open the Object Repository window by selecting **Resources > Object Repository** or clicking the **Object Repository** button.
- 3** In the Object Repository window, in the **Action** box, choose the action whose local objects you want to export.
- 4** Select **File > Export Local Objects**, or **File > Export and Replace Local Objects**. The Save Shared Object Repository dialog box opens.
- 5** In the sidebar, select the location in which you want to save the file, for example, File System or Quality Center Test Resources.
- 6** Browse to and select the folder in which you want to save the file.
- 7** In the **File name** box, enter a name for the file. Use a descriptive name that will help you easily identify the file. Do not use any of the following characters in the object repository name:
\\ : * " ? < > | ' \

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

- 8** Click **Save**.

Tip: If you want to save the file as an attachment to a test in the Test Plan module in Quality Center, select **Quality Center Test Plan** in the sidebar, browse to and double-click the test, and then click **Save**.

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

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

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

Copying an Object to the Local Object Repository

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

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

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

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

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

To copy an object to the local object repository:

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

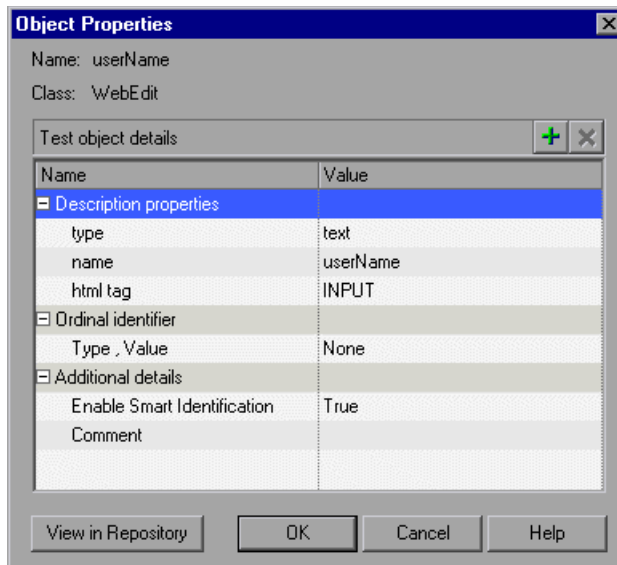
The Object Properties Dialog Box

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

To view object properties and property values in your test:

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

The Object Properties dialog box opens.



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

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

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

Managing Shared Object Repository Associations

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

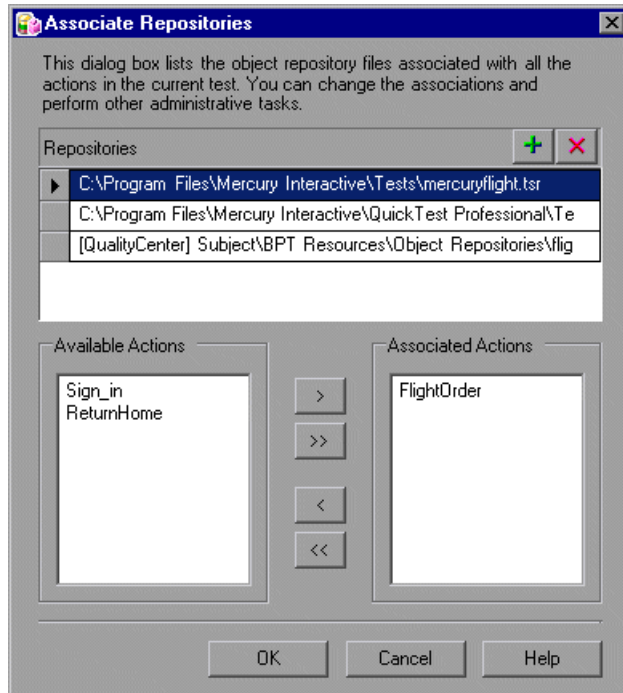
You can also associate, remove, prioritize, and view the properties of shared object repositories in the Resources pane. For more information, see “The Resources Pane” on page 1161.

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

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



The Associate Repositories dialog box opens.



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



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

- 3 To modify the name or path of an associated shared object repository, click a shared object repository in the **Repositories** list and then click the browse button to open a file selection dialog box in which you can select a different shared object repository. Alternatively, you can modify the shared object repository name or path directly in the **Repositories** list. The modified shared object repository remains associated with the same actions as the previous shared object repository.
- 4 To associate an object repository with one or more actions, or remove existing associations, select the object repository in the **Repositories** list, and then double-click the action names or select the action names and click the arrow buttons (> and <) to move them between the **Available Actions** and the **Associated Actions** lists.

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

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



- 5 To remove an object repository from the list and thereby remove all of its associations to any actions in the current test, select the object repository and click the **Remove Repository** button.
- 6 Click **OK**. The changes you made to the object repository associations are applied. You can view the new associations and change the object repository priorities in the Associated Repositories tab of the Action Properties dialog box. For more information, see “Associating Object Repositories with Actions” on page 446.

Mapping Repository Parameter Values

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

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

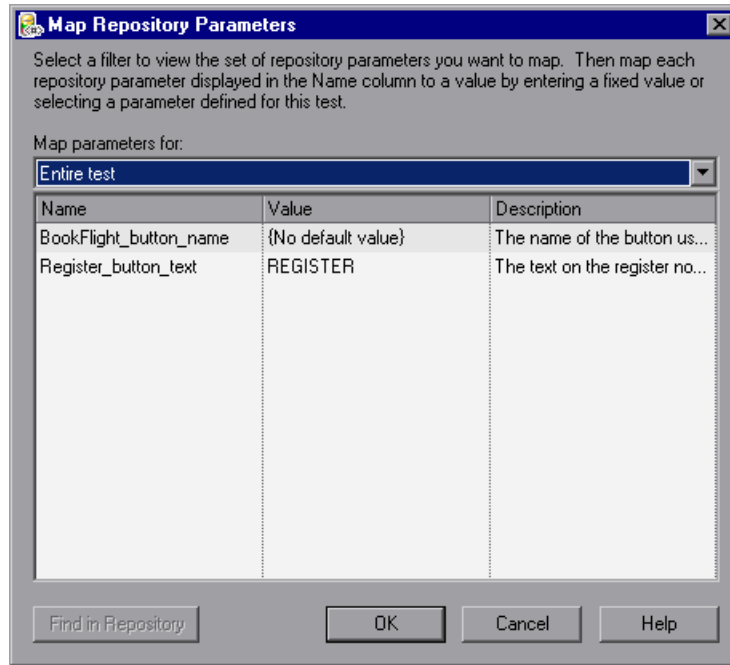
Before you map repository parameters, if you have more than one repository parameter with the same name in different shared object repositories that are associated with the same test, the repository parameter from the shared object repository with the highest priority (as defined in the shared object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

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

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



To map repository parameter values:

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



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

The Map Repository Parameters dialog box contains the following options:

Option name	Description
Map parameters for filter	<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 229.

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

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

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



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



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



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

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

Working with Test Objects During a Run Session

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

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

Creating Test Objects During a Run Session

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

Modifying Identification Properties During a Run Session

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

Use the following syntax for the SetTOProperty method:

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

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

7

Managing Object Repositories

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

This chapter includes:

- About Managing Object Repositories on page 208
- The Object Repository Manager on page 210
- Working with Object Repositories on page 217
- Managing Objects in Shared Object Repositories on page 222
- Working with Repository Parameters on page 228
- Modifying Object Details on page 234
- Locating Test Objects on page 239
- Performing Merge Operations on page 240
- Performing Import and Export Operations on page 241
- Managing Object Repositories Using Automation on page 244

About Managing Object Repositories

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

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

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

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

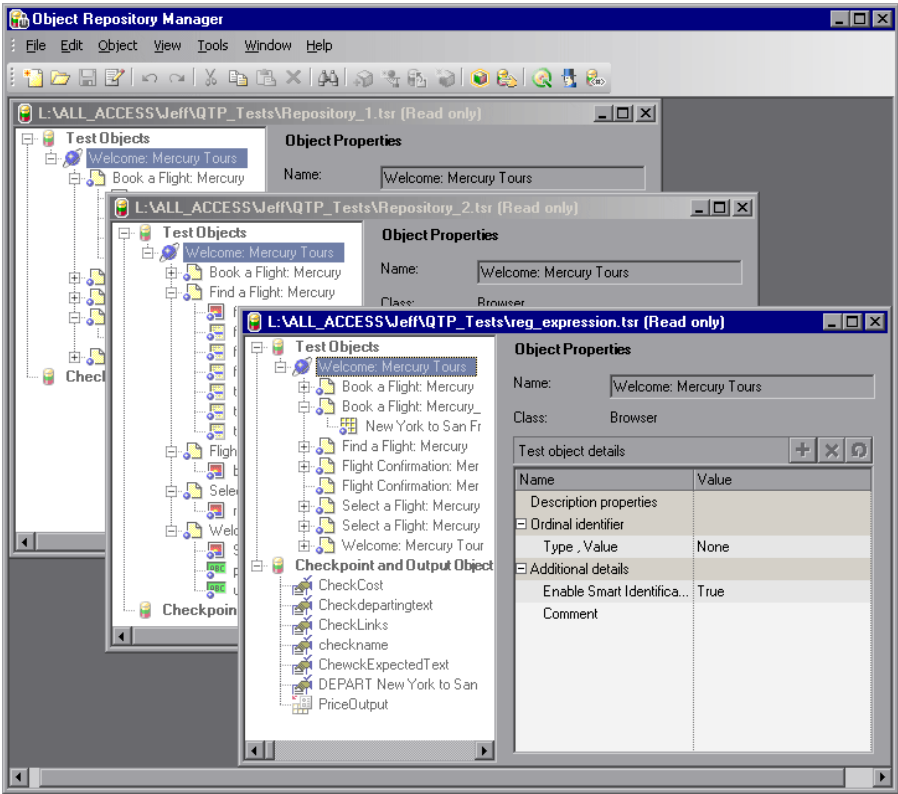
If an object with the same name and description is located in both the local object repository and in a shared object repository that is associated with the same action, the action uses the local object definition. If an object with the same name and description is located in more than one shared object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

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

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

The Object Repository Manager

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



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

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



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









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









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




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

Using the Object Repository Manager Toolbar

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

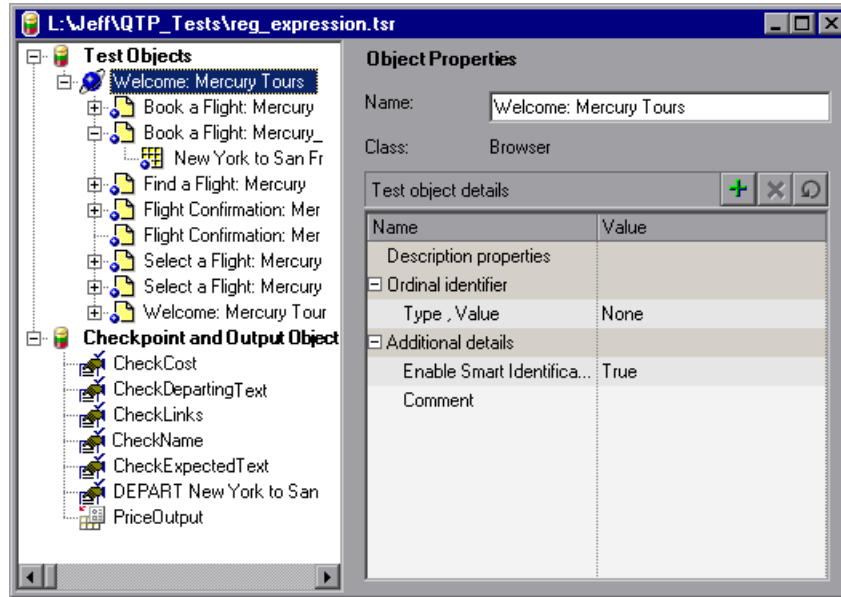
Button	Description
	Enables you to create a new shared object repository. For more information, see “Creating New Object Repositories” on page 217.
	Enables you to open a shared object repository from the file system or from Quality Center. For more information, see “Opening Object Repositories” on page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

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

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

Understanding the Shared Object Repository Windows

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



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

Notes:

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

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

Each object repository window contains the following information:

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

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

Working with Object Repositories

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

Creating New Object Repositories

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

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



To create a new object repository:

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

Opening Object Repositories

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

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



You connect to a Quality Center project either from QuickTest or from the Object Repository Manager by choosing **File > Quality Center Connection** or clicking the **Quality Center Connection** button. For more information on connecting to Quality Center, see “Connecting to and Disconnecting from Quality Center” on page 1418.

Note for users of previous QuickTest versions:

When you open an object repository that is stored in the file system and was created using a version of QuickTest earlier than version 9.0, QuickTest converts it to the current format when you make it editable.

If the object repository contains test objects from add-ins, the relevant add-in must be installed to convert the object repository to the current format. Otherwise, you can open it only in read-only format.

If you do not want to convert the object repository, you can view it in read-only format. After the file is converted and you save it, you cannot use it with earlier versions of QuickTest.

To open an object repository:



- 1** In the Object Repository Manager, select **File > Open** or click the **Open** button. The Open Shared Object Repository dialog box opens.
- 2** In the sidebar, select the location of the object repository file, for example, File System or Quality Center Test Resources. Browse to and select the object repository file you want to open, and click **Open**. The object repository opens.

By default, the object repository opens in read-only mode. You can open it in editable format by clearing the **Open in read-only mode** check box in the Open Shared Object Repository dialog box. You can also enable editing for an object repository as described in “Editing Object Repositories” on page 224.

If the object repository is editable, you can add objects to it, modify it, and save it. For more information, see “Managing Objects in Shared Object Repositories” on page 222 and “Saving Object Repositories” on page 219.

Tip: You can also open an object repository from the **Recent Files** list in the **File** menu.

Saving Object Repositories

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

After you finish creating or modifying an object repository, you should save it. When you modify an object repository, an asterisk (*) is displayed in the title bar until the object repository is saved.



You can save an object repository to the file system or to a Quality Center project (if you are connected to a Quality Center project). You connect to a Quality Center project either from QuickTest or from the Object Repository Manager by choosing **File > Quality Center Connection** or clicking the **Quality Center Connection** button. For more information on connecting to Quality Center, see “Connecting to and Disconnecting from Quality Center” on page 1418.

All changes you make to an object repository are automatically updated in all tests open on the same computer that use the object repository as soon as you make the change—even if you have not yet saved the object repository with your changes. If you close the object repository without saving your changes, the changes are rolled back in any open tests that were open at the time.

When you open a test on the same computer on which you modified the object repository, the test is automatically updated with all saved changes made in the associated object repository. To see saved changes in a test or repository open on a different computer, you must open the test or object repository file or lock it for editing on your computer to load the changes.

To save an object repository:

- 1 Make sure that the object repository you want to save is the active window.
- 2 Select **File > Save** or click the **Save** button. If the file has already been saved, the changes you made are saved. If the file has not yet been saved, the Save Shared Object Repository dialog box opens.
- 3 In the sidebar, select the location in which you want to save the file, for example, File System or Quality Center Test Resources.
- 4 Browse to and select the folder in which you want to save the file.



- 5 In the **File name** box, enter a name for the file. Use a descriptive name that will help you easily identify the file. Do not use any of the following characters in the object repository name:

\ / : * " ? < > | '

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

- 6 Click **Save**.

Tip: If you want to save the file as an attachment to a test in the Test Plan module in Quality Center, select **Quality Center Test Plan** in the sidebar, browse to and double-click the test, and then click **Save**.

Note: When you specify a path to a resource in the file system or in Quality Center 9.x, QuickTest checks if the path, or a part of the path, exists in the Folders pane of the Options dialog box (**Tools > Options > Folders** node). If the path exists, you are prompted to define the path using only the relative part of the path you entered. If the path does not exist, you are prompted to add the resource's location path to the Folders pane and define the path relatively. For more information, see “Using Relative Paths in QuickTest” on page 316.

If you are working with the Resources and Dependencies model with Quality Center 10.00, you should specify an absolute Quality Center path. For more information, see “Considerations for Working with Relative Paths in Quality Center” on page 1450.

QuickTest saves the object repository with a **.tsr** extension in the specified location and displays the object repository name and path in the title bar of the repository window.

Closing Object Repositories

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

After you finish modifying or using an object repository, you should close it. While an object repository is being edited, it is locked so that it cannot be modified by others. When you close the object repository, it is automatically unlocked. You can also choose to close all open object repositories.

Note: If you close QuickTest, the Object Repository Manager also closes. If you have made changes that are not yet saved, you are prompted to do so before the Object Repository Manager closes.

To close an object repository:

- 1 Make sure that the object repository you want to close is the active window.
- 2 Select **File > Close** or click the **Close** button in the object repository window's title bar. The object repository is closed and is automatically unlocked. If you have made changes that are not yet saved, you are prompted to do so before the file closes.

To close all open object repositories:

Select **File > Close All Windows**, or **Window > Close All Windows**. All open object repositories are closed and are automatically unlocked. If you have made changes that are not yet saved, you are prompted to do so before the files close.

Managing Objects in Shared Object Repositories

You can modify your shared object repositories in a variety of ways to either prepare them for initial use or update them throughout the testing process. You can add and modify objects and object properties in a shared object repository, copy or move objects from one object repository to another, drag objects to a different location in the hierarchy, delete objects, and rename objects. You can also drag and drop test objects from the Object Repository manager to your test. When you modify a shared object repository, an asterisk (*) is displayed in the title bar until the object repository is saved.

The following are tips and guidelines for working with the Object Repository Manager:



- You can use the **Edit > Undo** and **Edit > Redo** options or **Undo** and **Redo** buttons to cancel or repeat your changes as necessary. The **Undo** and **Redo** options are related to the active document. When you save an object repository, you cannot undo and redo operations that were performed on that file before the save operation.
- If you opened the object repository in read-only mode, you must enable editing for the object repository before you can modify it. This locks the object repository and prevents it from being modified simultaneously by multiple users.
- All changes you make to an object repository are automatically updated in all tests open on the same computer that use the object repository as soon as you make the change—even if you have not yet saved the object repository with your changes.

If you close the object repository without saving your changes, the changes are rolled back in any open tests that were open at the time.

- When you open a test on the same computer on which you modified the object repository, the test is automatically updated with all saved changes made in the associated object repository. To see saved changes in a test or repository open on a different computer, you must open the test or object repository file or lock it for editing on your computer to load the changes.

- You can also modify a shared object repository by merging it with another shared object repository. When you merge two shared object repositories, a new shared object repository is created, containing the content of both object repositories. If you merge a shared object repository with a local object repository, the shared object repository is updated with the content of the local object repository. For more information, see Chapter 8, “Merging Shared Object Repositories.”
- After making sure that your shared object repository is editable, and that it is the active window, you can modify it in the same way as you modify a local object repository. In addition to adding objects to a shared object repository in the same manner as to a local repository, you can also add objects to a shared object repository using the **Navigate and Learn** option.

For more information, see:

- “Editing Object Repositories” on page 224
- “Adding Test Objects to Your Test Using the Object Repository Manager” on page 225
- “Adding Test Objects to a Local or Shared Object Repository” on page 136
- “Adding Test Objects Using the Navigate and Learn Option” on page 225
- “Copying, Pasting, and Moving Objects in the Object Repository” on page 150
- “Deleting Objects from the Object Repository” on page 153

Editing Object Repositories

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


When you open an object repository, it is opened in read-only mode by default. You can open it in editable format by clearing the **Open in read-only mode** check box in the Open Shared Object Repository dialog box when you open it.

If you opened the object repository in read-only mode, you must enable editing for the object repository before you can modify it. You do not need to enable editing for an object repository if you only want to view it or copy objects from it to another object repository.

When you enable editing for an object repository, the object repository is locked so that it cannot be modified by other users. To enable other users to modify the object repository, you must first unlock it (by disabling edit mode, or by closing it). If an object repository is already locked by another user, if it is saved in read-only format, or if you do not have the permissions required to open it, you cannot enable editing for it.

Note for users of previous QuickTest versions: If you want to edit an object repository stored in the file system, and the object repository was created using a version of QuickTest earlier than version 9.0, QuickTest must convert it to the current format before you can edit it. If you do not want to convert it, you can view it in read-only format. After the file is converted and saved, you cannot use it with earlier versions of QuickTest.

To enable editing for an object repository:

- 1 Make sure that the object repository you want to edit is the active window.
- 2  Select **File > Enable Editing** or click the **Enable Editing** button. The object repository becomes editable.

Adding Test Objects to Your Test Using the Object Repository Manager

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

You can drag and drop test objects from the Object Repository Manager to your test. When you drag and drop a test object to your test, QuickTest inserts a step with the default operation for that test object in your test. You cannot drag and drop checkpoint or output objects from the Object Repository Manager.

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

You can also drag and drop test objects from other locations. For more information, see:

- “Understanding the Available Keywords Pane” on page 1165
- “The Object Repository Window” on page 183

Adding Test Objects Using the Navigate and Learn Option

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

The **Navigate and Learn** option enables you to add multiple test objects to a shared object repository while navigating through your application.

Each time you select a window to learn, the selected window and its descendant objects are added to the active shared object repository according to a predefined object filter. You can change the object filter definitions at any time to meet your requirements. The object filter is used for both the **Navigate and Learn** option and the **Add Objects** option. The settings you define are used in both places when QuickTest learns objects. For more information on modifying the filter definitions, see “Understanding the Define Object Filter Dialog Box” on page 144.

Note: The Navigate and Learn option is not supported for environments with mixed hierarchies (object hierarchies that include objects from different environments), for example, `Browser("Homepage").Page("Welcome").AcxButton("Save")` or `Dialog("Edit").AcxEdit("MyEdit")`. To add objects within mixed hierarchies, use other options, as described in “Adding Test Objects to a Local or Shared Object Repository” on page 136.

You can use the following keyboard shortcuts when learning objects using the **Navigate and Learn** option:

- **Learn Focused Window.** ENTER
- **Define Object Filter.** CTRL+F
- **Help.** F1
- **Return to Object Repository Manager.** ESC

Note: Minimized windows are not learned when using the **Navigate and Learn** option.

To add test objects using the Navigate and Learn option:

- 1** In the Object Repository Manager, make sure that the object repository to which you want to add objects is the active window and that it is editable.
- 2** Select **Object > Navigate and Learn** or press F6. The **Navigate and Learn** toolbar opens.





Note: If this is the first time you are adding objects to the object repository, you may want to change the filter definitions before you continue. You can view the current filter definitions in the **Define Object Filter** button tooltip (displayed in parentheses after the button name). You can change the filter definitions at any time by clicking the **Define Object Filter** button or pressing CTRL+F. For more information, see “Understanding the Define Object Filter Dialog Box” on page 144.

- 3** Click the parent object (for example, Browser, Dialog, Window) you want to add to the object repository to focus it. The **Learn** button in the toolbar is enabled.
- 4** Click the **Learn** button or focus the **Navigate and Learn** toolbar and press ENTER. A flashing highlight surrounds the focused window and the object and its descendants are added to the object repository according to the defined filter.
- 5** Navigate in your application to the next window you want to add and then repeat step 4.
- 6** When you finish adding the required objects to the object repository, click the **Close** button in the **Navigate and Learn** toolbar or press Esc. The **Navigate and Learn** toolbar closes and the Object Repository Manager is redisplayed, showing the objects you just added to the shared object repository.

Working with Repository Parameters

Repository parameters enable you to specify that certain property values should be parameterized, but leave the actual parameterization to be defined in each test that is associated with the object repository that contains the parameterized identification property values.

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

For example, you may have a button whose text property value changes in a localized application depending on the language of the user interface. You can parameterize the name property value using a repository parameter, and then in each test that uses the object repository you can specify the location from which the property value should be taken. For example, in one test that uses this object repository you can specify that the property value comes from an environment variable, in another test it can come from the Data Table, and in a third test you can specify it as a constant value.

You define all the repository parameters for a specific object repository using the Manage Repository Parameters dialog box. You define each repository parameter together with an optional default value and meaningful description. For more information, see “Managing Repository Parameters” on page 229.

When you open a test that uses an object repository with a repository parameter that has no default value, an indication that there is a repository parameter that needs mapping is displayed in the Missing Resources pane. You can then map the repository parameter as needed in the test. You can also map repository parameters that have default values, and change mappings for repository parameters that are already mapped. For more information on mapping repository parameters, see “Handling Unmapped Shared Object Repository Parameter Values” on page 1194.

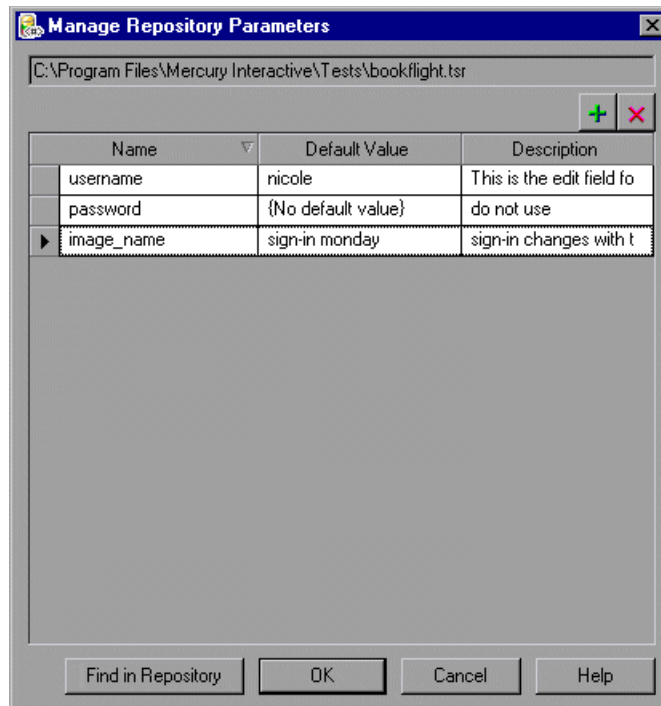
Managing Repository Parameters

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



The Manage Repository Parameters dialog box enables you to add, edit, and delete repository parameters for a single shared object repository.

To manage repository parameters:

- 1 Make sure that the object repository whose parameters you want to manage is the active window.
- 2 If the object repository is in read-only format, select **File > Enable Editing** or click the **Enable Editing** button. The object repository becomes editable.
- 3 Select **Tools > Manage Repository Parameters** or click the **Manage Repository Parameters** button. The Manage Repository Parameters dialog box opens.



The Manage Repository Parameters dialog box contains the following information and options:

Option	Description
Repository name	Displays the name and path of the object repository whose repository parameters you are managing.
	Enables you to add a new repository parameter. For more information, see “Adding Repository Parameters” on page 230.
	Enables you to delete the currently selected repository parameters. For more information, see “Deleting Repository Parameters” on page 233.
Parameter list (Name , Default Value , and Description)	Displays the list of repository parameters currently defined in this object repository. You can modify a parameter’s default value and description directly in the parameter list. For more information, see “Modifying Repository Parameters” on page 232.
Find in Repository button	Searches for and highlights the first test object in the object repository tree that uses the selected repository parameter. You can click this button again to find the next occurrence of the selected parameter, and so forth.

Adding Repository Parameters

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

The Add Repository Parameter dialog box enables you to define a new repository parameter. You can also specify a default value for the parameter, and a meaningful description to help identify it when it is used in a test step.

For more information on repository parameters, see “Working with Repository Parameters” on page 228.

To add a repository parameter:

- 1 In the Manage Repository Parameters dialog box, click the **Add Repository Parameter** button. The Add Repository Parameter dialog box opens.

- 2 In the **Name** box, specify a meaningful name for the parameter. Parameter names must start with an English (Roman) letter and can contain only English (Roman) letters, numbers, and underscores.
- 3 In the **Default value** box, you can specify a default value to be used for the repository parameter. This value is used if you do not map the repository parameter to a value or parameter type in a test that uses this object repository. If you do not specify a default value, the repository parameter will appear as unmapped in any tests that use this shared object repository.



Tip: If you specify a default value, you can later remove it by clicking in the **Default Value** cell of the relevant parameter in the Manage Repository Parameters dialog box and then clicking the **Clear Default Value** button. The text **{No Default Value}** is displayed in the cell.


- 4 In the **Description** box, you can enter a description of the repository parameter. The description will help you identify the parameter when mapping repository parameters within a test.
- 5 Click **OK** to add the parameter to the list of parameters in the Manage Repository Parameters dialog box.

Modifying Repository Parameters

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

You can modify the default value of a repository parameter or modify a repository parameter description directly in the Manage Repository Parameters dialog box. However, you cannot modify a repository parameter name.

To modify a repository parameter:

- 1 In the Manage Repository Parameters dialog box, select the required parameter.
-  2 To modify the default value, click in the **Default Value** cell of the required parameter. You can either modify the default value by entering a new value, or you can remove the default value by clicking the **Clear Default Value** button. If you remove the default value, the text **{No Default Value}** is displayed in the cell. If you do not specify a default value, the repository parameter will appear as unmapped in any tests that use this shared object repository.



Note: If you delete the text manually, it does not remove the default value. It creates a default value of an empty string. You must click the **Clear Default Value** button if you want to remove the default value.

- 3 To modify the parameter description, click in the **Description** cell of the required parameter and enter the required description.

Deleting Repository Parameters

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

You can delete a repository parameter definition if it is no longer needed. When you delete a repository parameter that is used in a test object definition, the identification property value remains mapped to the parameter, even though the parameter no longer exists. Therefore, before deleting a repository parameter, you should make sure that it is not used in any test object descriptions, otherwise tests that have steps using these test objects will fail when you run them.

Tip: You can use the **Find in Repository** button in the Manage Repository Parameters dialog box to see where a repository parameter is being used.

To delete a repository parameter:

- 1 In the Manage Repository Parameters dialog box, select the repository parameters that you want to delete by clicking in the selection area to the left of the parameter name.
- 2 Click the **Delete Repository Parameter** button. The selected repository parameter is deleted.



Modifying Object Details

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

After making sure that your shared object repository is editable, and that it is the active window, you modify object details for objects in a shared object repository in the same way as you modify them for local objects. For more information, see:

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



Note: You can use the **Edit > Undo** and **Edit > Redo** options or **Undo** and **Redo** buttons to cancel or repeat your changes as necessary. The **Undo** and **Redo** options are related to the active document. When you save a repository, you cannot undo and redo operations that were performed on that file before the save operation.

You use the Object Repository Manager to specify property values for object descriptions in a shared object repository. The options available when specifying property values for objects in shared object repositories are different from those available when specifying properties for objects in local repositories. For more information on specifying property values for objects in shared object repositories, see “Specifying a Property Value” on page 235.

Specifying a Property Value

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

You can specify or modify values for properties in the test object description. You can specify a value using a constant value (either a simple value or a constant value that includes regular expressions) or you can parameterize it using a repository parameter. For more information on repository parameters, see “Working with Repository Parameters” on page 228.

You can also specify or modify values for properties of a checkpoint or output object.

Specifying and Modifying Values for Properties of a Test Object

You specify or modify the values for properties of a test object in the **Test object details** area.

To specify a property value of a test object:

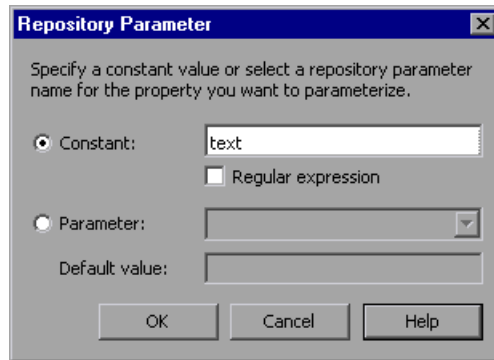
- 1** Select the test object whose property value you want to specify.
- 2** In the **Test object details** area, click in the **Value** cell for the required property.

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

- If you want to specify a simple constant value, enter it in the **Value** cell. The remaining steps in this procedure are not necessary if you specify a constant value in the **Value** cell. You can also specify a constant value using a regular expression in the Repository Parameter dialog box, as described below.



- If you want to parameterize the value using a repository parameter, click the parameterization button in the **Value** cell. The Repository Parameter dialog box opens.

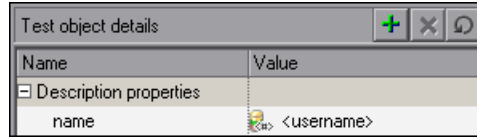


4 Select one of the following options to specify a value for the property:

- Select the **Constant** radio button and specify a constant value. You can also enter a constant value directly in the **Value** cell of the **Test object details** area. If you used a regular expression in the constant value, select the **Regular expression** check box.
- Select the **Parameter** radio button and select a repository parameter from the list of defined parameters. If a default value is defined for the parameter, it is also shown.

Note: You define repository parameters using the Manage Repository Parameters dialog box. For more information, see “Managing Repository Parameters” on page 229.

- 5 Click **OK** to close the Repository Parameter dialog box. If you parameterized the value, the parameter name is shown with an icon in the **Value** column of the **Test object details** area, as shown below. Otherwise, the constant value you specified is shown in the **Value** column.



Specifying and Modifying Values for Properties of a Checkpoint Object

You specify or modify the values for properties of a checkpoint object in the **Object Properties** pane.

To specify or modify the values for properties of a checkpoint object:

- 1 Select the checkpoint object whose property values you want to specify or modify from the **Checkpoint and Output Objects** tree.
- 2 Specify or modify the values for properties of a checkpoint object the same way as you do in the relevant checkpoint properties dialog box.

For more information on specifying and modifying values for properties of a checkpoint object, see:

- “Understanding the Checkpoint Properties Dialog Box” on page 508
- “Understanding the Image Checkpoint Properties Dialog Box” on page 512
- “The Bitmap Checkpoint Properties Dialog Box” on page 522
- “Understanding the Table Checkpoint Properties Dialog Box” on page 535
- “The Text / Text Area Checkpoint Properties Dialog Box” on page 557
- “Understanding the Database Checkpoint Properties Dialog Box” on page 581

- “Understanding the XML Checkpoint Properties Dialog Box” on page 607
- The Web section of the *HP QuickTest Professional Add-ins Guide* (for Page and Accessibility checkpoints)

Specifying and Modifying Values for Properties of an Output Object

You specify or modify the values for properties of an output object in the **Object Properties** pane.

To specify or modify the values for properties of an output object:

- 1** Select the output object whose property values you want to specify or modify from the **Checkpoint and Output Objects** tree.
- 2** Specify or modify the values for properties of an output object the same way as you do in the relevant output value properties dialog box.

For more information on specifying and modifying values for properties of an output object, see:

- “Defining Standard Output Values” on page 679
- “Defining Text and Text Area Output Values” on page 692
- “Outputting Table Content” on page 703
- “Defining Database Output Values” on page 715
- “Understanding the XML Output Properties Dialog Box” on page 727

Locating Test Objects

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

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

After making sure that your shared object repository is the active window, you locate an object in a shared object repository in the same way as you locate it in a local object repository. If you want to replace property values, you must also make sure that the object repository is editable.

For more information, see:

- “Finding Objects in an Object Repository” on page 154
- “Highlighting an Object in Your Application” on page 157
- “Locating a Test Object in the Object Repository” on page 159

Performing Merge Operations

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

The Object Repository Merge Tool enables you to merge test objects from the local object repository of one or more actions to a shared object repository using the **Update from Local Repository** option in the Object Repository Manager (**Tools > Update from Local Repository**). For example, you may have learned test objects locally in a specific action in your test and want to add them to the shared object repository so they are available to all actions in different tests that use that object repository. You can also use the Object Repository Merge Tool to merge two shared object repositories into a single shared object repository.

You open the Object Repository Merge Tool by choosing **Tools > Object Repository Merge Tool** in the Object Repository Manager. For more information on performing merge operations and updating object repositories with local objects, see Chapter 8, “Merging Shared Object Repositories.”

Notes:

- While the Object Repository Merge Tool is open, you cannot work with the Object Repository Manager.
 - The Object Repository Merge Tool does not merge checkpoint and output objects.
-

Performing Import and Export Operations

You can import and export object repositories from and to XML files. XML provides a structured, accessible format that enables you to make changes to object repositories using the XML editor of your choice and then import them back into QuickTest. You can view the required format for the object repository in the *HP QuickTest Professional Object Repository Schema Help* (**Help > QuickTest Professional Help > HP QuickTest Professional Advanced References > HP QuickTest Professional Object Repository Schema**), or by exporting a saved object repository.

You can import and export files either from and to the file system or a Quality Center project (if QuickTest is connected to Quality Center).



You connect to a Quality Center project either from QuickTest or from the Object Repository Manager by choosing **File > Quality Center Connection** or clicking the **Quality Center Connection** button. For more information on connecting to Quality Center, see “Connecting to and Disconnecting from Quality Center” on page 1418.

For more information, see:

- “Importing from XML” on page 242
- “Exporting to XML” on page 243
- “Understanding the XML File Structure” on page 244

Importing from XML

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

You can import an XML file (created using the required format) as an object repository. For information on the XML format, see “Understanding the XML File Structure” on page 244. The XML file can either be an object repository that you exported to XML format using the Object Repository Manager, or an XML file created using a tool such as QuickTest Siebel Test Express or a custom built utility. You must adhere to the XML structure and format.

Tip: To view the required XML structure and format, see the *HP QuickTest Professional Object Repository Schema Help* (**Help > QuickTest Professional Help > HP QuickTest Professional Advanced References > HP QuickTest Professional Object Repository Schema**). You can also export an existing shared object repository to XML and then use the XML file as a guide. For more information, see “Exporting to XML” on page 243.

To import from XML:

- 1 Select **File > Import from XML**. The Open XML File dialog box opens.

Note: Checkpoint and output objects are not included when importing the contents of an object repository from an XML file.

- 2 In the sidebar, select the location of the file, for example, **File System** or **Quality Center Test Resources**. Browse to and select the XML file you want to import, and click **Open**.

The XML file is imported and a summary message box opens showing information regarding the number of objects, parameters, and metadata that were successfully imported from the specified file.

- 3 Click **OK** to close the message box. The imported XML file is opened as a new object repository. You can now modify it as required and save it as an object repository.

Exporting to XML

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

You can export the test objects in an object repository to an XML file. This enables you to edit it using any XML editor, and also enables you to save it in an accessible, versatile format.

To export to XML:

- 1 Make sure that the object repository whose test objects you want to export is the active window.
- 2 Make sure that the object repository is saved.
- 3 Select **File > Export Test Objects to XML**. The Save XML File dialog box opens.

Note: Checkpoint and output objects are not included when exporting the contents of an object repository to an XML file.

- 4 In the sidebar, select the location in which you want to save the file, for example, File System or Quality Center Test Resources.
- 5 Browse to and select the folder in which you want to save the file.
- 6 In the **File name** box, enter a name for the file and click **Save**.

Tip: If you want to save the file as an attachment to a test in the Test Plan module in Quality Center, select **Quality Center Test Plan** in the sidebar, browse to and double-click the test, and then click **Save**.

The test objects of the object repository are exported to the specified XML file, and a summary message box opens showing information regarding the number of objects, parameters, and metadata that were successfully exported to the specified file.

- 7 Click **OK** to close the message box. You can now open the XML file and view or modify it with any XML editor.

Understanding the XML File Structure

QuickTest uses a defined XML schema for object repositories. You must follow this schema when creating or modifying object repository files in XML format. The schema of this file is documented in the *HP QuickTest Professional Object Repository Schema Help* (**Help > QuickTest Professional Help > HP QuickTest Professional Advanced References > HP QuickTest Professional Object Repository Schema**).

Managing Object Repositories Using Automation

QuickTest provides an Object Repository automation object model that enables you to manage QuickTest shared object repositories and their contents from outside of QuickTest. The automation object model enables you to use a scripting tool to access QuickTest shared object repositories via automation.

Just as you use the QuickTest Professional automation object model to automate your QuickTest operations, you can use the objects and methods of the Object Repository automation object model to write scripts that manage shared object repositories, instead of performing these operations manually using the Object Repository Manager. For example, you can add, remove, and rename test objects; import from and export to XML; retrieve and copy test objects; and so forth.

After you have retrieved a test object, you can manipulate it using the methods and properties available for that test object class. For example, you can use the `GetTOPProperty` and `SetTOPProperty` methods to retrieve and modify its properties. For more information on available test object methods and properties, see the *HP QuickTest Professional Object Model Reference*.

Automation programs are especially useful for performing the same tasks multiple times or on multiple object repositories. You can write your automation scripts in any language and development environment that supports automation. For example, you can use VBScript, JavaScript, Visual Basic, Visual C++, or Visual Studio .NET. For general information on controlling QuickTest using automation, see “Automating QuickTest Operations” on page 1403.

Using the QuickTest Professional Object Repository Automation Reference

The QuickTest Professional Object Repository Automation Reference is a Help file that provides detailed descriptions, syntax information, and examples for the objects and methods in the QuickTest object repository automation object model.

The Help topic for each automation object includes a list and description of the methods associated with that object. Method Help topics include detailed description, syntax, return value type, and argument value information.

You can open the *QuickTest Professional Object Repository Automation Reference* from the main QuickTest Help (**Help > QuickTest Professional Help > HP QuickTest Professional Advanced References > HP QuickTest Professional Object Repository Automation**).

Note: The syntax and examples in the Help file are written in VBScript-style. If you are writing your automation program in another language, the syntax for some methods may differ slightly from what you find in the corresponding Help topic. For information on syntax for the language you are using, see the documentation included with your development environment or to general documentation for the programming language.

8

Merging Shared Object Repositories

QuickTest Professional enables you to merge two shared object repositories into a single shared object repository using the Object Repository Merge Tool. You can also use this tool to merge objects from the local object repository of one or more actions into a shared object repository.

This chapter includes:

- About Merging Shared Object Repositories on page 248
- Understanding the Object Repository Merge Tool on page 250
- Using Object Repository Merge Tool Commands on page 257
- Defining Default Settings on page 262
- Merging Two Object Repositories on page 267
- Updating a Shared Object Repository from Local Object Repositories on page 269
- Viewing Merge Statistics on page 276
- Understanding Object Conflicts on page 277
- Resolving Object Conflicts on page 280
- Filtering the Target Repository Pane on page 282
- Finding Specific Objects on page 284
- Saving the Target Object Repository on page 285

About Merging Shared Object Repositories

When you have multiple shared object repositories that contain test objects from the same area of your application, it may be useful to combine those test objects into a single object repository for easier maintenance. You could do this by moving or copying objects in the Object Repository Manager. However, if you have test objects in different object repositories that represent the same object in your application, and the descriptions for these objects in the different object repositories are not identical, it may be difficult to recognize and handle these conflicts.

The Object Repository Merge Tool helps you to solve the above problem by merging two selected object repositories for you and providing options for addressing test objects with conflicting descriptions. Using this tool, you merge two shared object repositories (called the **primary** object repository and the **secondary** object repository), into a new third object repository, called the **target** object repository. Objects in the primary and secondary object repositories are automatically compared and then added to the target object repository according to preconfigurable rules that define the defaults for how conflicts between objects are resolved.

After the merge process, the Object Repository Merge Tool provides a graphic presentation of the original objects in the primary and secondary object repositories, which remain unchanged, as well as the objects in the merged target object repository. Objects that had conflicts are highlighted. The conflict of each object that you select in the target object repository is described in detail. The Object Repository Merge Tool provides specific options that enable you to keep the default resolution for each conflict, or modify conflict resolutions individually, according to your requirements.

The Object Repository Merge Tool also enables you to merge objects from the local object repository of one or more actions into a shared object repository. For example, if QuickTest learned objects locally in a specific action in your test, you may want to add the objects to the shared object repository, so that they are available to all actions in different tests that use that object repository.

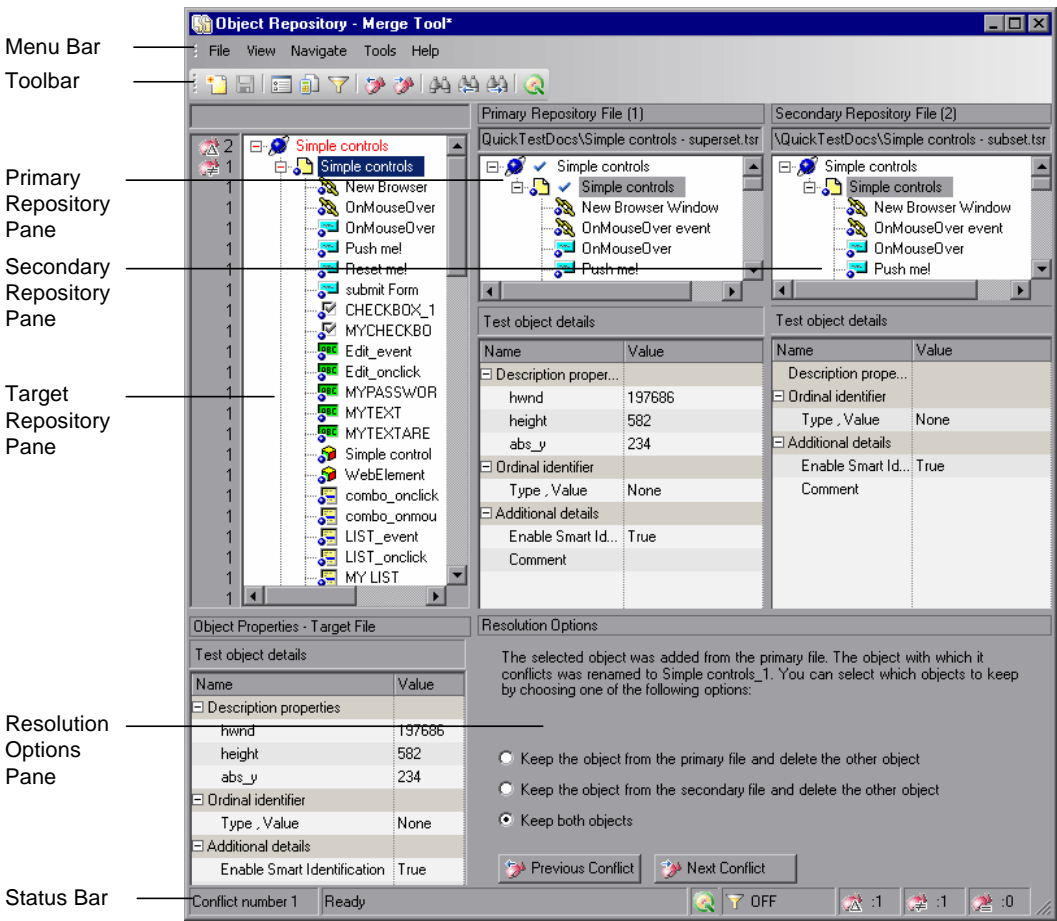
Notes:

- The Object Repository Merge Tool does not merge checkpoint or output objects from the primary and secondary object repositories into the target shared object repository. You can copy or manually move these objects to your target object repository after you complete the merge process, using the Object Repository Manager.
 - When the Object Repository Merge Tool is open, you cannot work with the Object Repository Manager or Object Repository Comparison Tool. For more information on the Object Repository Manager, see Chapter 7, “Managing Object Repositories.”
-

Understanding the Object Repository Merge Tool

You open the Object Repository Merge Tool by choosing **Tools > Object Repository Merge Tool** in the Object Repository Manager.

An example of the Object Repository - Merge Tool window is shown below:



Note: For information about changing the view presented by the Object Repository Merge Tool, see “Changing the View” on page 252.

The Object Repository - Merge Tool window contains the following key elements:

- **Menu bar.** Displays menus of Object Repository Merge Tool commands. These commands are described in various places throughout this chapter. Shortcut keys for menu commands are described in “File Menu Commands” on page 258.
- **Toolbar.** Contains buttons of commonly used menu commands to assist you in merging, managing, and saving object repositories. Toolbar buttons are described in “Using Toolbar Commands” on page 257.
- **Target Repository Pane.** Displays the objects that were merged from the primary and secondary object repositories. You can also choose to show or hide the Target Repository Object Properties pane, which displays the properties of any object that is selected in the Target Repository pane. For more information, see “Target Repository Pane” on page 252.
- **Primary Repository Pane.** Displays the objects in the primary object repository. For more information, see “Primary and Secondary Repository Panes” on page 254.
- **Secondary Repository Pane.** Displays the objects in the secondary object repository. For more information, see “Primary and Secondary Repository Panes” on page 254.
- **Resolution Options Pane.** Provides source, conflict, and resolution details about the objects in the target object repository pane, and enables you to modify how a selected conflict is resolved. For more information, see “Resolution Options Pane” on page 254.
- **Status Bar.** Provides source, conflict, and resolution details about the object selected in the target object repository pane, the filter status, and an icon legend. For more information, see “Status Bar” on page 255.

Changing the View

You can change the view presented by the Object Repository Merge Tool according to your working preferences.

- Drag the edges of the panes to resize them in the Object Repository Merge Tool window.
- Select **Primary Repository**, **Secondary Repository**, **Target Repository Object Properties**, or **Resolution Options** from the **View** menu to hide or show these panes in the Object Repository Merge Tool.
- Select **View > Set as Default Layout** to set your current view as the default view, which displays each time you open the Object Repository Merge Tool. You can select **View > Restore Default Layout** to restore the view to the default settings after you make changes.

Target Repository Pane

The target object repository pane displays a hierarchy of the objects, as well as their respective properties and values, that were merged from the primary and secondary object repositories. In the column to the left of the object hierarchy, the pane displays the source file of each object (**1** is displayed for the primary file and **2** for the secondary file), and an icon representing the type of conflict, if any.

When you save the target object repository, the file path is displayed above the object hierarchy.

Note: To make it easier to see the status of an object at a glance, the text colors of the object names in the target object repository can be set according to their source and whether they caused a conflict. For more information, see “Specifying Color Settings” on page 265.

The target object repository pane provides the following functionality:

- When you select an object in the target object repository, the corresponding object in the primary and/or secondary source file hierarchy is located and indicated by a check mark.
- When you select an object in the target object repository, its properties and values are displayed in the **Object Properties - Target File** area at the bottom of the target object repository pane (**View > Target Repository Object Properties**).
- If the merge results in a conflict, an icon is displayed to the left of the conflicting object in the target object repository. You can see a tooltip description of the conflict type by positioning your pointer over the icon.
- When you right-click an object, a context-sensitive menu opens. You can expand an option or collapse the entire hierarchy in the target object repository, or, when applicable, you can change the conflict resolution method and result.
- You can expand or collapse the hierarchy of the node by double-clicking a node. You can also expand or collapse the entire hierarchy in the target object repository by choosing **Collapse All** or **Expand All** from the **View** menu.
- You can jump directly to the next or previous conflict in the target object repository hierarchy by choosing **Next Conflict** or **Previous Conflict** from the **Navigate** menu, or by clicking the **Next Conflict** or **Previous Conflict** buttons in the toolbar or Resolution Options pane.
- You can locate one or more objects in the target object repository by using the Find dialog box. For more information, see “Finding Specific Objects” on page 284.
- You can show or hide the target object repository object properties by choosing **View > Target Repository Object Properties**.



Primary and Secondary Repository Panes

The primary and secondary object repository panes display the hierarchies of the objects, and their properties and values, in the original source object repositories that you chose to merge. The file path is shown above each object hierarchy.

The panes provide the following functionality:

- You can expand or collapse the hierarchy of a selected item by double-clicking the item.
- You can view the properties and values of an object in the **Test object details** area by selecting it in the relevant pane.
- You can show or hide the panes by selecting or clearing **Primary Repository** or **Secondary Repository** in the **View** menu.

Resolution Options Pane

The Resolution Options pane provides information about any conflict encountered during the merge for the object selected in the target object repository. The pane also provides options that enable you to keep or change the conflict resolution method that was applied using the default resolution options.

The Resolution Options pane provides the following functionality:

- When you select a conflicting object in the target object repository, the pane displays a textual description of the conflict and the resolution method used by the Object Repository Merge Tool. A choice of alternative resolution methods is offered.
- You can select a radio button to choose an alternative resolution method for the conflict. Every time you make a change, the target object repository is automatically updated and is redisplayed.
- You can jump directly to the next or previous conflict in the target object repository hierarchy by clicking the **Previous Conflict** or **Next Conflict** buttons.

- For a local object repository merge, you can click the **Ignore Object** button to exclude a specific local object repository object from the merge process. The object remains in the local object repository when the merge is complete.
- You can show or hide the pane by selecting or clearing **Resolution Options** in the **View** menu.

Status Bar

The status bar shows the following information about the merge process and the results that are displayed:

- The conflict number (if any) of the object selected in the target object repository pane.
- A progress bar is displayed during the merge process. **Ready** is displayed when the is complete.
- The Quality Center icon is displayed when QuickTest is connected to a Quality Center project.
- The filter status is shown next to the Filter icon: **OFF** indicates that the object repositories are not filtered and all objects are shown. **ON** indicates a filter is active and that some objects may have been filtered out of the display.
- A legend of the icons used in the target object repository pane. The following icons may be displayed:



- Similar Description Conflict
- Same Name Different Description Conflict
- Same Description Different Name Conflict

For more information on conflict types, see “Understanding Object Conflicts” on page 277.

Tips:

- Position your pointer over a conflict icon in the status bar to see a tooltip description of the conflict type.
 - Click any of the conflict icons to view the Statistics dialog box. For more information, see “Viewing Merge Statistics” on page 276.
 - Click the **Filter** icon in the status bar to view the Filter dialog box. The filter is shown as **ON** in the status bar when a filter is currently in use. For more information, see “Filtering the Target Repository Pane” on page 282.
-














Using Object Repository Merge Tool Commands

You can select Object Repository Merge Tool commands from the menu bar or from the toolbar. You can perform certain commands by pressing shortcut keys. You can also select an object in the target object repository pane and choose commands from the context (right-click) menu.

Using Toolbar Commands

You can perform frequently used commands by clicking buttons in the Object Repository Merge Tool toolbar.






	Description
	New Merge (described in “File Menu Commands” on page 258)
	Save (described in “File Menu Commands” on page 258)
	Settings (described in “Tools Menu Commands” on page 261)
	Statistics (described in “View Menu Commands” on page 259)
	Filter (described in “Tools Menu Commands” on page 261)
	Previous Conflict (described in “Navigate Menu Commands” on page 261)
	Next Conflict (described in “Navigate Menu Commands” on page 261)
	Find (described in “Navigate Menu Commands” on page 261)
	Find Previous (described in “Navigate Menu Commands” on page 261)
	Find Next (described in “Navigate Menu Commands” on page 261)
	Quality Center Connection (described in “File Menu Commands” on page 258)

Performing Object Repository Merge Tool Commands

You can perform frequently-used commands by clicking toolbar buttons or choosing the relevant menu option. You can also perform some commands by pressing the relevant shortcut keys.

File Menu Commands


You can manage your merged object repository using the following **File** menu commands:

	Command	Shortcut Key	Function
	New Merge	CTRL+N	Enables you to specify two object repositories with which to perform a new merge operation.
	Save	CTRL+S	Saves the merged shared object repository.
	Save As		Opens the Save Shared Object Repository dialog box, enabling you to specify a name, file type, and storage location for the merged shared object repository.
	Quality Center Connection		Enables you to connect QuickTest to a Quality Center project. For more information, see “Connecting to and Disconnecting from Quality Center” on page 1418.
	Exit		Closes the Object Repository - Merge Tool window. (Also prompts you to save the merged object repository if you did not yet save it.)

View Menu Commands






You can manage the way that the Object Repository Merge Tool is displayed on your screen using the following **View** menu commands:

	Command	Function
	Primary Repository	Displays the Primary Repository File pane, containing a hierarchical view of the objects from the first source object repository that you chose to merge. Also displays the details for each object selected in this pane. For more information, see “Primary and Secondary Repository Panes” on page 254 and “Merging Two Object Repositories” on page 267.
	Secondary Repository	Displays the Secondary Repository File pane, containing a hierarchical view of the objects from the second source object repository that you chose to merge. Also displays the details for each object selected in this pane. For more information, see “Primary and Secondary Repository Panes” on page 254 and “Merging Two Object Repositories” on page 267.
	Target Repository Object Properties	Displays the Object Properties - Target File pane, which displays the details for each test object selected in the target repository pane. For more information, see “Target Repository Pane” on page 252.
	Resolution Options	Displays the Resolution Options pane, which provides information about any conflict that occurred during the merge. For more information, see “Resolution Options Pane” on page 254 and “Resolving Object Conflicts” on page 280.
	Restore Default Layout	Restores the view that you saved using the Set as Default Layout option (described below). This is useful if you resize a pane, or show or hide specific panes and then want to restore your saved view. For more information, see “Changing the View” on page 252.

	Command	Function
	Set as Default Layout	Enables you to save the current view so that each time you open the Object Repository - Merge Too, this view is displayed. If you later modify this view by resizing panes, or showing or hiding them, you can restore your default view using the Restore Default Layout option (described above). For more information, see “Changing the View” on page 252.
	Statistics	Opens the Statistics dialog box, which describes how the files were merged, and the number and type of any conflicts that were resolved during the merge. For more information, see “Viewing Merge Statistics” on page 276.
	Collapse All	Collapses the entire hierarchy in the Target Object Repository pane. Tip: You can collapse a single node by double-clicking it.
	Expand All	Expands the entire hierarchy in the Target Object Repository pane. Tip: You can expand a single node by double-clicking it.



Navigate Menu Commands

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

	Command	Shortcut Key	Function
	Next Conflict	F4	Finds the next conflicting object in the merged object repository.
	Previous Conflict	SHIFT+F4	Finds the previous conflicting object in the merged object repository.
	Find	CTRL+F	Opens the Find dialog box.
	Find Next	F3	Finds the next object in the merged object repository according to the search specifications in the Find dialog box.
	Find Previous	SHIFT+F3	Finds the previous object in the merged object repository according to the search specifications in the Find dialog box.

Tools Menu Commands

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

	Command	Function
	Settings	<p>Opens the Settings dialog box, enabling you to:</p> <ul style="list-style-type: none"> ► Configure how the Object Repository Merge Tool deals with conflicting objects during a merge ► Specify the text color of the object names displayed in the target object repository <p>For more information, see “Defining Default Settings” on page 262.</p>
	Filter	<p>Opens the Filter dialog box, enabling you to show all of the test objects in the Target Repository pane, or show only the objects that had conflicts that were resolved during the merge. For more information, see “Filtering the Target Repository Pane” on page 282.</p>

Help Menu Command

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

Command	Shortcut Key	Function
Object Repository Merge Tool Help	F1	Opens the Object Repository Merge Tool Help.

Defining Default Settings

The Object Repository Merge Tool is supplied with predefined settings that are used when merging object repositories or when updating a shared object repository from local object repositories. These settings:

- Configure how the Object Repository Merge Tool deals with conflicting objects in the primary and secondary object repositories (or local and shared object repositories when updating a shared object repository from local object repositories).
- Specify the text color of the object names that are displayed in the target object repository.

You can change these settings at any time to create new default settings. After you change the settings, all new merges are performed according to the new default settings.

Tip: If you want to change the settings before merging two object repositories, you must click **Cancel** to close the New Merge dialog box, change the settings as described in the next sections, and then perform the merge.

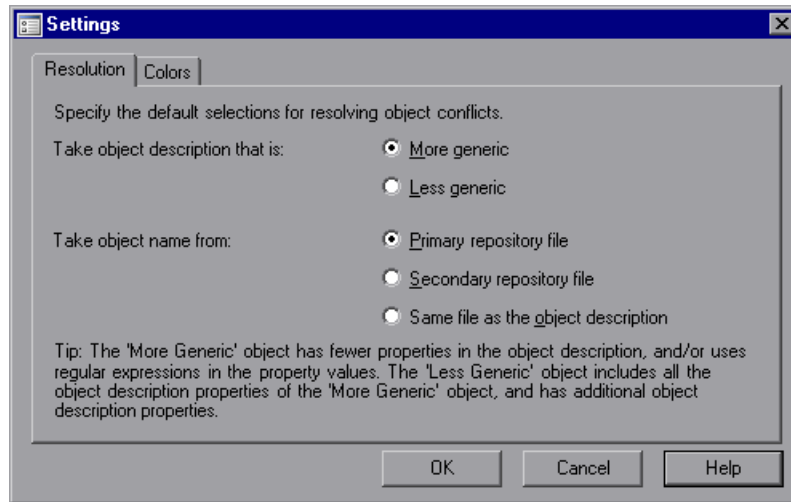
Specifying Default Resolution Settings

You can configure how the Object Repository Merge Tool automatically deals with conflicting objects during the merge process or when performing an **Update from Local Repository** operation.

To specify default resolution settings:



- 1 Select **Tools > Settings** or click the **Settings** button. The Settings dialog box opens.
- 2 Click the **Resolution** tab.



- 3 Select the appropriate radio buttons to specify the default resolution settings that the Object Repository Merge Tool applies when dealing with conflicting objects.
 - **Take object description that is.** Specifies how to resolve conflicts in which two objects have the same name, but their descriptions differ. You can specify that the target object repository takes the object description that is more generic or less generic.
 - **More generic.** Instructs the Object Repository Merge Tool to take the object that has fewer identifying properties than the object with which it conflicts, or uses regular expressions in its property values. This is the default setting.
 - **Less generic.** Instructs the Object Repository Merge Tool to take the object that has all the identifying properties of the object with which it conflicts, plus additional identifying properties.
 - **Take object name from.** Specifies how to resolve conflicts where two objects have the same or similar descriptions, but their names differ. You can select the source from which the target object repository takes the object name:
 - **Primary repository file.** The target object repository takes the object name from the object in the primary object repository. This is the default setting. (When updating a shared object repository from a local object repository, this option is for the **Local object repository**.)
 - **Secondary repository file.** The target object repository takes the object name from the object in the secondary object repository. (When updating a shared object repository from a local object repository, this option is for the **Shared object repository**.)
 - **Same file as the object description.** The target object repository takes the object name from the object in the same object repository from which it took the object description.

Note: When updating a shared object repository from a local object repository, the object repositories are referred to as the Local and Shared object repository.

- 4 Click **OK**. The Object Repository Merge Tool will apply your selections when resolving conflicts between objects in all future object repository merges.

Note: If you make any change to the resolution settings while a merged object repository is open, you are asked whether you want to merge the open files again with the new settings. Click **Yes** to merge the files again with the new settings, or click **No** to keep the existing merge created with the previous settings. If you click **No**, the new settings will apply only to future merges.

Specifying Color Settings

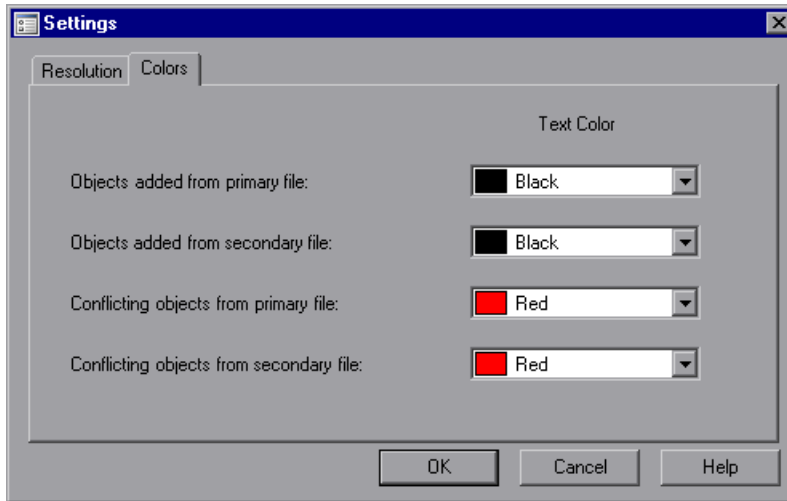
You can specify the color in which object names are displayed in the target object repository according to their source, and whether they caused a conflict. This enables you to see the status of each object more easily.

Note: The options in the Colors tab of the Settings dialog box apply equally to objects added from the local (primary) and shared (secondary) object repositories, when performing an **Update from Local Repository** operation.

To specify color settings:



- 1 Select **Tools > Settings** or click the **Settings** button. The Settings dialog box opens.



- 2 For each item in the Colors tab, click the down arrow ▼ next to the text box and select an identifying color from the Custom, Web, or System tabs.
- 3 Click **OK**. Object names in the target object repository are displayed in the selected color according to your selections.

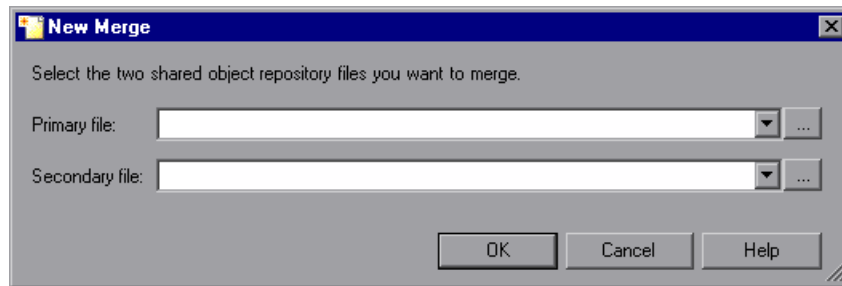
Merging Two Object Repositories

Using the Object Repository Merge Tool, you can merge two source object repositories to create a new shared object repository. Objects in the object repositories are automatically compared and added to the new object repository according to configurable rules that define how conflicts between objects are resolved. The original source files are not changed.

Note: An object repository that is currently open by another user is locked. If you try to merge the locked file, a warning message displays, but you can still perform the merge because the merge process does not modify the source files. Note that changes made to the locked file by the other user may not be included in the merged object repository.

To merge two object repositories:


- 1 In the Object Repository Manager, select **Tools > Object Repository Merge Tool**. The New Merge dialog box opens on top of the Object Repository - Merge Tool window.



Tips:



- If the Object Repository - Merge Tool window is already open, you can select **File > New Merge** or click the **New Merge** button to open the New Merge dialog box.
- If you want to change the configured settings before merging the object repositories, click **Cancel** to close the New Merge dialog box, change the settings as described in “Defining Default Settings” on page 262, and then perform the merge.

-
- 2** In the **Primary file** and **Secondary file** boxes, enter (or browse to) the **.tsr** object repositories that you want to merge into a single object repository. You can click the down arrow  next to each box to view and select recently used files.

Notes:



- It is recommended that you select as your primary object repository the object repository in which you have invested the most effort, meaning the object repository with more objects, object properties, and values.
 - A warning icon is displayed next to the relevant text box if you enter the name of a file without a **.tsr** suffix, a file with an incorrect path, or a file that does not exist. You can position your pointer over the icon to see a tooltip explanation of the error. Enter or select an existing **.tsr** file with the correct path.
 - If you want to merge an object repository that was created using a version of QuickTest earlier than version 9.0, you must first open and save it in the Object Repository Manager to update it to the new format.
 - If you are connected to Quality Center, you can enter (or browse to) object repositories from Quality Center as well as from the file system.
-

- 3 Click **OK**. The Object Repository Merge Tool automatically merges the selected object repositories into a new target object repository according to the configured resolution settings, and displays the results in the Statistics dialog box on top of the Object Repository - Merge Tool window.
- 4 Review the merge statistics, as described in “Viewing Merge Statistics” on page 276, and click **Close**.

In the Object Repository - Merge Tool window, you can:

- Modify any conflict resolutions between objects from the source object repositories, if necessary, as described in “Resolving Object Conflicts” on page 280.
- Filter the objects in the target object repository, as described in “Filtering the Target Repository Pane” on page 282.
- Save the target object repository to the file system or to a Quality Center project, as described in “Saving the Target Object Repository” on page 285.

Updating a Shared Object Repository from Local Object Repositories

You can update a shared object repository by merging local object repositories associated with actions in one or more tests into the shared object repository. The objects that are merged from the local object repositories are then available to any actions that use that shared object repository in any tests.

In the merge process, the objects in the local object repository for the selected action are moved to the target shared object repository. The action then uses the objects from the updated shared object repository.

You can view or change how conflicting objects are dealt with during the update process in the Settings dialog box. For more information, see “Defining Default Settings” on page 262.

If you choose to add local object repositories for more than one action, QuickTest performs multiple merges, merging each action's local object repository with the target object repository one at a time, for all the actions in the list. You can view and modify the results of each merge if necessary.

Notes:

- The Object Repository Merge Tool does not merge checkpoint or output objects from a local object repository into the target shared object repository. You can export checkpoint or output objects from a local object repository to a shared object repository and then manually move the checkpoint and output objects from the exported object repository to your target object repository after you complete the merge process, using the Object Repository Manager.
 - You can merge local object repositories only from actions that are associated with the shared object repository you are updating.
-

To update a shared object repository from a local object repository:

- 1** Select **Resources > Object Repository Manager**. The Object Repository Manager opens.
-

Note: For more information on the Object Repository Manager, see Chapter 7, “Managing Object Repositories.”

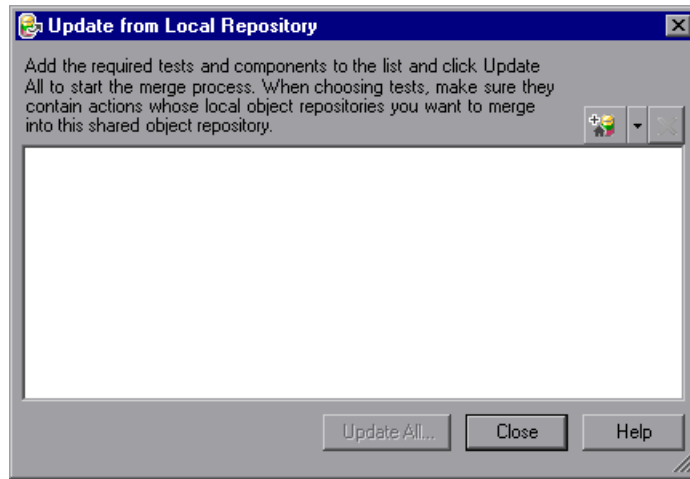



- 2** In the Object Repository Manager, select **File > Open** or click the **Open** button. The Open Shared Object Repository dialog box opens.
- 3** In the sidebar, select the location of the object repository file, for example, File System or Quality Center Resources. Browse to and select the **.tsr** file that contains the shared object repository you want to update, clear the **Open in read-only mode** check box, and click **Open**. The file opens with the objects and properties displayed in editable format.



Tip: If you opened the object repository in read-only mode, select **File > Enable Editing** or click the **Enable Editing** button in the Object Repository Manager toolbar. The object repository file is made editable.

- 4 Select **Tools > Update from Local Repository**. The Update from Local Repository dialog box opens.



- 5 Click the down arrow  next to the **Add Tests** button, and select **Browse for Test**. The Open Test dialog box opens.

In the sidebar, select the location of the test containing actions whose local object repositories you want to merge into the shared object repository, for example, **File System** or **Quality Center Test Plan**, and then select the test.

You can only add a test containing actions that are associated with the shared object repository you are updating and whose local object repositories contain objects.

- 6 Repeat step 5 to add additional tests if required.



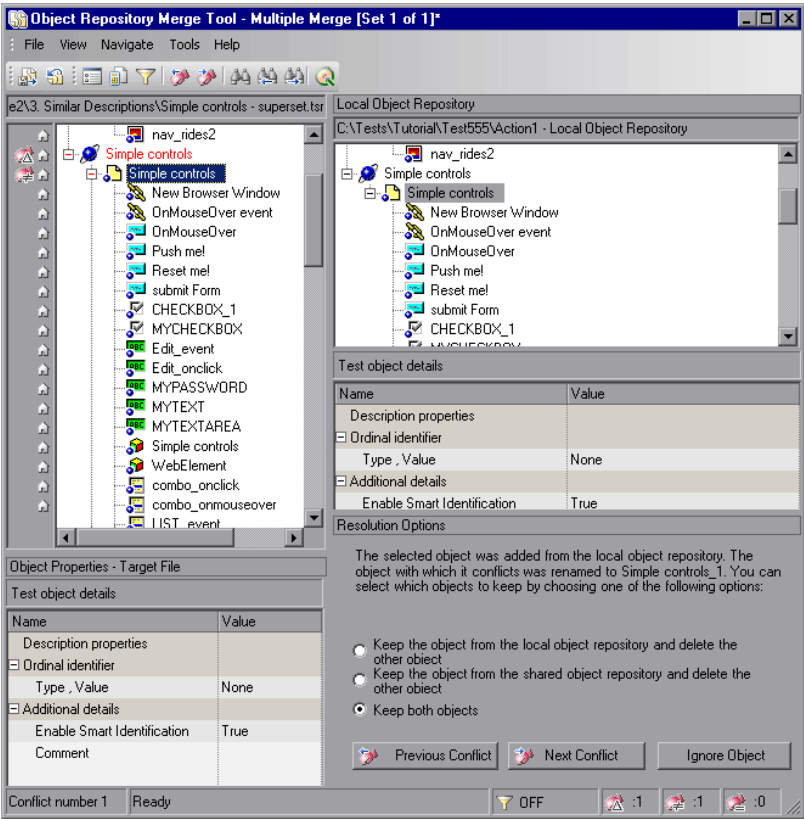
Note: The local object repositories associated with all the actions contained in the listed tests are included in the merge. If you want to remove an action from the merge, select it in the list and click **Delete**.

- 7 Click **Update All**. QuickTest automatically merges the first action local object repository into the shared object repository according to the configured settings, and displays the results in the Statistics dialog box on top of the Object Repository Merge Tool window.


Note: Before each merge, QuickTest checks whether the local object repository is in use by another user. If so, the local object repository is locked and the objects for the selected action cannot be moved to the target shared object repository. A warning message is displayed. The merge can be performed when the local object repository is no longer in use by the other user.


- 8 Review the merge statistics, as described in “Viewing Merge Statistics” on page 276, and click **Close**.

The Object Repository - Merge Tool window for a local object repository merge displays the local object repository as the primary object repository, and the shared object repository as the target object repository.



At the left of each object in the target object hierarchy is an icon that indicates the source of the objects:

 indicates that the object was added from the local object repository.

 indicates that the object already existed in the shared object repository.

Note: If you specified more than one action in the Update from Local Repository dialog box, QuickTest performs multiple merges, merging each action's local object repository with the target object repository one at a time. The Statistics dialog box and the Object Repository Merge Tool - Multiple Merge window displayed after this step show the merge results of the first merge (the local object repository of the first action being merged into the shared object repository). QuickTest enables you to view, and modify if necessary, the results of each merge in sequence. The number of each merge set in a multiple merge is displayed in the title bar, for example, [Set 2 of 3].

- 9 For each object merged into the shared object repository, you can accept the automatic merge or use the Resolution Options pane to:
- Keep a specific object from the shared object repository and delete the conflicting object from the local object repository.
 - Keep a specific object from the local object repository and delete the conflicting object from the shared object repository.
 - Keep conflicting objects from both the shared object repository and the local object repository.
 - Exclude a specific local repository object from the merge process so that it is not included in the shared object repository. Select the object in the Shared Object Repository pane and click **Ignore Object** at the bottom of the Resolution Options pane. The object is removed from the shared object repository and grayed in the local object repository tree. It remains in the action's local object repository when the merge is complete.

Notes:

- The **Ignore Object** button is only visible in the Merge Tool window for a local object repository merge, and is only enabled when an object in the local object repository is selected.
 - The **Ignore Object** operation cannot be reversed. To include the object again in the merge process, you must repeat the merge by clicking **Revert to Original Merged Files** in the toolbar.
-

For more information, see “Resolving Object Conflicts” on page 280.



- 10 If you are performing multiple merges, click the **Save and Merge Next** button in the Object Repository Merge Tool toolbar to perform the next merge (the local object repository of the next action being merged into the shared object repository).
- 11 Click **Yes** to save your changes between merges. If you click **No**, the current merge (objects merged from the last action) will not be saved.
- 12 Repeat steps 8 to 11 to complete the multiple merges.
- 13 Select **File > Exit**, then click **Yes** to save the updated object repository.

Viewing Merge Statistics

After you merge two object repositories, the Object Repository Merge Tool displays the Statistics dialog box, which describes how the files were merged, and the number and type of any conflicts that were resolved during the merge.



Note: The Statistics dialog box shown after performing an **Update from Local Repository** merge differs slightly from the dialog box shown above.



Tip: You can view the merge statistics in the Statistics dialog box at any time by choosing **View > Statistics** in the Object Repository - Merge Tool window, by clicking the **Statistics** button in the toolbar, or by clicking a conflict icon in the status bar.

The Statistics dialog box displays the following information:

- The number and type of any conflicts between the objects added to the target object repository. Conflict types are described in “Resolving Object Conflicts” on page 280.
- The number of items added to the target object repository that are unique in each of the primary or secondary (or local) files, or are identical in both files.

Tip: Select the **Go to first conflict** check box to jump to the first conflict in the target object repository immediately after you close the Statistics dialog box.

Understanding Object Conflicts

Merging two object repositories can result in conflicts arising from similarities between the objects they contain. The Object Repository Merge Tool identifies three possible conflict types:



- **Similar Description Conflict.** Two objects that have the same name and the same object hierarchy, but that have slightly different descriptions. In this conflict type, one of the objects always has a subset of the properties set of the other object. These conflicts are described on page 278.

By default, the conflict resolution settings for conflicts of this type are configured so that the target object repository takes the object that has fewer identifying properties than the object with which it conflicts. For information on changing the default settings, see “Defining Default Settings” on page 262.



- **Same Name Different Description Conflict.** Two objects that have the same name and the same object hierarchy, but differ somehow in their description (for example, they have different properties, or the same property with different values). These conflicts are described on page 279.

By default, the conflict resolution settings for conflicts of this type are configured so that the target object repository takes the object from both files. The object that is added from the secondary file is renamed by adding an incremental numeric suffix to the name, for example, `Edit_1`. For information on changing the default settings, see “Defining Default Settings” on page 262.



- **Same Description Different Name Conflict.** Two objects that have identical descriptions and have the same object hierarchy, but differ in their object names. These conflicts are described on page 280.

By default, the conflict resolution settings for conflicts of this type are configured so that the target object repository takes the object name from the primary source file. For information on changing the default settings, see “Defining Default Settings” on page 262.

Note: Objects that do not have a description, such as Page or Browser objects, are compared by name only. If the same object is contained in both the source object repositories but with different names, they will be merged into the target object repository as two separate objects.

Similar Description Conflict

An object in the primary object repository and an object in the secondary object repository have the same name, and they have similar, but not identical, description properties and values. One of the objects always has a subset of the properties set of the other object. For example, an object named `Button_1` in the secondary object repository has the same description properties and values as an object named `Button_1` in the primary object repository, but also has additional properties and values.

You can resolve this conflict type by:

- Keeping the object added from the primary object repository only.
- Keeping the object added from the secondary object repository only.
- Keeping the objects from both object repositories. In this case, the Object Repository Merge Tool automatically renames the object that is added from the secondary file by adding an incremental numeric suffix to the name, for example, Edit_1.
- Ignoring the object from the local object repository and keeping the object from the shared object repository (when updating a shared object repository from a local object repository).

Same Name Different Description Conflict

An object in the primary object repository and an object in the secondary object repository have the same name, but completely different description properties and values.

You can resolve this conflict type by:

- Keeping the object added from the primary object repository only.
- Keeping the object added from the secondary object repository only.
- Keeping the objects from both object repositories. In this case, the Object Repository Merge Tool automatically renames the object that is added from the secondary file by adding an incremental numeric suffix to the name, for example, Edit_1.
- Ignoring the object from the local object repository and keeping the object from the shared object repository (when updating a shared object repository from a local object repository).

Same Description Different Name Conflict

An object in the primary object repository and an object in the secondary object repository have different names, but the same description properties and values.

You can resolve this conflict type by:

- Taking the object name from the object in the primary object repository.
- Taking the object name from the object in the secondary object repository.
- Ignoring the object from the local object repository and keeping the object from the shared object repository (when updating a shared object repository from a local object repository).

Resolving Object Conflicts

Conflicts between objects in the primary and secondary object repositories are resolved automatically by the Object Repository Merge Tool according to the default resolution settings that you can configure before performing the merge. For more information, see “Defining Default Settings” on page 262.

However, the Object Repository Merge Tool also allows you to change the way the merge was performed for each individual object that causes a conflict.

For example, an object in the primary object repository could have the same name as an object in the secondary object repository, but have a different description. You may have defined in the default settings that in this case, the object with the more generic object description, meaning the object with fewer properties, should be added to the target object repository. However, when you review the conflicts after the automatic merge, you could decide to handle the specific conflict differently, for example, by keeping both objects.

Note: Changes that you make to the default conflict resolution can themselves affect the target object repository by causing new conflicts. In the above example, keeping both objects would cause a name conflict. Therefore, the target object repository is updated after each conflict resolution change and redisplayed.

You can identify objects that caused conflicts, and the conflict type, by the icon displayed to the left of the object name in the target object repository pane of the Object Repository Merge Tool and the text color. When you select a conflicting object, a full description of the conflict, including how it was automatically resolved by the Object Repository Merge Tool, is displayed in the Resolutions Options pane.

The Resolutions Options pane offers alternative resolution options. You can choose to keep the default resolution if it suits your needs, or use the alternative options to resolve the conflict in a different way. In addition, for a local object repository merge, you can click the **Ignore Object** button to exclude a specific local object repository object from the target shared object repository.

Tip: You can also change the default resolution settings and merge the files again. For more information, see “Defining Default Settings” on page 262.

To change the way in which object conflicts are resolved:

- 1** In the target object repository, select an object that had a conflict, as indicated by the icon to the left of the object name. The conflicting objects are highlighted in the source object repositories.

A description of the conflict and the resolution method used by the Object Repository Merge Tool is described in the Resolution Options pane. A radio button for each possible alternative resolution method is displayed. For information on each of the conflict types, see “Understanding Object Conflicts” on page 277.

- 2** In the Resolution Options pane, select a radio button to choose an alternative resolution method. The target object repository is updated according to your selection and redisplayed.
- 3** In the Resolution Options pane, click the **Previous Conflict** or **Next Conflict** buttons to jump directly to the next or previous conflict in the target object repository hierarchy.
- 4** Repeat steps 1 to 3 to modify additional conflict resolutions, as necessary.
- 5** Save the target object repository, as described in “Saving the Target Object Repository” on page 285.

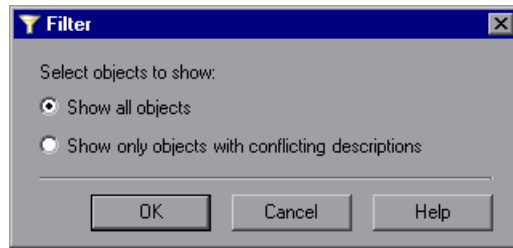
Filtering the Target Repository Pane

Merging two object repositories can result in a target object repository containing a large number of objects. To make navigation and the location of specific objects easier in the target object repository pane, the Object Repository Merge Tool enables you to filter the objects in the pane and show only the objects that had conflicts that were resolved during the merge.

Note: The filter only affects which objects are displayed in the target object repository pane. It does not affect which objects are included in the target object repository.

To filter the objects in the target object repository pane:

- 1 Select **Tools > Filter** or click the **Filter** button. The Filter dialog box opens.



Tip: You can also click the **Filter** icon in the status bar to view the Filter dialog box. The Filter is shown as **ON** in the status bar when a filter is currently in use.

- 2 Select a radio button according to the objects you want to view in the target object repository.
 - **Show all objects.** Shows all objects in the target object repository
 - **Show only objects with conflicting descriptions.** Shows only objects in the target object repository that have description conflicts
- 3 Click **OK**. The objects in the pane are filtered and the target object repository displays only the requested object types. A progress bar is displayed in the status bar during the filter process.

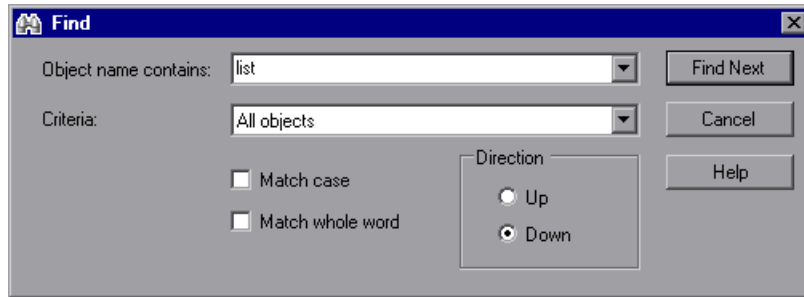
Finding Specific Objects

You can use the Find feature in the Object Repository Merge Tool to locate one or more objects in the target object repository whose name contains a specified string. The located object is also highlighted in the relevant primary and/or secondary object repositories.

To find an object:



- 1 Select **Navigate > Find** or click the **Find** button. The Find dialog box opens.



- 2 In the **Object name contains** box, enter the full or partial name of the object you want to find.
- 3 In the **Criteria** box, refine your search by selecting which objects to search. The following criteria are available:
 - **All objects**
 - **Objects from one source**
 - **Objects with conflicts**
 - **Objects with conflicts or from one source**
- 4 Select one or both of the following options to help fine-tune your search:
 - **Match case.** Distinguishes between upper-case and lower-case characters in the search. When **Match case** is selected, QuickTest finds only those occurrences in which the capitalization exactly matches the text you entered in the **Object name contains** box.
 - **Match whole word.** Searches for occurrences that are whole words only and not part of larger words.

- 5 Specify the direction from the current cursor location in which you want to search: **Up** or **Down**. The Find operation will continue to search the entire object repository after it reaches the beginning or end of the file.
- 6 Click **Find Next** to highlight the next object that matches the specified criteria in the target object repository.

You can also close the Find dialog box and use the following commands:



- Click the **Find Next** button or select **Navigate > Find Next** to highlight the next object that matches the specified criteria.



- Click the **Find Previous** button or select **Navigate > Find Previous** to highlight the previous object that matches the specified criteria.

Saving the Target Object Repository

When you are sure that the object conflicts are resolved satisfactorily, you can save the target object repository to the file system or to a Quality Center project (if QuickTest is currently connected to the Quality Center project).

Saving the Object Repository

You can save the new merged shared object repository to the file system. If you are connected to Quality Center, you can also save your merged shared object repository in the Test Resources module of your project.

To save an object repository to the file system:



- 1 Select **File > Save** or click the **Save** button. If the file was saved previously, the current changes you made are saved. If the file has not yet been saved, the Save Shared Object Repository dialog box opens.
- 2 In the sidebar, select the location in which you want to save the file, for example, File System or Quality Center Test Resources.
- 3 Browse to and select the folder in which you want to save the file.

- 4 In the **File name** box, enter a name for the file. Use a descriptive name that will help you easily identify the file. Do not use any of the following characters in the object repository name:

\ / : * " ? < > | '

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

- 5 Click **Save**.

Tip: If you want to save the file as an attachment to a test in the Test Plan module in Quality Center, select **Quality Center Test Plan** in the sidebar, browse to and double-click the test, and then click **Save**.

QuickTest saves the object repository with a **.tsr** extension in the specified location and displays the file name and path above the target object repository in the Object Repository - Merge Tool window.

9

Comparing Shared Object Repositories

QuickTest Professional enables you to compare two shared object repositories using the Object Repository Comparison Tool, and view the differences in their objects, such as different object names, different object descriptions, and so on.

This chapter includes:

- About Comparing Shared Object Repositories on page 288
- Understanding the Object Repository Comparison Tool on page 289
- Using Object Repository Comparison Tool Commands on page 293
- Understanding Object Differences on page 297
- Changing Color Settings on page 298
- Comparing Object Repositories on page 299
- Viewing Comparison Statistics on page 301
- Filtering the Repository Panes on page 302
- Synchronizing Object Repository Views on page 303
- Finding Specific Objects on page 304

Tip: If you are connected to a Quality Center 10.00 project with version control enabled, you can compare two versions of the same object repository. For more information, see “Viewing and Comparing Versions of QuickTest Assets” on page 1461.

About Comparing Shared Object Repositories

QuickTest Professional enables you to compare existing assets from two different object repositories using the Object Repository Comparison Tool. The tool is accessible from the Object Repository Manager, and enables you to compare different object repository resources, or different versions of the same object repository resource, and identify similarities, variations, or changes.

Differences between objects in the two object repository files, named the **First** and **Second** files, are identified according to default rules. During the comparison process, the object repository files remain unchanged. For more information about the types of differences identified by the Object Repository Comparison Tool, see “Understanding Object Differences” on page 297.

After the compare process, the Comparison Tool provides a graphic presentation of the objects in the object repositories, which are shown as nodes in a hierarchy. Objects that have differences, as well as unique objects that are included in one object repository only, can be identified according to a color configuration that you can select. Objects that are included in one object repository only are identified in the other object repository by the text “Does not exist”. You can also view the properties and values of each object that you select in either object repository.

You can use the information displayed by the Object Repository Comparison Tool when managing or merging object repositories. For more information, see Chapter 9, “Comparing Shared Object Repositories,” or Chapter 8, “Merging Shared Object Repositories.”

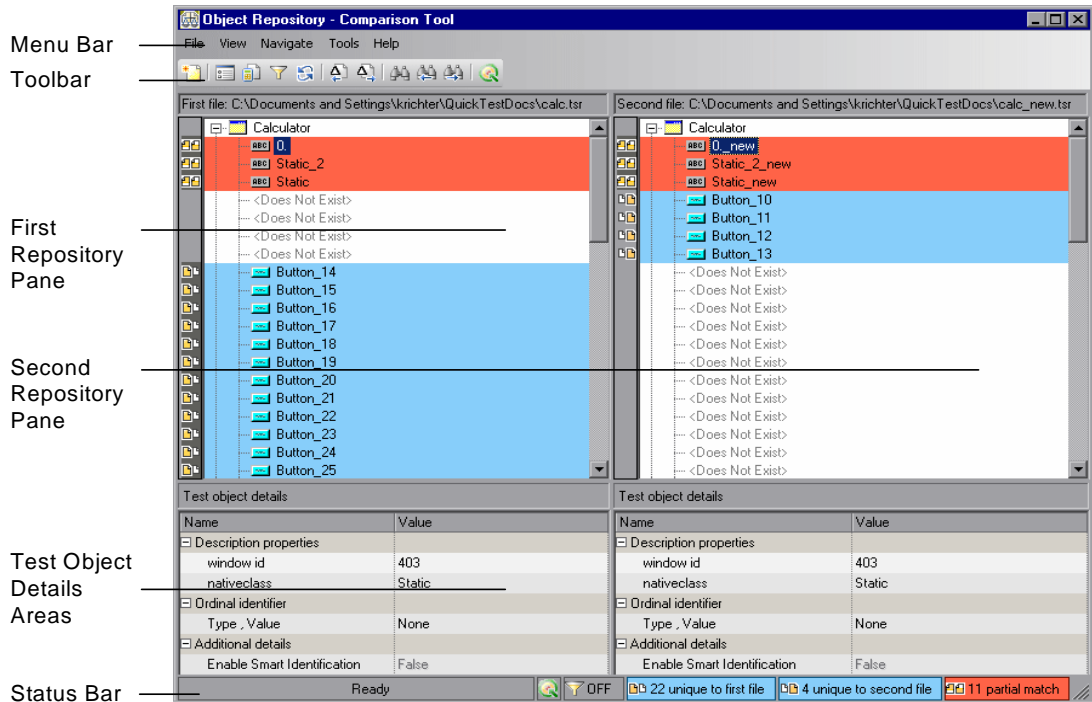
Notes:

- The Object Repository Comparison Tool does not compare checkpoint or output objects.
 - You cannot work with the Object Repository Manager or the Object Repository Merge Tool when the Object Repository Comparison Tool is open.
-

Understanding the Object Repository Comparison Tool

You open the Object Repository Comparison Tool by choosing **Tools > Object Repository Comparison Tool** in the Object Repository Manager.

An example window of the Object Repository - Comparison Tool is shown below:



The Object Repository - Comparison Tool window contains the following key elements:

- **Menu bar.** Displays menus of Object Repository Comparison Tool commands. These commands are described in various places throughout this chapter. Shortcut keys for menu commands are described in “Object Repository Comparison Tool Menu Commands and Shortcut Keys” on page 294.
- **Toolbar.** Contains buttons of commonly used menu commands to assist you in comparing your object repositories and viewing the similarities and differences in their objects. Toolbar buttons are described in “Object Repository Comparison Tool Toolbar Commands” on page 293.
- **Repository Panes.** Display a hierarchical view of the objects in the object repositories being compared. In the column to the left of the object hierarchies, each pane displays icons representing the comparison of each object. For more information, see “Understanding the Repository Panes” on page 290.
- **Test Object Details areas.** Show the properties and values of the object selected in an object repository pane. For more information, see “Understanding the Repository Panes” on page 290.
- **Status Bar.** Shows the status of the comparison process and details of the differences found during the object repository comparison. For more information, see “Understanding the Status Bar” on page 292.




Understanding the Repository Panes

The object repository panes display the hierarchies of the objects, and their properties and values, in the object repository files that you are comparing. The file path is shown above each object hierarchy.

To make it easier to see the status of an object at a glance, the text and background of object names in the object repositories are displayed using different colors, according to the type of difference found.

You can change the default colors used in the object repositories to indicate the difference type. For more information, see “Changing Color Settings” on page 298.

Differences can also be identified by the icons used to the left of the objects in the object repository panes, as follows:

-  ➤ Objects that are unique to the first file
-  ➤ Objects that are unique to the second file
-  ➤ Objects in both the first and second file that are not identical but partially match

For more information on all difference types, see “Understanding Object Differences” on page 297.

The object repository panes provide the following functionality:

- When you select an object in one object repository pane, the corresponding object in the other file hierarchy is located and highlighted. You can press the CTRL button when you select an object to highlight only the selected object without highlighting the corresponding object in the other file.
- When you select an object in an object repository pane, its properties and values are displayed in the respective **Test object details** area at the bottom of the pane.
- When you position your cursor over an icon to the left of an object in an object repository pane, the comparison details are displayed as a tooltip, for example, Partial match, or Unique to second file.
- You can expand or collapse the hierarchy of a parent node by double-clicking the node, or by clicking the expand (+) or collapse (-) symbol to the left of the node name. You can also expand or collapse the entire hierarchy in the object repository pane by choosing **Collapse All** or **Expand All** from the **View** menu.








- You can jump directly to the next or previous difference in the object repository hierarchy by choosing **Next Difference** or **Previous Difference** from the **Navigate** menu, by clicking the **Next Difference** or **Previous Difference** buttons in the toolbar, or by using keyboard shortcuts. For more information about shortcuts, see “Object Repository Comparison Tool Menu Commands and Shortcut Keys” on page 294.

- You can locate one or more objects in the object repository panes by using the Find dialog box. For more information, see “Finding Specific Objects” on page 304.
- You can drag the edges of the panes to resize them in the Object Repository Comparison Tool window.

Understanding the Status Bar

The status bar shows information about the comparison process and the results that are displayed:

- A progress bar is displayed on the left of the status bar during the comparison process. **Ready** is displayed when the process is complete.
-  ➤ The Quality Center icon is displayed when QuickTest is connected to a Quality Center project.
-  ➤ The filter status is shown next to the Filter icon: **OFF** indicates that the object repositories are not filtered and all objects are shown. **ON** indicates a filter is active and that some objects may have been filtered out of the display. You can click the **Filter** icon to view the Filter dialog box. For more information, see “Filtering the Repository Panes” on page 302.
- The number of differences found during the comparison are displayed, as follows:
 -  ➤ The number of objects that are unique to the first file
 -  ➤ The number of objects that are unique to the second file
 -  ➤ The number of objects in the first and second file that are not identical but partially match

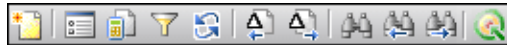
For more information on all difference types, see “Understanding Object Differences” on page 297.

Using Object Repository Comparison Tool Commands

You can select Object Repository Comparison Tool commands from the menu bar or from the toolbar. You can also perform certain commands by pressing shortcut keys.

Object Repository Comparison Tool Toolbar Commands

You can perform frequently used commands by clicking buttons in the toolbar.





	Description
	New Comparison (described in “File Menu Commands” on page 294)
	Color Settings (described in “Tools Menu Commands” on page 296)
	Statistics (described in “View Menu Commands” on page 295)
	Filter (described in “Tools Menu Commands” on page 296)
	Synchronized Nodes (described in “Navigate Menu Commands” on page 295)
	Previous Difference (described in “Navigate Menu Commands” on page 295)
	Next Difference (described in “Navigate Menu Commands” on page 295)
	Find (described in “Navigate Menu Commands” on page 295)
	Find Previous (described in “Navigate Menu Commands” on page 295)
	Find Next (described in “Navigate Menu Commands” on page 295)
	Quality Center Connection (described in “File Menu Commands” on page 294)

Object Repository Comparison Tool Menu Commands and Shortcut Keys

You can perform frequently-used commands by clicking toolbar buttons or choosing the relevant menu option. You can also perform some commands by pressing the relevant shortcut keys.


File Menu Commands

You can manage your object repository comparison using the following **File** menu commands:

	Command	Shortcut Key	Function
	New Comparison	CTRL+N	Enables you to specify two object repositories on which to perform a new comparison operation.
	Quality Center Connection		Enables you to connect QuickTest to a Quality Center project. For more information, see “Connecting QuickTest to Quality Center” on page 1418.
	Exit		Closes the Object Repository - Comparison Tool window.




View Menu Commands



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

	Command	Function
	Statistics	Opens the Statistics dialog box, which describes the comparison between the two repositories, including the number and type of any differences found. For more information, see “Viewing Comparison Statistics” on page 301.
	Collapse All	Collapses the entire hierarchy in both comparison panes. Tip: Double-clicking an expanded node collapses it in both panes simultaneously.
	Expand All	Expands the entire hierarchy in both comparison panes. Tip: Double-clicking a collapsed node expands it in both panes simultaneously.

Navigate Menu Commands




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

	Command	Shortcut Key	Function
	Next Difference	F4	Finds the next difference between objects in the object repositories.
	Previous Difference	SHIFT+F4	Finds the previous difference between objects in the object repositories.
	Find	CTRL+F	Opens the Find dialog box.

	Command	Shortcut Key	Function
	Find Next	F3	Finds the next object in the object repositories according to the search specifications in the Find dialog box.
	Find Previous	SHIFT+F3	Finds the previous object in the object repositories according to the search specifications in the Find dialog box.

Tools Menu Commands

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

	Command	Function
	Synchronized Nodes	Enables you to navigate the two object repository panes simultaneously or independently of one another. For more information, see “Synchronizing Object Repository Views” on page 303.
	Filter	Opens the Filter dialog box, enabling you to specify the types of test object matches that you want to show. For more information, see “Filtering the Repository Panes” on page 302.
	Color Settings	Opens the Settings dialog box, enabling you to specify the text color and background of the object names and empty nodes displayed in the comparison panes. For more information, see “Changing Color Settings” on page 298.

Help Menu Command

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

Command	Shortcut Key	Function
Object Repository Comparison Tool Help	F1	Opens the Object Repository Comparison Tool Help.

Understanding Object Differences

The Comparison Tool automatically identifies objects during the comparison process by classifying them into one of the following types:

- **Identical.** Objects that appear in both object repository files. There is no difference in their name or in their properties.
- **Matching description, different name.** Objects that appear in both object repository files that have different names, but the same description properties and values.
- **Similar description.** Objects that appear in both object repository files that have similar, but not identical, description properties and values. One of the objects always has a subset of the properties set of the other object. This implies that it is likely to be a less detailed description of the same object. For example, an object named `Button_1` in the second object repository has the same description properties and values as an object named `Button_1` in the first object repository, but also has additional properties and values.

Objects that do not have a description, such as `Page` or `Browser` objects, are compared by name only. If the same object is contained in both the object repositories but with different names, they will be shown in the object repositories as two separate objects.

Note: The Object Repository Comparison Tool gives precedence to matching object descriptions over the matching of object names. For this reason, certain object nodes may be linked during the comparison process and not others.

- **Unique to first file, or Unique to second file.** Objects that appear in only one of the object repository files.
- **Does not exist.** Objects that do not exist in one of the repository files, but do exist in the other file.

Changing Color Settings

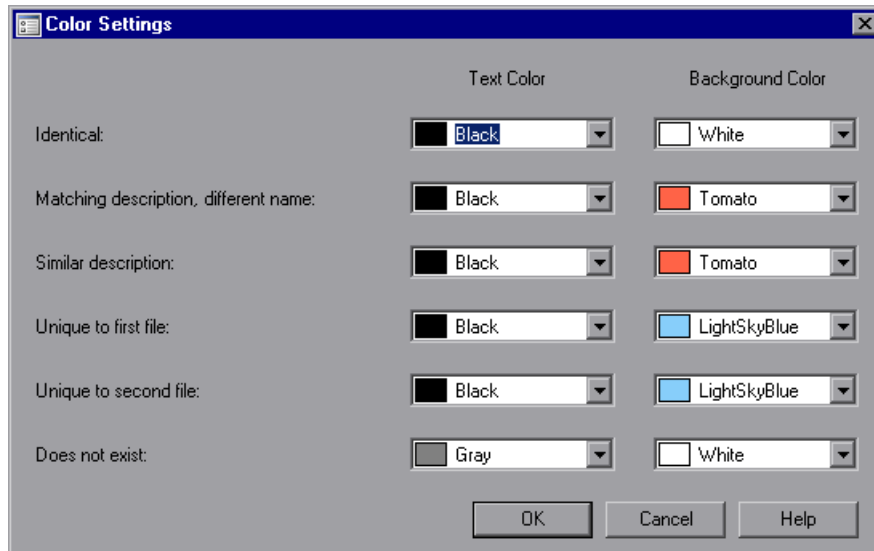
The text and background of object names, and empty nodes representing objects that exist in the other object repository only, are displayed in the Comparison Tool window in default colors, according to their difference types. This enables you to see the status of each object in the object repository panes. These text colors are also used in the Statistics dialog box.


You can change the default color settings if required.

To change color settings:



- 1 Select **Tools > Color Settings** or click the **Color Settings** button in the toolbar. The Color Settings dialog box opens.



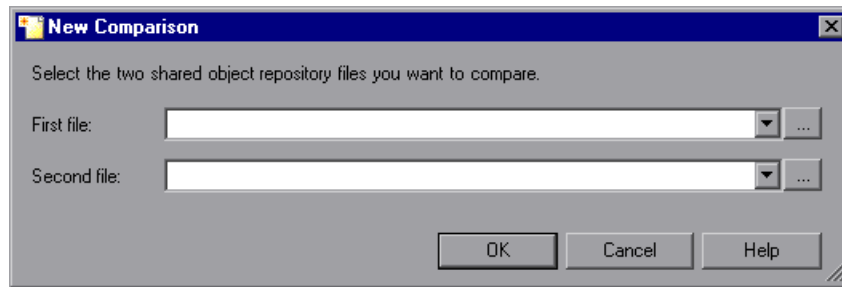
- 2 For each difference type, click the down arrow  next to the text box and select an identifying text color and background color from the Custom, Web, or System tabs.
- 3 Click **OK**. After performing a comparison of object repositories, object names and empty nodes in the respective object repository panes are displayed according to your selections.

Comparing Object Repositories

Using the Object Repository Comparison Tool, you can compare two object repositories according to predefined settings that define how differences between objects are identified.

To compare two object repositories:


- 1 In QuickTest Professional, select **Resources > Object Repository Manager**.
- 2 In the Object Repository Manager, select **Tools > Object Repository Comparison Tool**. The New Comparison dialog box opens on top of the Object Repository - Comparison Tool window.



Tips:



- If the Object Repository - Comparison Tool window is already open, you can select **File > New Comparison** or click the **New Comparison** button in the toolbar to open the New Comparison dialog box.
 - If you want to change the color settings before comparing the object repositories, click **Cancel** to close the New Comparison dialog box, change the settings as described in “Changing Color Settings” on page 298, and then perform the comparison.
-

- 3 In the **First file** and **Second file** boxes, enter or browse to and select the **.tsr** object repository files that you want to compare. The object repository files can be located in the file system or Quality Center. By default, the boxes display the last files selected for comparison using the Object Repository Comparison Tool. You can click the down arrow  next to each box to view and select recently used files.

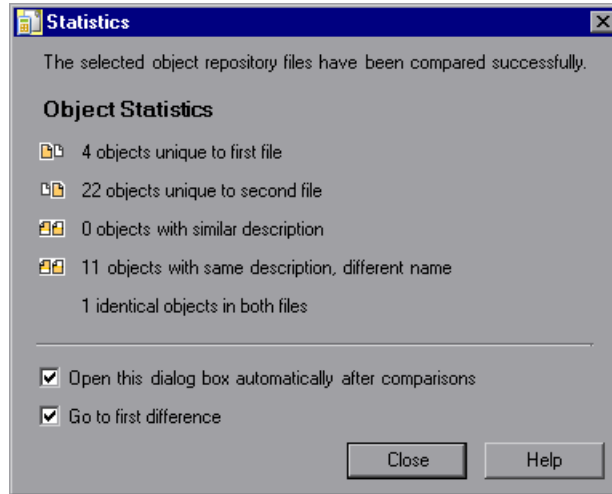
Notes:



- A warning icon is displayed next to the relevant text box if you enter the name of a file without a **.tsr** suffix, a file with an incorrect path, or a file that does not exist. You can position your pointer over the icon to see a tooltip explanation of the error. Enter or select an existing **.tsr** file with the correct path.
 - If you want to compare an object repository that was created using a version of QuickTest earlier than version 9.0, you must first open and save it in the Object Repository Manager to update it to the new format.
 - If you are connected to Quality Center, you can enter (or browse to) object repositories from Quality Center as well as from the file system.
-
- 4 Click **OK**. The Object Repository Comparison Tool compares the objects in the selected object repositories and displays the results in the Statistics dialog box on top of the Object Repository - Comparison Tool window.
 - 5 Review the statistics, as described in “Viewing Comparison Statistics” on page 301, and click **Close**.
 - 6 In the Object Repository - Comparison Tool window, you can:
 - Filter the objects in the object repositories, as described in “Filtering the Repository Panes” on page 302.
 - Find specific objects in the object repositories, as described in “Finding Specific Objects” on page 304.

Viewing Comparison Statistics

After you compare two object repositories, the Object Repository Comparison Tool displays the Statistics dialog box, which describes how the files were compared, and the number and type of any differences found.



Tip: You can choose not to view the Statistics dialog box every time you compare object repositories by clearing the **Open this dialog box automatically after comparisons** check box. You can view the comparison statistics in the Statistics dialog box at any time by choosing **View > Statistics** in the Comparison Tool window, or by clicking the **Statistics** button in the toolbar.

The Statistics dialog box displays the following information:

- The number and type of any differences between the objects in the object repositories. Difference types are described in “Understanding Object Differences” on page 297.
- The number of items that are unique to the first or the second file, or are identical in both files.

The icons displayed for each difference type in the object statistics are the same as those used in the object repository panes. For more information, see “Understanding the Repository Panes” on page 290.

Tip: Select the **Go to first difference** check box to jump to the first difference in the object repositories immediately after you close the Statistics dialog box.

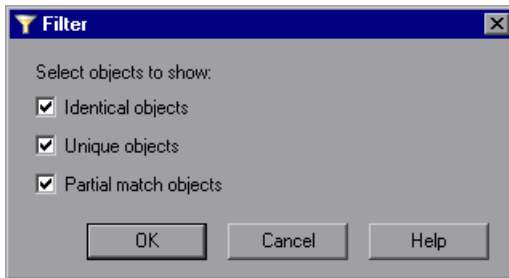
Filtering the Repository Panes

Object repositories can contain a large number of objects. To make navigation and the location of specific objects easier in the object repository panes, the Object Repository Comparison Tool enables you to filter the objects and show only the objects that you want to view.

To filter the objects in the object repository panes:



- 1 Select **Tools > Filter** or click the **Filter** button in the toolbar. The Filter dialog box opens.



Tip: The **Filter** button in the toolbar is surrounded by a border when a filter is currently in use. In addition, the filter is shown as **ON** in the status bar. You can click the **Filter** icon in the status bar to open the Filter dialog box.

- 2 Select one or more check boxes according to the objects you want to view in the object repositories.
 - **Identical Objects.** Objects that appear in both object repository files and have no differences in their name or in their properties
 - **Unique objects.** Objects that appear only in the first object repository file or only in the second object repository file
 - **Partial match objects.** Objects in the object repository files that match but have name or description differences

Tip: Select all the check boxes to view all the objects in both object repositories.

- 3 Click **OK**. The objects in the panes are filtered and the object repositories display only the requested object types.

Synchronizing Object Repository Views

The Object Repository Comparison Tool enables you to navigate the two object repositories independently. You can also resize the various panes to display only some of the objects contained in the object repositories. When using large object repositories, this can result in the various panes displaying different areas of the object repository hierarchies, making it difficult to locate and track specific objects affected by the comparison process.



To synchronize the object repositories to display the same object in both views, select the object in the first or second object repository in which it is currently visible and click the **Synchronized Nodes** button in the toolbar. The matching node is highlighted in the other object repository and both object repositories scroll simultaneously.

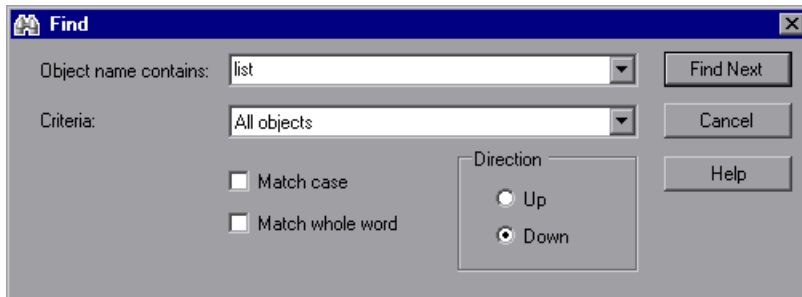
Tip: The **Synchronized Nodes** button in the toolbar is surrounded by a border when the object repositories are currently synchronized. Click the **Synchronized Nodes** button again to navigate the two object repositories independently. When the object repositories are synchronized, you can also press the CTRL button while selecting an object to highlight the selected object only.


Finding Specific Objects

You can use the Find feature in the Object Repository Comparison Tool to locate one or more objects in a selected object repository whose name contains a specified string. The located object is also highlighted in the other object repository if it exists there.

To find an object:

- 1 Click the object repository pane that contains the required object.
- 2 Select **Navigate > Find** or click the **Find** button in the toolbar. The Find dialog box opens.



- 3 In the **Object name contains** box, enter the full or partial name of the object you want to find. You can click the down arrow  next to the box to view and select a recently used string.

4 In the **Criteria** box, refine your search by selecting which objects to search. The following criteria are available:

- **All objects**
- **Unique objects**
- **Partial match objects**
- **Unique or partial match objects**

5 Select one or both of the following options to help fine-tune your search:

- **Match case.** Distinguishes between upper-case and lower-case characters in the search. When **Match case** is selected, QuickTest finds only those occurrences in which the capitalization exactly matches the text you entered in the **Object name contains** box.
- **Match whole word.** Searches for occurrences that are whole words only and not part of larger words.

6 Specify the direction from the current cursor location in which you want to search: **Up** or **Down**. The Find operation will continue to search the entire file after it reaches the beginning or end of the object repository.

7 Click the **Find Next** button to highlight the next object that matches the specified criteria in the object repository.

You can also close the Find dialog box and use the following commands:



- Click the **Find Next** button in the toolbar, select **Navigate > Find Next**, or press F3, to highlight the next object that matches the specified criteria.



- Click the **Find Previous** button in the toolbar, select **Navigate > Find Previous**, or press SHIFT+F3, to highlight the previous object that matches the specified criteria.

Part III

Designing Tests

10

Creating Tests — Overview

You can create tests using the keyword-driven methodology, step recording, or a combination of both. The keyword-driven methodology enables you to select keywords to indicate the operations you want to perform on your application. Step recording enables you to record the operations you perform on your application.

After you create your tests, you can enhance them using checkpoints and other special testing options.

This chapter includes:

- About Creating Tests on page 309
- Deciding Which Methodology to Use - Keyword-Driven or Recording on page 311
- Understanding Your Test on page 313
- Enhancing Your Test on page 315
- Using Relative Paths in QuickTest on page 316

About Creating Tests

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

Creating tests using the keyword-driven methodology requires an infrastructure for all of the required resources. Resources include shared object repositories, function libraries, and recovery scenarios. Setting up the infrastructure requires in-depth knowledge of your application and a high level of QuickTest expertise.

Although setting up the infrastructure may initially require a longer time-investment in comparison to recording tests, using the keyword-driven methodology enables you to create tests at a more application-specific level and with a more structured design. This enables you to maintain your tests more efficiently and provides you with more flexibility than a recorded test.

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

While creating your test, you can insert checkpoints. A **checkpoint** compares the value of an element captured when the object was saved in the object repository, with the value of the same element captured during the run session. This helps you determine whether or not your application is functioning correctly. For more information, see “Understanding Checkpoints” on page 495.

When you test your application, you may want to check how it performs the same operations with different data. This is called **parameterizing** your test. You can supply data in the Data Table, define environment variables, instruct QuickTest to generate random numbers, and so on. For more information, see “Parameterizing Values” on page 625.

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

Deciding Which Methodology to Use - Keyword-Driven or Recording

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

Recording Tests

Recording can be useful in the following situations:

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

For information on recording tests, see “Creating Tests Using the Recording Mechanism” on page 361.

Creating Tests Using Keyword-Driven Testing

Keyword-driven testing advantages include the following:

- Keyword-driven testing enables you to design your tests at a business level rather than at the object level. For example, QuickTest may recognize a single option selection in your application as several steps: a click on a button object, a mouse operation on a list object, and then a keyboard operation on a list sub-item. You can create an appropriately-named function to represent all of these lower-level operations in a single, business-level keyword.
- By incorporating technical operations, such as a synchronization statement that waits for client-server communications to finish, into higher level keywords, tests are easier to read and easier for less technical application testers to maintain when the application changes.

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

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

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

For information on creating tests using the keyword-driven methodology, see “Creating Tests Using the Keyword-Driven Methodology” on page 335.







Understanding Your Test

When you create a test, QuickTest creates a graphical representation of the steps you perform on your application. These steps are displayed in the Keyword View tab.

The following is a sample test of a login procedure to the Mercury Tours site, the sample Web site.

Item	Operation	Value	Documentation
▼ Action1			
▼ Welcome: Mercury Tours			
▼ Welcome: Mercury Tours			
username	Set	"tutorial"	Enter "tutorial" in the "userName" ed
password	SetSecure	"477cdb71935682eda"	Enter the encrypted string "477cdb7
Sign-In	Click	30,12	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 "tutorial"	userName is the name of the edit box. Set is the method performed on the edit box. tutorial is the value of the edit box.
 password 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.

In the Expert View, these same steps are displayed using a VBScript program based on the QuickTest object model.

```

Browser("Welcome: Mercury Tours").Page("Welcome: Mercury Tours").
    WebEdit("userName").Set "tutorial"
Browser("Welcome: Mercury Tours").Page("Welcome: Mercury Tours").
    WebEdit("password").SetSecure
    "4082986e39ea469e70dbf8c5a29429fe138c6efc"
Browser("Welcome: Mercury Tours").Page("Welcome: Mercury Tours").
    Image("Sign-In").Click 2,2

```

Enhancing Your Test

You can use a variety of options to enhance your existing tests. This section describes some of the ways in which you can enhance your existing tests.

Checkpoints

You can add checkpoints to your test. A **checkpoint** is a step in your test that compares the a specified item during a run session with the values stored for the same item within the test. This enables you to identify whether or not your application is functioning correctly. There are several different checkpoint types. For more information on creating checkpoints, see Chapter 17, “Understanding Checkpoints.”

Tip: You can also use the `CheckProperty` method, which enables you to verify the property value of an object without using the checkpoint interface. For more information, see *HP QuickTest Professional Object Model Reference*.

Parameterization

You can parameterize your test to replace fixed values with values from an external source during your run session. The values can come from a Data Table, environment variables you define, or values that QuickTest generates during the run session. For more information, see Chapter 24, “Parameterizing Values.”

Output Values

You can retrieve values from your test and store them in the Data Table as output values. You can subsequently use these values as an input parameter in your test. This enables you to use data retrieved during a test in other parts of the test. For more information, see Chapter 25, “Outputting Values.”

Actions

You can divide your test into actions to streamline the testing process of your application. For more information, see Chapter 15, “Working with Actions.”

Programming Statements

You can use special QuickTest options to enhance your test with programming statements. The Step Generator guides you step-by-step through the process of adding recordable and non-recordable operations (methods and properties) to your test. You can also synchronize your test to ensure that your application is ready for QuickTest to perform the next step in your test, and you can measure the amount of time it takes for your application to perform steps in a test by defining and measuring transactions. For more information, see Chapter 28, “Adding Steps Containing Programming Logic.”

You can also manually enter standard VBScript statements, as well as statements using QuickTest test objects and operations, in the Expert View. For more information, see Chapter 29, “Working in the Expert View and Function Library Windows.”

Using Relative Paths in QuickTest

QuickTest enables you to define the path to a resource that you are adding to the file system or to Quality Center, as a relative or an absolute path. (For information about relative or absolute paths, see “Understanding Absolute and Relative Paths” on page 319.)

Note: If you are working with the Resources and Dependencies model with Quality Center 10.00, specify an absolute Quality Center path. For more information, see “Considerations for Working with Relative Paths in Quality Center” on page 1450.

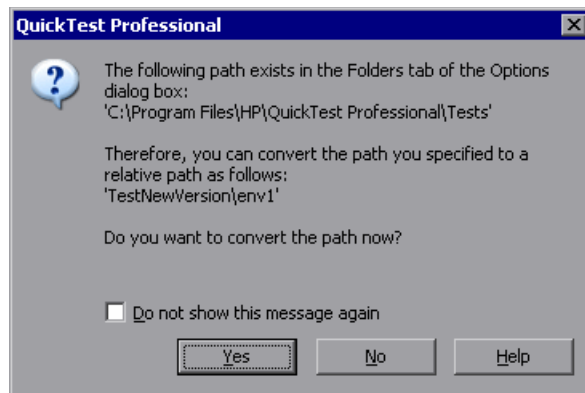
When you specify a path to a function library, shared object repository, recovery scenario, or environment variable file, QuickTest checks if the path, or the initial part of the path, exists in the Folders pane of the Options dialog box (**Tools > Options > Folders** node). The Folders pane contains a search list in which you can define where QuickTest searches for tests, actions, or files.

QuickTest then opens one of the following two dialog boxes, depending on whether the path you specified, or a part of the path, exists in the Folders pane.

Note: If you are connected to Quality Center 10.00, these dialog boxes are displayed only if you select a path in the file system or in a Quality Center 9.x project.

Path Exists in the Folders Pane

If the resource path you specify matches an existing search path in the Folders pane, you are prompted whether to define the path using only the relative part of the path.

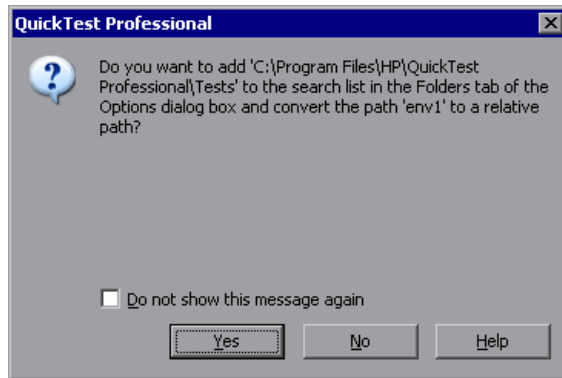


- Click **Yes** to truncate the path to a relative path.
- Click **No** to define the path to the resource as an absolute path.

In cases where a part of the path you enter matches more than one path in the Folders pane, the closest match is applied. For example, if both C:\Current_Version and C:\Current_Version\Libraries are defined in the search path list, the latter is applied.

Path Does Not Exist in the Folders Pane

If the resource path you specify does not match an existing search path in the Folders pane, you are prompted whether to add the resource's location path to the Folders pane and define the path relatively.



- Click **Yes** to add the resource's location path to the Folders pane and truncate the path to a relative path.
- Click **No** to define the path to the resource as an absolute path.

Notes:

- You can choose not to show one or both of these dialog boxes when you enter a path to a resource by selecting the **Do not show this message again** check box. To show these dialog boxes again, select the **Remind me to use relative paths when specifying a path to a resource** check box in the Folders pane of the Options dialog box. This check box is selected by default when you first start QuickTest.
 - For more information on the Folders pane, which enables you to enter the folders (search paths) in which QuickTest searches for searches for tests, actions, or files, see "Setting Folder Testing Options" on page 1237.
-

Understanding Absolute and Relative Paths

You can save QuickTest resources, such as shared object repositories, function libraries, recovery scenarios or environments, using absolute or relative paths.

Note: If you are working with the Resources and Dependencies model with Quality Center 10.00, specify an absolute Quality Center path. For more information, see “Considerations for Working with Relative Paths in Quality Center” on page 1450.

- An **absolute** path describes the full path to a specific file starting from a fixed location such as the root directory, or the drive on which the file is located, and contains all the other sub-directories in the path. An absolute path always points to the specified file, regardless of the current directory.
- A **relative** path describes the path to a specific file starting from a given directory, and is generally only a portion of the absolute path. A relative path therefore specifies the location of the file relative to the given location in the file system.

Using relative paths means that the paths remain valid when files or folders containing files are moved or copied to other locations or computers, provided that they are moved within the same folder structure. For this reason, we recommend that you use relative paths when saving resources in QuickTest.

For example, consider a QuickTest resource file named `FunctionLibrary1.qfl` located in `C:\Current_Version\Libraries`. The absolute path to the file is `C:\Current_Version\Libraries\FunctionLibrary1.qfl`. The relative path to the file from within the folder named `Libraries` is specified using only the name of the file, `FunctionLibrary1.qfl`. Alternatively, the relative path to the file from within another folder, such as `C:\Current_Version\Libraries\MyFiles`, would be `Libraries\FunctionLibrary1.qfl`.

Using a relative path, you could copy the `FunctionLibrary1.qfl` file from `C:\Current_Version\Libraries` to an updated version in `C:\New_Version\Libraries`, and the path used by QuickTest would remain valid.

In addition, relative paths are quicker to type and, being shorter, minimize any chance for error.

For more information, see “Using Relative Paths in QuickTest” on page 316.

Note: Prior to QuickTest 9.0, if you specified a path for a resource starting with `\..`, it was considered to be a relative path. In QuickTest 9.0 and later, a path that starts with `\..` is considered to be a full path, with the backslash representing the root folder of the current drive.

If you defined paths starting with `\..` using earlier versions of QuickTest, you should change the path to be a standard relative path by removing the backslash (`\`).

11

Managing Your Test

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

Tip: As the content of your application changes, you can update the selected Active Screen display and use the Active Screen to add new steps to your test instead of re-recording steps on new or modified objects. For more information, see "Updating a Test Using the Update Run Mode Option" on page 1125.

This chapter includes:

- Creating a New Test on page 321
- Opening an Existing Test on page 322
- Saving a Test on page 324
- Creating Portable Copies of Your Tests on page 326
- Zipping a Test on page 331
- Unzipping a Test on page 331
- Printing a Test on page 332

Creating a New Test

To create a new test, click the **New** button or select **File > New > Test**. A new test opens, with a new action selected in the Keyword View. You are ready to start creating your test.

Opening an Existing Test

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

To open a test stored in Quality Center, QuickTest must be connected to the Quality Center project. For more information, see Chapter 51, "Integrating with Quality Center."

If the test you select was last saved in an older version of QuickTest, you may be asked whether to convert the test to the current version or view it in read-only format. For more information, see "Considerations for Opening Tests Created in Previous Versions of QuickTest" on page 323.

To open an existing test:

- 1** (Optional) Connect to a Quality Center server and project. For more information, see "Connecting to and Disconnecting from Quality Center" on page 1418.
- 2** Select **File > Open > Test**, or click the **Open** down arrow and select **Test**. The Open Test dialog box opens.
- 3** In the sidebar, select the location of the test, for example, File System or Quality Center Test Plan.
- 4** Browse to and select a test. You can select the **Open in read-only mode** option at the bottom of the dialog box. Click **Open**. The test opens and the title bar displays the test name.

Note: If the test is stored in a version control-enabled Quality Center project, the **Open** button contains a down arrow, enabling you to open the test and immediately check it out. For more information, see "Checking Assets Out of the Version Control Database" on page 1483.

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

Considerations for Opening Tests Created in Previous Versions of QuickTest

- If a test is stored in Quality Center and was created using an earlier version, it opens in read-only mode. To edit the test, it must be upgraded to the current version using the QuickTest Professional Asset Upgrade Tool for Quality Center. You install this tool from the QuickTest Professional installation DVD. After installation, this tool is available from the **Start** menu by choosing **Start > Programs > QuickTest Professional > Tools > QuickTest Professional Asset Upgrade Tool**.
- When you open a test that was created using an older version of QuickTest, you may be asked whether you want to convert it or view it in read-only format.
 - If the test contains objects in the local object repositories of one or more actions in the test, the relevant add-in must be installed to convert the test to the current format. Otherwise, it is opened in read-only format.
 - If you choose to convert the test, it is updated to the current format and you can modify it as needed. If you save the converted test, it cannot be used with earlier versions of QuickTest.
 - If you choose to view the test in read-only format, it appears as it did previously, using all of its original settings, but you cannot modify it.
 - If you have many tests that need to be updated to the current format, you can create an automation script that iterates through all of your tests to open and save each one in the new format.

For more information on creating automation scripts, see Chapter 50, "Automating QuickTest Operations."

To view a sample automation script that converts old tests to the current version, see the *QuickTest Professional Automation Object Model Reference* (**Help > QuickTest Professional Help > HP QuickTest Professional Advanced References > HP QuickTest Professional Automation Object Model**).

- You cannot open a test that was created with a later version of QuickTest on a computer running an earlier version of QuickTest. For example, you cannot open a test created in QuickTest 10.00 on a computer running QuickTest 8.0.


Saving a Test

You can save a new test or save changes to an existing test.

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

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

To save a new test:

- 1 (Optional) Connect to a Quality Center server and project. For more information, see "Connecting to and Disconnecting from Quality Center" on page 1418.
- 2  Click the **Save** button or select **File > Save** to save the test. The Save QuickTest Test dialog box opens.
- 3 In the sidebar, select the location to save the test, for example, File System or Quality Center Test Plan.
- 4 Browse to and choose the folder in which to save the test.

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

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

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

- 6** If you are recording and want to save the Active Screen files with your test, confirm that **Save Active Screen files** is selected.

If you clear this box, your Active Screen files will not be saved, and you will not be able to edit your test using the options that are normally available from the Active Screen.

Clearing the **Save Active Screen files** check box can be especially useful for conserving disk space once you have finished designing the test and you are using the test only for test runs.

Tip: If you clear the Save Active Screen files check box and then later want to edit your test using Active Screen options, you can regenerate the Active Screen information by performing an **Update Run** operation. For more information, see "Updating a Test Using the Update Run Mode Option" on page 1125.

Note: You can also instruct QuickTest not to capture Active Screen files while recording or to only capture Active Screen information under certain conditions. You can set these preferences in the Active Screen pane of the Options dialog box. For more information, see "Setting Active Screen Options" on page 1240.

- 7** Click **Save**. QuickTest displays the test name in the title bar.

To save changes to an existing test:



- Click the **Save** button to save changes to the current test.
- Select **File > Save As** to save an existing test to a new name or a new location. If you select **File > Save As**, the following options are available:
 - Select or clear the **Save Active Screen files** check box to indicate whether or not you want to save the Active Screen files with the new test. For more information, see step 6 above.
 - Select or clear the **Save test results** check box to indicate whether or not you want to save any existing test results with your test.

Note that if you clear this box, your test result files will not be saved, and you will not be able to view them later. Clearing the **Save test results** check box can be useful for conserving disk space if you do not require the test results for later analysis, or if you are saving an existing test under a new name and do not need the test results.

Creating Portable Copies of Your Tests

Tests and their resource files are often stored on a network drive or in Quality Center, as this enables the reuse of actions and other resources, and helps ease test management.

Sometimes, you may need to open or run a test when you do not have access to a network drive or Quality Center. For example, you may need to create a portable copy of a test for use when travelling to other sites. You can save a standalone copy of your test and its resource files to a local drive or to another storage device using the **File > Save Test with Resources** command.


When you save a test in this manner, QuickTest creates a copy of the following and saves the files in the location you specify:

- **Source test.** QuickTest saves a copy of this test in the location you specify.
- **Resource files.** QuickTest saves a copy of all resource files associated with the source test, such as function libraries and shared object repositories. QuickTest stores these files in sub-folders of the copied test.
- **Called actions.** QuickTest saves a copy of any external actions called by the source test. For example, if Test A calls actions that are stored in Test B, QuickTest creates a local copy of the actions stored in Test B and stores them in a sub-folder of Test A. The sub-folder has the same name as the test from which the called actions were copied. In this example, the sub-folder is named Test_B. QuickTest also creates a copy of any resources associated directly with these actions, such as its local shared object repositories and action sheets in the Data Table. QuickTest does not, however, save the resource files associated with Test B, so you must ensure that these resources are associated with the source test, Test A.

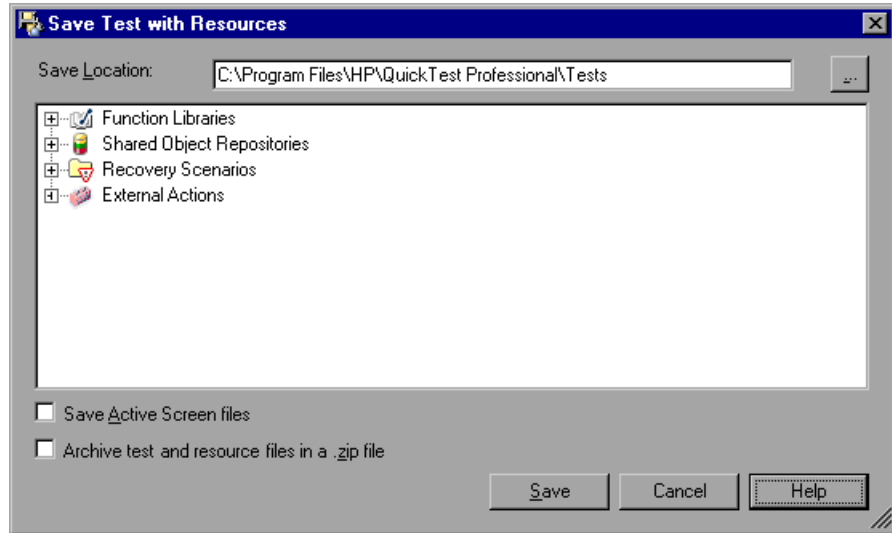
This enables you to modify or run the test without access to a network drive or Quality Center.

Tip: If you use QuickTest with a concurrent license but do not have access to the concurrent license server (for example, during a business trip), you can install a commuter license. For more information, see the *HP QuickTest Professional Installation Guide*.

The Save Test with Resources Dialog Box

Description	Enables you to save a full copy of a test and its resource files to a local drive or other storage device, eliminating the need for network or Quality Center connections.
How to Access	<ul style="list-style-type: none"> ➤ Select the File > Save Test with Resources menu command. ➤ Click the Save Test with Resources toolbar button .
Important Information	<p>Before you create a copy of the test:</p> <ul style="list-style-type: none"> ➤ Resolve any missing resources. ➤ Save the original test. ➤ Make sure that all files associated with the source test are writable. ➤ Make sure you have write permissions for the folder in which you want to create a copy of the test. <p>After you make a copy of the test:</p> <ul style="list-style-type: none"> ➤ A report is displayed in HTML format, listing: <ul style="list-style-type: none"> ➤ the name of the test, the name of the user that saved this copy of the test, and the date on which the test was copied. ➤ a record for each resource that was copied with the test, specifying: <ul style="list-style-type: none"> -- the name of the resource -- the type of resource (for example, function library) -- the path from which the resource was copied -- the status of the copied resource (for example, the resource was saved successfully) -- the current location of the copied resource <p>You can also open this file from the copied test's root folder.</p> ➤ The copied test becomes the active test in the QuickTest window. ➤ All links to the source files are severed. Therefore, any modifications you make to the copied test are applied only to the copied test.
Learn More	<p>Conceptual overview: "Creating Portable Copies of Your Tests" on page 326</p> <p>Additional related topic: "Guidelines for Working with Tests Created Using an Earlier Version of QuickTest" on page 330</p>

Below is an image of the Save Test with Resources dialog box:



Save Test with Resources Dialog Box Options

Option	Description
Save Location	Specifies the root folder in which to save the test. By default, the root folder is <QuickTest installation folder>\Tests , however, you can specify any folder on a local, network, or portable drive. Important: The folder you specify must not already contain a sub-folder with the same name as the test.
Resources tree	Lists the external resources that are currently associated with or attached to your test.

Option	Description
Save Active Screen files	<p>(Relevant only for recorded tests.) Instructs QuickTest to save any existing Active Screen files with your test.</p> <p>Clearing the Save Active Screen files check box can be especially useful for conserving disk space if you have finished designing the test and are using the test only for test runs.</p> <p>Note: If you clear this box, your Active Screen files will not be copied over with the test and its resources, and you will not be able to edit your test using the options that are normally available from the Active Screen.</p> <p>Tip: If you clear the Save Active Screen files check box and then later want to edit your test using Active Screen options, you can regenerate the Active Screen information by performing an Update Run operation. For more information, see "Updating a Test Using the Update Run Mode Option" on page 1125.</p>
Archive test and resource files in a .zip file	<p>Creates a .zip file of the test and its resources, and stores the .zip file in the folder you specified in the Save Location box.</p> <p>For more information, see "Zipping a Test" on page 331.</p>

Guidelines for Working with Tests Created Using an Earlier Version of QuickTest

Before you can save a standalone copy of a test that was created in an earlier version of QuickTest, you must upgrade the test and its resource files to the current version of QuickTest, as follows:

- Open the test in QuickTest and save it (**Save** or **Save As**). If the test contains calls to external actions (actions stored in other tests), you must open and save those tests, too.
- Alternatively, if your tests are stored in Quality Center, you can use the QuickTest Professional Asset Upgrade Tool for Quality Center. This converts your test's attached resource files to linked assets and upgrades your tests to the current version of QuickTest.

Zippping a Test

QuickTest tests contain a series of configuration, run-time, setup data, and (optionally) Active Screen files. QuickTest saves these files together with the test. You can zip these files to conserve space and make the tests easier to transfer.

To zip a test:

- 1** Do one of the following:
 - Select **File > Export Test to Zip File** to open the Export to Zip File dialog box.
 - Select the **Archive test and resource files in a .zip file** check box in the Save Test with Resources dialog box (**File > Save Test with Resources**). For more information, see "The Save Test with Resources Dialog Box" on page 328.
- 2** Type a zip file name and path, or accept the default name and path, and click **OK**. QuickTest zips the test and its associated files.

Unzipping a Test

You can unzip a test when needed.

To unzip a zipped test:

- 1** Select **File > Import Test from Zip File**. The Import from Zip File dialog box opens.
- 2** Enter or select the name of the zip file that you want to unzip, choose a target folder into which you want to unzip the files, and click **OK**. QuickTest unzips the test and its associated files.

Printing a Test

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

To print from the Keyword View:



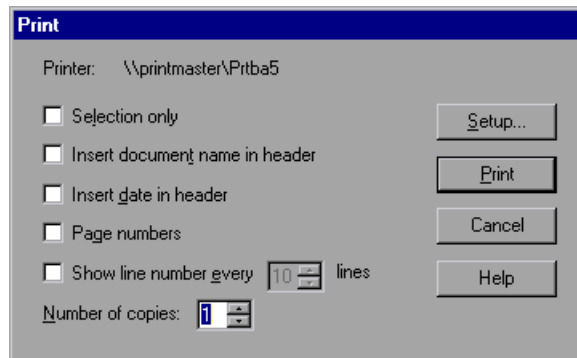
- 1 Click the **Print** button or select **File > Print**. A standard Print dialog box opens.
- 2 Click **OK** to print the content of the Keyword View to your default Windows printer.

Tip: You can select **File > Print Preview** to display the Keyword View on screen as it will look when printed. Note that the **Print Preview** option works only with tests created using QuickTest 8.0 and later.

To print from the Expert View:



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



2 Specify the print options that you want to use:

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

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

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

12

Creating Tests Using the Keyword-Driven Methodology

You can create a test using the keyword-driven methodology, which enables you to select keywords to indicate the operations you want to perform on your application. This enables you to create structured tests that are easier to update and maintain over time.

The keyword-driven methodology is especially useful for organizations that have both technical and less technical users because it offers a clear division of automation tasks. This enables a few experts to maintain the resource framework while less technical users design and maintain automated test steps. Additionally, once the basic infrastructure is in place, both types of users can often do their tasks simultaneously.

Before you begin creating tests, you need to plan your tests to ensure that your tests cover your testing requirements. For more information on planning your tests, see “Creating Tests — Overview” on page 309.

After you create your test, you can enhance it using checkpoints and other special testing options.

Tip: You can also create a test by recording the operations you perform on your application, as described in “Creating Tests Using the Recording Mechanism” on page 361. After you create your test, you can enhance it using checkpoints and other special testing options.

This chapter includes:

- Understanding the Keyword-Driven Methodology on page 336
- Using the Keyword-Driven Methodology on page 341
- Sample Implementation of the Keyword-Driven Methodology on page 351

Understanding the Keyword-Driven Methodology

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

This section provides a general overview of the steps you perform when planning and implementing your tests.

Stage 1: Analyzing Your Application

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

First, determine the development environments in which your application controls were developed, such as Web, Java, or .NET, so that you can load the required QuickTest add-ins.

Then determine the functionality that you want to test. To do this, consider the various activities that customers perform in your application to accomplish specific tasks. Which objects and operations are relevant for the set of business processes that need to be tested? Which operations require customized keywords to provide additional functionality?

While you are thinking about the business processes you want to test, consider how you can divide these processes into smaller units, which will be represented by your test's actions. Each action should emulate an activity that a customer might perform when using your application.

As you plan, try to keep the amount of steps you plan to include in each action to a minimum. Creating small, modular actions helps make your tests easier to read, follow, and maintain.

Stage 2: Preparing the Testing Infrastructure

To complete the infrastructure that is part of the planning process, you need to build the set of resources to be used by your tests, including shared object repositories containing test objects (which are representations of the objects in your application), function libraries containing functions that enhance QuickTest functionality, and so on. For more information, see Chapter 5, “Managing Test Objects in Object Repositories” and Chapter 31, “Working with User-Defined Functions and Function Libraries.”

At this stage you also need to configure QuickTest according to your testing needs. This can include setting up your global testing preferences, your run session preferences, any test-specific preferences, and recovery scenarios. You can also create automation scripts that automatically set the required configurations (such as the add-ins to load) on the QuickTest client at the beginning of a run session. For more information, see Chapter 50, “Automating QuickTest Operations.”

Lastly, you create one or more tests that serve as action repositories in which you can store the actions to be used in your tests. Generally, you create an action repository test for each area of your application to be tested. Storing all of your actions in specific tests enables you to maintain your actions in a central location. When you update an action in the action repository, the update is reflected in all tests that contain a call to that action. When you run a test, only the relevant action repository tests are loaded.

You then associate the shared object repositories with the relevant actions. This enables you to later insert steps using the objects stored in the object repositories.

When you create your tests, you insert calls to one or more of the actions stored in this repository.

Stage 3: Adding Steps to Your Actions

In this stage, you add steps to the actions in your test action repository.

Before you begin adding steps, make sure that you associate your function libraries and recovery scenarios with the relevant tests, so that you can insert steps using keywords.

You can create steps using the keyword-driven functionality available in the table-like, graphical Keyword View—or you can use the Expert View, if you prefer to program steps directly in VBScript. You can add steps to your test in one or both of the following ways:

- Drag objects from your object repository or from the Available Keywords pane to add keyword-driven steps in the Keyword View or Expert View. The object repository and Available Keywords pane contain all of the objects that you want to test in your application. (You create one or more object repositories when you prepare the testing infrastructure, as described in “Stage 2: Preparing the Testing Infrastructure” on page 337.)

When you drag an object into the Keyword View, a step is created in the action with the default operation for that object. For example, if you drag a button object into the Keyword View, the click operation is automatically defined for the step. You can then modify the step as needed. For more information, see Chapter 14, “Working with the Keyword View” and Chapter 39, “Adding Keywords to Your Test.” Advanced users can also add steps using the Expert View. For more information, see Chapter 29, “Working in the Expert View and Function Library Windows.”

- Record on your application.

As you navigate through your application during a recording session, QuickTest graphically displays each step you perform as a row in the Keyword View. A step is something that causes or makes a change in your application, such as clicking a link or image, or submitting a data form. In the Expert View, these steps are displayed as lines in a test script (VBScript). The **Documentation** column of the Keyword View also displays a description of each step in easy-to-understand sentences. For more information, see Chapter 14, “Working with the Keyword View.”

Stage 4: Enhancing Your Test

You can enhance the testing process by modifying your test with special testing options and/or with programming statements, such as:

- Insert checkpoints and output values into your test.

A **checkpoint** checks specific properties or other characteristics of an object and enables you to identify whether or not your application is functioning correctly. For more information, see Chapter 17, “Understanding Checkpoints.”

You can also use output values to extract data from your test. An **output value** is a value retrieved during the run session and entered into your Data Table or stored in a variable or a parameter. You can subsequently use this output value as input data in your test. This enables you to use data retrieved during a run session in other parts of the test. For more information, see Chapter 25, “Outputting Values.”

- Broaden the scope of your test by replacing fixed values with parameters.

When you test your application, you can parameterize your steps to check how your application performs the same operations with different data. You may supply data in the Data Table, define environment variables and values, define test or action parameters and values, or instruct QuickTest to generate random numbers for current user and test data.

When you parameterize your test, QuickTest substitutes the fixed values in your test with the values stored in the relevant parameters. When you use Data Table parameters, QuickTest uses the values from a different row in the Data Table for each iteration of the test or action. (Each run session that uses a different set of parameterized data is called an iteration.) For more information, see Chapter 24, “Parameterizing Values.”

- Add user-defined functions by creating function libraries and calling their functions from your test. For more information, see Chapter 31, “Working with User-Defined Functions and Function Libraries.”
- Use the many functional testing features included in QuickTest to enhance your test and/or add programming statements to achieve more complex testing goals. For more information, see Chapter 28, “Adding Steps Containing Programming Logic.”

Stage 5: Running and Debugging Your Test

After you create your test, you can perform different types of runs to achieve different goals.

- **Run your test to debug it.** You can control your run session to help you identify and eliminate defects in your test. You can use the **Step Into**, **Step Over**, and **Step Out** commands to run your test step by step. You can begin your run session from a specific step in your test, or run the test until a specific step is reached. You can also set breakpoints to pause your test at predetermined points. You can view or change the value of variables in your test each time it stops at a breakpoint in the Debug Viewer. You can also manually run VBScript commands in the Debug Viewer. For more information, see Chapter 35, “Debugging Tests and Function Libraries.”
- **Run your test to check your application.** The test starts running from the first line in your test and stops at the end of the test. While running, QuickTest connects to your application and performs each operation in your test, including any checkpoints, such as checking any text strings, objects, tables, and so forth. If you parameterized your test with Data Table parameters, QuickTest repeats the test (or specific actions in your test) for each set of data values in the Data Table. For more information, see Chapter 32, “Running Tests.”
- **Run your test to update it.**
 - You can run your test using **Maintenance Run Mode** when you know that your application has changed, and you therefore expect that QuickTest will not be able to identify the objects in your test. When you run a test in Maintenance Run Mode, a wizard opens for steps that fail because an object could not be found in the application. The wizard then guides you through the steps of resolving the issue, and, after you resolve the issue, the run continues. For more information, see Chapter 36, “Maintaining Tests.”
 - You can run your test using **Update Run Mode** to update the property sets used for test object descriptions, the expected checkpoint values, the data available to retrieve in output values, and/or the Active Screen images and values.

Stage 6: Analyzing Test Results and Reporting Defects

After you run your test, you can view the results of the run in the Test Results window. You can view a summary of your results as well as a detailed report. If you captured still images or movies of your application during the run, you can view these from the Screen Recorder tab of the Test Results window. For more information, see Chapter 33, “Viewing Run Session Results.” If you enabled local system monitoring for your test, you can view the results in the System Monitor tab of the Test Results window. For more information, see “Viewing System Monitor Results” on page 1063.

Finally, you can report defects detected during a run session. If you have access to Quality Center, the HP centralized quality solution, you can report the defects you discover to the project database. You can instruct QuickTest to automatically report each failed step in your test, or you can report them manually from the Test Results window. For more information, see Chapter 51, “Integrating with Quality Center.”

Using the Keyword-Driven Methodology

By creating your tests with a keyword-driven methodology in mind, your tests become more modular, focusing on the operations to test using both QuickTest built-in keywords and your own user-defined keywords. Additionally, because it is possible to add objects to the object repository before they exist in an application, it is possible to begin preparing your automated keyword-driven tests even before a software build containing the new objects is available.

One or a few automation experts usually develop the test automation infrastructure that all tests related to a certain application or functionality can use. The automation infrastructure usually includes one or more shared object repositories and one or more function libraries.

The information in the sections below provides guidance on the main tasks involved in creating these resources and describes where you can find detailed documentation for these tasks.

Analyzing Your Application

In this step, you analyze your application to determine your testing needs. This step is divided into multiple tasks:

► **Determine the development environments that QuickTest needs to support.**

From the perspective of QuickTest, your application comprises windows containing a hierarchy of objects that were created in one or more development environments. QuickTest provides support for these environments using add-ins.

You load QuickTest add-ins when QuickTest opens by using the Add-in Manager dialog box. You can check which add-ins are loaded by choosing **Help > About QuickTest Professional**. For more information, see the *HP QuickTest Professional Add-ins Guide*.

► **Prepare the information that QuickTest needs to identify objects in your application and to (optionally) open your application at the beginning of a run session.** You need to know the URL, the executable file name and path, or other command-line information. Later, you will enter this in Record and Run Settings dialog box. For more information, see the sections describing the Record and Run options for your testing environment in the *HP QuickTest Professional Add-ins Guide*.

► **Analyze the various business processes that customers perform while using your application to determine the actions you need to create.** You create an action for each sub-process, or task, a customer might perform.

Navigate through your application from a customer's perspective and perform the tasks that customers might perform. Each process you perform in your application will be represented as a test in QuickTest. You can create your tests now, or you can wait until you are ready to add steps to your tests

As you perform a process, try to compartmentalize or "chunk" it into modular units.

Example

An application that enables users to purchase items online might contain various business processes, including registering on the site and purchasing items. Each process may require one or more tasks—you create actions based on these tasks. For example, registering on the site may be a simple process requiring only one action, whereas purchasing items may be more complex, requiring several actions, such as a Login action, a Browse action, an AddToCart action, a PurchaseItems action, and a Logout action.

By creating separate reusable actions for each task, you can include calls to the same actions from multiple tests. For example, you may want to include a Login action in many of your tests.

You can create empty actions now to set up a skeleton infrastructure for your tests, or you can create them when you are ready to add steps to your actions. For more information, see “Working with Actions” on page 425.

You may also want to create a single test storing all actions relevant for an application. Then all other tests can call the actions stored in this central repository. This helps with test structure and maintenance.

Tip: As you plan your tests and actions, keep in mind that short tests and actions that check specific functions of the application or complete a transaction are better than long ones that perform several tasks.

Setting Up Object Repositories

In this step, you build one or more object repositories and ensure that all objects have clear names that follow any predetermined naming conventions defined by your organization.

You can create object repositories using QuickTest functionality to recognize and learn the objects in your application, or you can manually define objects. The object repository should contain all the objects that are relevant for the tests using this infrastructure.

By creating and populating shared object repositories that can be associated with multiple actions, you can use the same object repository in multiple tests. By maintaining all objects that are relevant to an area of an application within one shared object repository, and by associating that object repository with all relevant actions, changes to the application can be reflected in the object repository without the need to update tests.

Before you create a new object repository, verify whether an object repository containing the objects you are testing already exists. If not, you can create a new object repository or add objects to an existing one.

Creating shared object repositories for the test automation infrastructure can include the following tasks:

- **Change the way that QuickTest identifies specific objects, if needed.** This is particularly helpful when your application contains objects that change frequently or are created using dynamic content, for example, from a database. This task needs to be done before you create your object repository. For more information, see “Configuring Object Identification” on page 105.
- **Decide how you want to organize your object repositories.** For individual tests, you can work with the individual action’s object repositories, or you can work with a common (shared) object repository that can be used with multiple tests. If you are new to testing, you may want to keep the default object repository per-action setting for tests. As you feel more comfortable with the basics of test design, you may want to take advantage of the shared object repository option.

If you decide to work with shared object repositories, you need to determine how many shared object repository files are required for your application. You also need to determine which shared object repository will be used for each area of your application.

For more information, see “Managing Test Objects in Object Repositories” on page 135.

- **Add (learn) objects from your application.** You instruct QuickTest to learn the objects in your application according to filters that you define. For more information, see “Adding Test Objects to a Local or Shared Object Repository” on page 136.
- **If necessary, create new objects to represent objects that do not yet exist in your application.** Then update the properties and values of these objects as necessary after they exist in the application. For more information, see “Defining New Test Objects” on page 147.
- **Ensure that objects in the object repository have names that are easy for application testers to recognize and that follow any established object naming guidelines.** This helps make both test creation and maintenance easier over time.
- **Copy or move objects from one repository to another, as needed.** For more information, see Chapter 7, “Managing Object Repositories.”
- **Merge objects added to local repositories by application testers into the shared object repositories of the automation infrastructure.** You can also merge two or more existing repositories. For more information, see Chapter 8, “Merging Shared Object Repositories.”

Creating Function Libraries

Creating function libraries involves developing customized functions for the application you want to test. You may want to develop functions to test special application functionality that is not already supplied by the methods in the QuickTest object model. This enables you to create keywords that perform operations that are not normally available for use with a particular test object class. For example, you may need to add a worksheet to an Excel file, or to generate a text file during a run session.

It may also be useful to wrap existing methods and functions together with additional programming to create application-specific functions for testing operations or sequences that are commonly performed in your application. The functions you create will be available either as extra keywords or as replacements for built-in QuickTest keywords during the test creation stage.

By encapsulating much of the complex programming into function libraries, and by making these functions flexible enough to use in many testing scenarios (through the use of function parameters that control the way the functions behave), one or a few automation experts can prepare the keywords that many application testers (who are less technical) can include in multiple tests. This also makes it possible to update testing functionality without having to update all the tests that use the keywords.

You may perform the following tasks when creating a function library for the test automation infrastructure:

- **Determine whether you need to create any user-defined functions or whether you should associate any existing function libraries with your test.**
- **Determine which keywords are needed.**
- **Develop and document business-level keywords in function libraries using the QuickTest Function Library window.** For more information, see “Working with User-Defined Functions and Function Libraries” on page 905 and “Creating a Function Library” on page 909.
- **Create the actual functions within the function libraries.** You can do this manually, or you can use the Function Definition Generator to generate function definitions and header information. For more information, see “Using the Function Definition Generator” on page 923.
- **Optionally define functions as new or replacement methods for test objects.** For more information, see “Registering User-Defined Functions as Test Object Methods” on page 939.
- **Debug the function libraries.** For more information, see “Debugging a Function Library” on page 916.

Configuring QuickTest According to Your Testing Needs

After you set up the test automation infrastructure, you need to configure QuickTest to use this infrastructure:

- **Define your global testing preferences.** You need to specify configuration settings that affect how you create and run tests in general—these settings are not test-specific. For example, you can instruct QuickTest to record a movie of the run session under certain conditions, and to enable other HP products to run QuickTest tests (for example, if you want to run your tests from Quality Center).

You can set global testing options using the Options dialog box (**Tools > Options**) or by inserting statements in the Expert View. For more information, see “About Setting Global Testing Options” on page 1231.

- **Determine whether you need to create any recovery scenarios, and create them, if needed.** Although not directly associated with the keyword-driven methodology, the automation experts who maintain the object repositories and function libraries also often maintain a set of recovery scenarios that all application testers can associate with their tests. Recovery scenarios instruct QuickTest how to proceed when a step fails. For more information, see “Defining and Using Recovery Scenarios” on page 1329.
- **Configure the QuickTest IDE to suit your testing preferences.** This enables you to easily access any needed panes, such as the Test Flow pane, the Resources pane, the Available Keywords pane, or the Data Table. For more information, see “QuickTest Window Layout” on page 1135.

Building Your Tests

You can create tests that are as simple or complex as needed. In general, it is best to create tests and actions that check just one or a few simple functions or complete a transaction rather than creating long tests and actions that perform several complex tasks or that perform many tasks.

You may perform the following tasks when creating tests and test steps:

- **Create new tests, if needed.** To do so, select **File > New > Test**.
- **Create the required actions.** For more information, see “Analyzing Your Application” on page 342.
- **Insert calls to the relevant actions.** For example, if the first task performed in a test logs in to the application, and you already created a Login action, insert a call to that action to include it in your test. For more information, see “Inserting Calls to Existing Actions” on page 464.
- **Associate your object repositories with the relevant actions.** This enables you to insert steps that perform operations on those objects. For more information, see “Associating Object Repositories with Actions” on page 446.
- **Associate your function libraries with the relevant tests.** This enables you to use your special keywords in any of the associated tests. For more information, see “Associating a Function Library with a Test” on page 921.
- **Optionally associate recovery scenarios with your test.** For more information, see “Associating Recovery Scenarios with Your Tests” on page 1372.

Adding Steps to Your Test Actions

When your actions are ready, you can add steps to them.

- **Add steps by selecting the keywords (operations) that represent the application functionality you want to test.** For more information, see “Working with the Keyword View” on page 383.

You can insert steps in the Keyword View, the Expert View, or a combination of both. You can add steps by dragging test objects from the Available Keywords pane, using the **New Step** option, using the Step Generator, entering steps manually, and so on. Make sure to fill in any missing values, as needed.

For more information, see “Adding a Standard Step to Your Test” on page 392, “Adding Other Types of Steps to Your Test” on page 407, and “Generating Statements in the Expert View or in a Function Library” on page 833.

- **Consider enhancing your tests by inserting checkpoint and output value steps to verify that your application is behaving as expected during a run session.**

You can insert checkpoints to check for differences in the text strings, objects, and tables in your application. For more information, see “Understanding Checkpoints” on page 495.

You can insert output value steps that retrieve values in your test and store them for use as input values at a different stage in the run session. For more information, see “Outputting Values” on page 669.

- **Consider data-driving your test to check how your application behaves with different data input during subsequent run sessions.** You can also data-drive your test to check how your application behaves during multiple iterations of the same action during a single run session. For more information, see “Working with Data Tables” on page 1197.
- **Consider increasing the power and flexibility of your test by replacing fixed values with parameters, if applicable.** When you parameterize your test, you can check how it performs the same operations with multiple sets of data, or from data stored or generated by an external source. For more information, see “Parameterizing Values” on page 625.

Note: If you have useful WinRunner assets, you may want to link to WinRunner tests and call WinRunner TSL functions from your QuickTest test. For more information, see “Working with WinRunner” on page 1517.

Running and Troubleshooting Your Tests

When your tests are ready, you run them, view the run results, and troubleshoot your tests, as needed.

- **Before you run a test, ensure that all of the required settings are configured as needed and that the required QuickTest add-ins are loaded.** Make sure that your application is open to the appropriate location for the beginning of the test, or that you instructed QuickTest to open it for you. Additionally, make sure that the Test Settings dialog box (**File > Settings**) and Record and Run Settings dialog box (**Automation > Record and Run Settings**) are configured for your test. For more information, see “Running Tests” on page 953.
- **After your test runs, view the run results.** Expand the nodes in the Test Results window to see where steps failed and to try to understand why. For more information, see “Viewing Run Session Results” on page 969.
- **Troubleshoot your test so that it runs correctly.** For example, you may need to add or modify test steps. For more information, see “Maintaining Tests” on page 1101.

Sample Implementation of the Keyword-Driven Methodology

As you have seen above, the process of creating a test is actually comprised of several steps.

This section walks you through the activities you might perform for each of these steps, if you were preparing a test suite for the Mercury Tours application, including:

- Define the Testing Environment for the Mercury Tours Application
- Analyze the Mercury Tours Application
- Plan and Create the Mercury Tours Test Action Repository
- Set Up the Object Repositories for the Mercury Tours Application
- Create the Function Libraries and Functions Required for Testing the Mercury Tours Application
- Create Tests and Test Steps for the Mercury Tours Business Processes

Mercury Tours is a Web-based demo application that simulates an online flight reservation application. You can view and experiment with this demo application at <http://newtours.demoaut.com>.

Define the Testing Environment for the Mercury Tours Application

Defining the testing environment includes determining which add-ins to load and the data required to activate the application.

Mercury Tours is a Web application that contains a few Java applets. Therefore, we need to ensure that the QuickTest Web and Java Add-ins are installed and loaded.

To activate the application, we need to run a URL in a Web browser. The URL is <http://newtours.demoaut.com>.

Analyze the Mercury Tours Application

When analyzing the application to determine which business processes we may want to test, we can consider both the existing business processes in the application as well as functionality that is planned for the upcoming release of the application.

The business processes that should be tested for the Mercury Tours application include:

- Registering on the site
- Reserving a flight
- Viewing the itinerary of a pending reservation
- Cancelling a reservation
- Updating user profile information
- *Reserving hotel rooms*
- *Renting a car*

Although the last two items above have not yet been implemented in the application we want to test, it is important to take them into account in the planning stage.

Now that we have determined the primary business processes, we should analyze each one to determine the break-down of these business processes into their reusable building-block elements (what will later become the test actions of our tests).

A logical breakdown of the above business processes could be:

- **Registering on the site**
 - Open the application
 - Go to the registration page
 - Enter the required information in the form
 - Submit the form
 - Verify that the form information is valid

- If a mandatory field did not have a value, an error message is displayed.
 - If the password and confirm password values are not the same, an error message is displayed.
 - If the username entered in the form already exists in the database, an error message is displayed.
 - Otherwise, the successful registration page is displayed.
- **Reserving a flight**
- Open the application
 - Sign on
 - Navigate to the Flight Finder page
 - Enter the flight details
 - Enter the service class and airline preferences
 - Click Next to navigate to the next page
 - Select the departure and return flights
 - Click Next to navigate to the next page
 - Enter the passenger details
 - Verify that the form information is valid
 - If the return date is earlier than the departure date, an error message is displayed.
 - If a mandatory field was not entered, an error message is displayed.
 - Otherwise, the flight confirmation page is displayed.

- **Viewing the itinerary of a pending reservation**
 - Open the application
 - Sign on
 - Navigate to the Itinerary page
- **Cancelling a reservation**
 - Open the application
 - Sign on
 - Navigate to the Itinerary page
 - Select the reservation to cancel
 - Click the **Cancel Checked Reservations** button
 - Verify
 - Successful cancellation
- **Updating user profile information**
 - Open the application
 - Sign on
 - ...

And so on for each of the remaining processes.

Comparing the sub-items in each of the business-processes helps to identify the reusable elements of each business process.

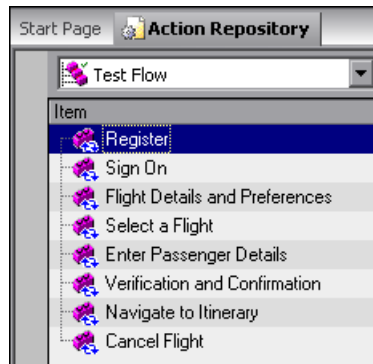
Plan and Create the Mercury Tours Test Action Repository

By analyzing the breakdown performed in the previous step, we are able to identify some logical, and reusable sub-processes. Each of these is created as a reusable action.

The required actions for the set of business processes we defined could include:

- Register
- Sign On
- Flight Details and Preference
- Select a Flight
- Enter Passenger Details
- Verification and Confirmation
- Navigate to Itinerary
- Cancel Flight

Although we are not yet ready to create the actual tests or steps yet, we can go ahead and create a single test. In the test, we can already define empty test actions for each of these. This test then acts as the **action repository**, and the tests that test each of our business processes all call actions from this action repository test.



Set Up the Object Repositories for the Mercury Tours Application

Now that we know which business processes and sub-processes we want to test, we can analyze the application in detail to determine which objects are important to test and how we want to organize the objects we will learn for these tests.

We know that it is best to create manageable-sized object repositories that are organized by areas of the application.

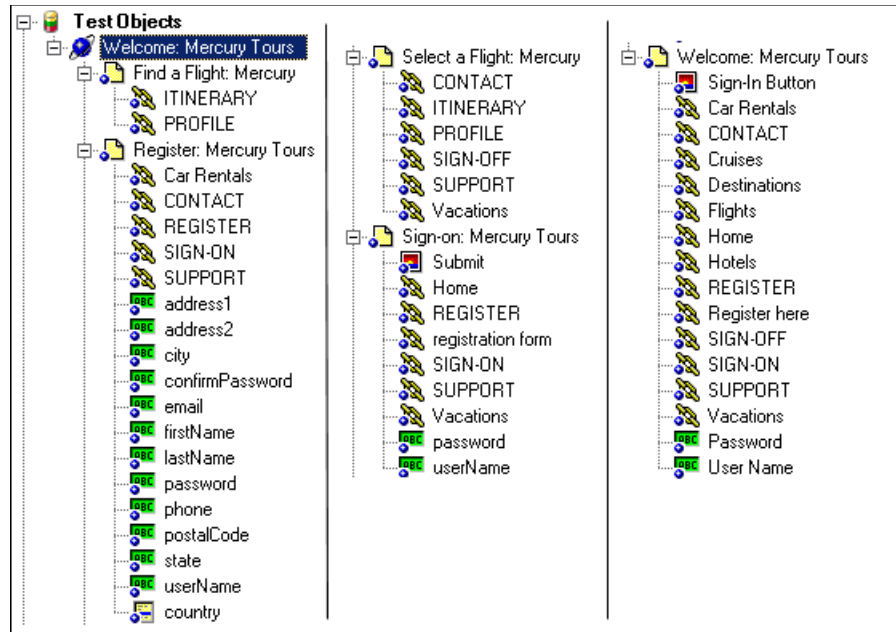
Most of the business processes we plan to test are in the central flight reservation area of the application and thus many of the same objects will be used in each of the relevant tests, but the sign on and registration processes are more standalone areas and it makes sense to store their objects separately. Thus it seems logical to create two object repository files:

- SignOn_Register
- Reservations

To create each of these repositories, we take advantage of the Navigate and Learn feature, which enables us to navigate to each page that is relevant for the object repository automatically learn all the objects in the page. By using the filter options in the Navigate and Learn feature, we can ensure that we learn only the types of objects we need. For example, we can avoid learning all the non-link image objects on every page, since these objects probably do not need to be tested and would otherwise result in a larger and less manageable object repository.

Afterwards, we should open the object repository for editing to delete specific objects that are not necessary and to rename objects that may otherwise be difficult to recognize when we later want to create steps with these objects.

Our **SignOn_Register** object repository may look something like this:



Note that each page contains only the relevant objects for the Sign on and Register business processes.

Create the Function Libraries and Functions Required for Testing the Mercury Tours Application

In some of our business processes, we want to test not only that the business processes can be performed to completion, but that certain features in the application behave as expected.

Because testing such functionality requires complex programming, and because we want to test the functionality in several different sub-processes, it makes sense to create these functionality checks in the form of functions, and to store them in function libraries, so that we can call the functions from more than one test action.

For example, we want to verify that the Mercury Tours application properly handles various invalid data in forms and we want to verify that the application properly calculates ticket prices for various types of itineraries.

We also want to make sure that we have ways to recover from certain application problems so that if such a problem occurs while a step is running, it does not prevent the action or test from completing its run or prevent other tests from running afterwards. This recovery function can be used by recovery scenarios that we will associate with our tests at later stages.

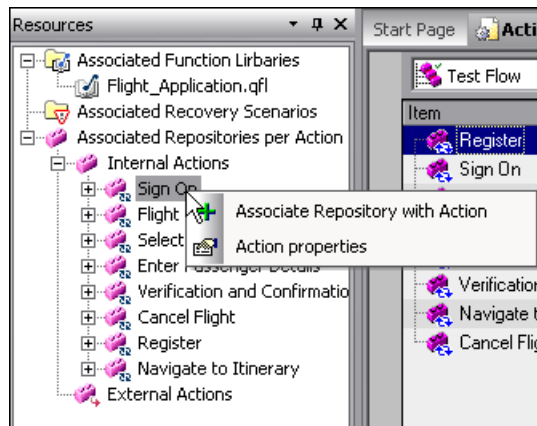
At this stage, we can create a function library containing functions such as:

- VerifyForm
- VerifyTicketPrice
- DataBaseFailureRecoveryFunction

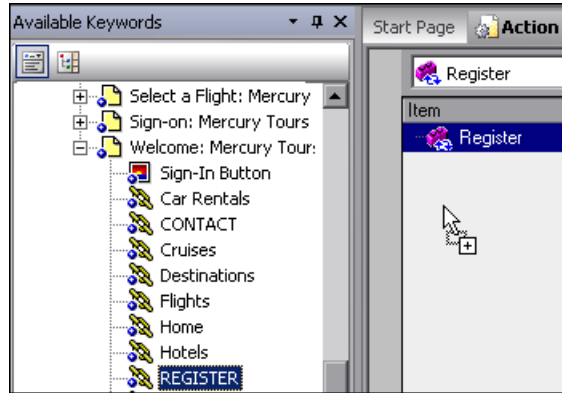
Create Tests and Test Steps for the Mercury Tours Business Processes

Now that we have planned and prepared all of the required resources for our tests, we are ready to use them to create tests and test steps that represent the steps a real user would perform on the Mercury Tours application as well as inserting functions that verify the expected functionality of various features.

We start by using the Resources pane to associate the relevant object repository with each action in the action repository test and to associate our function library with the test as well.



Then we use the Available Keywords pane to drag objects and functions into our actions to create the individual steps of each action, and add checkpoint and output value steps to verify expected behavior.



As we design our steps, we make sure to parameterize method arguments as necessary to maximize reusability of the actions in different business processes (tests).

Finally, we create new tests for each of the processes we defined in the Analyze the Mercury Tours Application step (see page 352). We use the Resources pane to associate our function library with each test and then we insert calls to the relevant actions.

13

Creating Tests Using the Recording Mechanism

You can create a test by recording the operations you perform on your application. After you create your test, you can enhance it using checkpoints and other special testing options.

Important: Before you begin recording, you need to ensure that your tests cover your testing requirements. For more information on planning your tests, see “Creating Tests — Overview” on page 309.

Tip: You can also create a test by using the keyword-driven methodology, which enables you to select keywords to indicate the operations you want to perform on your application, as described in “Creating Tests Using the Keyword-Driven Methodology” on page 335.

This chapter includes:

- About Recording Tests on page 362
- Recording a Test on page 364
- Choosing the Recording Mode on page 368
- Working with the Active Screen on page 376

About Recording Tests

You record your tests while navigating through your application. As you navigate, QuickTest graphically displays each step you perform as a row in the Keyword View and a line in the Expert View. A step is anything a user does that changes the content of a page or object in your application, for example, clicking a link or typing data in an edit box. Your test steps represent the operations you perform on your application. During a run session, QuickTest uses the recorded steps to replicate the operations you performed while recording.

While you record your test steps, QuickTest creates test objects representing the objects in your application on which you perform operations. This enables QuickTest to identify the objects in your application both while creating a test and during a run session.

Recording can be useful in the following circumstances:

- You are new to QuickTest and want to learn how QuickTest interprets the operations you perform on your application and how it converts them to QuickTest objects and built-in operations.
- You need to quickly create a test that tests the basic functionality of an application or feature, and the test does not require long-term maintenance.
- You are working with a new application or with major new features of an existing application, and you want to learn how QuickTest interacts with the application.
- You are developing functions that incorporate built-in QuickTest keywords.

After creating your initial test, you can further enhance it by adding and modifying steps in the Keyword View or Expert View.

Guidelines for Recording Tests

Consider the following when recording tests:

- If you are recording steps on a Web-based application, evaluate the types of events you need to record. If you need to record more or fewer events than QuickTest generally records by default, you can configure the events you want to record. For more information, see the section on configuring Web event recording in the *HP QuickTest Professional Add-ins Guide*.
- Consider increasing the power and flexibility of your test by replacing fixed values with parameters. When you parameterize your test, you can check how it performs the same operations with multiple sets of data, or from data stored or generated by an external source. For more information, see “Parameterizing Values” on page 625.
- Consider using actions to streamline the testing process. For more information, see “Working with Actions” on page 425.
- If you have useful WinRunner assets, you can link to WinRunner tests and call WinRunner TSL functions from your QuickTest test. For more information, see “Working with WinRunner” on page 1517.
- When you record tests, you may not notice that new objects are being added to the local object repository. This may result in many testers maintaining local object repositories with copies of the same objects. When using a keyword-driven methodology, you select the objects for your steps from the existing object repository. When you need a new object, you can add it to your local object repository temporarily, but you are also aware that you need to add it to the shared object repository for future use.
- When you record a test, QuickTest enters the correct objects, methods, and argument values for you. Therefore, it is possible to create a test with little preparation or planning.

Recording a Test

You can create the main body of a test by recording the typical processes that users perform. QuickTest records the operations you perform, displays them as steps in the Keyword View, and generates them in a script (in the Expert View).

Note that by default, each test includes a single action, but can include multiple actions. This chapter describes how to record a test with a single action. For information on why and how to work with multiple actions, see Chapter 15, “Working with Actions.”

By default, QuickTest records in the normal recording mode. If you are unable to record on an object in a given environment in the standard recording mode, or if you want to record mouse clicks and keyboard input with the exact x- and y-coordinates, you may want to record on those objects using analog or low-level recording. For more information, see “Choosing the Recording Mode” on page 368.

Tip: If you have objects that behave like standard objects, but are not recognized by QuickTest, you can define your objects as virtual objects. For more information, see Chapter 47, “Learning Virtual Objects.”

Consider the following when recording a test:

- Before you start to record, close all applications not required for the recording session.
- If you are recording on a Web site, determine the security zone of the site. When you record on a Web browser, the browser may prompt you with security alert dialog boxes. You may choose to disable/enable these dialog boxes.
- Decide how you want to open the application when you record and run your test. You can choose to have QuickTest open one or more specified applications, or record and run on any application that is already open. The Record and Run Settings dialog box contains tabbed pages corresponding to the add-ins loaded. For more information, see the section on setting Record and Run options in the *HP QuickTest Professional Add-ins Guide*.

- Choose how you want QuickTest to record and run your test by setting global testing options in the Options dialog box and settings specific to your test in the Test Settings dialog box. For more information, see Chapter 44, “Setting Global Testing Options” and Chapter 45, “Setting Options for Individual Tests.”
- If you are recording on a Web object, you must make a change to the object’s value to make QuickTest record the step. For example, to record a selection in a WebList object, you must click on the list, scroll to an entry that was not originally showing, and select it. If you want to select the item in the list that is already displayed, you must first select another item in the list (click it), then return to the originally displayed item and select it (click it).

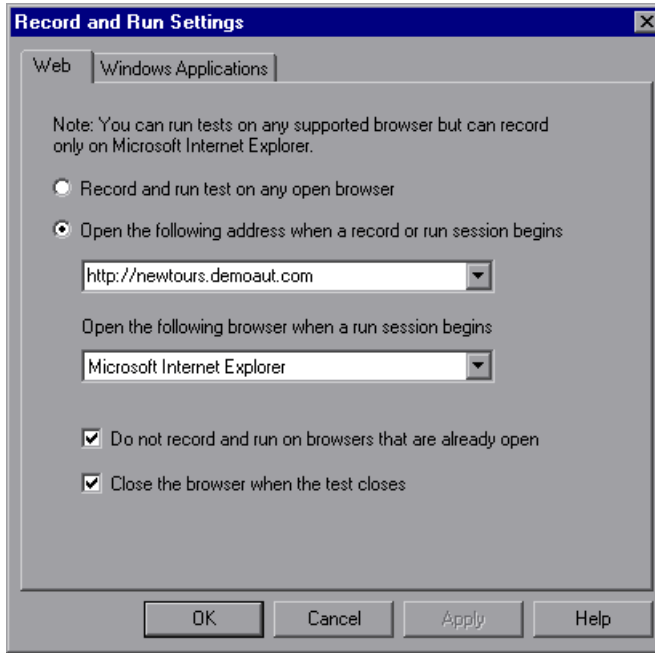
Note: If you are creating a test on Web objects, you can record your test on Microsoft Internet Explorer and run it on another supported browser (according to the guidelines specified in the *HP QuickTest Professional Readme*). QuickTest supports running tests on the following browsers—Microsoft Internet Explorer, Netscape Browser, Mozilla Firefox, and applications with embedded Web browser controls. For more information, see the *HP QuickTest Professional Add-ins Guide*.

To record a test:

- 1** Open QuickTest. For more information, see “Starting QuickTest” on page 20.
- 2** Open a test:
 - To create a new test, select **File > New > Test**, or click the down arrow next to the **New** button and select **Test**. Alternatively, click the **New** button down arrow and select **Test**.
 - To open an existing test, select **File > Open > Test** or click the down arrow next to the **Open** button and select **Test**. In the Open Test dialog box, browse to and select a test and click **Open**.

For more information, see “Managing Your Test” on page 321.


- 3 Click the **Record** button or select **Automation > Record**. If you are recording a new test and have not yet set your record and run settings in the Record and Run Settings dialog box (from **Automation > Record and Run Settings**), the Record and Run Settings dialog box opens.



After you set the record and run settings for a test, the Record and Run Settings dialog box will not open the next time you start a session in the same test. However, you can select **Automation > Record and Run Settings** to open the Record and Run Settings dialog box. You can use this option to set or modify your record and run preferences in the following scenarios:

- You have already recorded one or more steps in the test and you want to modify the settings before you continue recording.
- You want to run the test on a different application than the one you previously used.

The tabs available in the Record and Run Settings dialog box depend on the loaded add-ins.

- 4 Set the required options. For information on the tab to use and the options available for the environment you are testing, see the relevant add-in chapter in the *HP QuickTest Professional Add-ins Guide*.
- 5 To apply your changes and keep the Record and Run Settings dialog box open, click **Apply**.
- 6 Click **OK** to close the Record and Run Settings dialog box and begin recording your test.
- 7 Navigate through your application. QuickTest records each step you perform and displays it in the Keyword View and Expert View.
- 8 To determine if your application is functioning correctly, you can insert text checkpoints, object checkpoints, and bitmap checkpoints. For more information, see Chapter 17, “Understanding Checkpoints.”
- 9 You can parameterize your test to check how it performs the same operations with multiple sets of data, or with data from an external source. For more information, see Chapter 24, “Parameterizing Values.”
- 10 When you complete your recording session, click the **Stop** button, select **Automation > Stop**, or press the Stop command shortcut key. (To define a Stop command shortcut key, see “Setting Run Testing Options” on page 1253.)
-  11 To save your test, click the **Save** button or select **File > Save**. In the Save QuickTest Test dialog box, assign a name to the test. QuickTest suggests a default folder called **Tests**. For more information, see “Saving a Test” on page 324.

Choosing the Recording Mode

Normal recording mode records the objects in your application and the operations performed on them. This mode is the default and takes full advantage of the QuickTest test object model, recognizing the objects in your application regardless of their location on the screen.

When working with specific types of objects or operations, however, you may want to choose from the following, alternative recording modes:

- **Analog Recording.** Enables you to record the exact mouse and keyboard operations you perform in relation to either the screen or the application window. In this recording mode, QuickTest records and tracks every movement of the mouse as you drag the mouse around a screen or window.

This mode is useful for recording operations that cannot be recorded at the level of an object, for example, recording a signature produced by dragging the mouse.

Note: You cannot edit **Analog Recording** steps from within QuickTest.

- **Low Level Recording.** Enables you to record on any object in your application, whether or not QuickTest recognizes the specific object or the specific operation. This mode records at the object level and records all run-time objects as Window or WinObject test objects. Use low-level recording for recording in an environment or on an object not recognized by QuickTest. You can also use low-level recording if the exact coordinates of the object are important for your test.

Note: Steps recorded using **Low Level Recording** mode may not run correctly on all objects.

Guidelines for Analog and Low Level Recording

Consider the following guidelines when choosing **Analog Recording** or **Low Level Recording**:

- Use analog recording or low-level recording only when normal recording mode does not accurately record your operation.
- Analog recording and low-level recording require more disk space than normal recording mode.
- You can switch to either **Analog Recording** or **Low Level Recording** in the middle of a recording session for specific steps. After you record the necessary steps using analog recording or low-level recording, you can return to normal recording mode for the remainder of your recording session.

Analog Recording

- Use analog recording for applications in which the actual movement of the mouse is what you want to record. These can include drawing a mouse signature or working with drawing applications that create images by dragging the mouse.
- You can record in **Analog Recording** mode relative to the screen or relative to a specific window.
 - **Record relative to a specified window** if the operations you perform are on objects located within one window and that window does not move during the analog recording session. This ensures that during the run session, QuickTest will accurately identify the window location on which the analog steps were performed even if the window is in a different location when you run the analog steps. QuickTest does not record any click or mouse movement performed outside the specified window. When using this mode, QuickTest does not capture any Active Screen images.
 - **Record relative to the screen** if the window on which you are recording your analog steps moves during recording or if the operations you perform are on objects located within more than one window. This can include dragging and dropping an object from one window to another. When using this mode, QuickTest captures only the Active Screen image of the final state of the window on which you are recording.

- The steps recorded using analog recording are saved in a separate data file. This file is stored with the action in which the analog steps are recorded.
- When you record in **Analog Recording** mode, QuickTest adds to your test a **RunAnalog** statement that calls the recorded analog file. The corresponding Active Screen displays the results of the last analog step that was performed during the analog recording session.

Low Level Recording

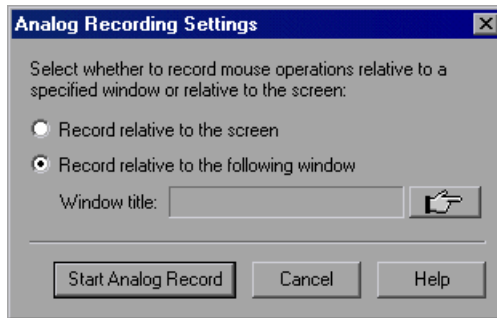
- Use low-level recording for recording on environments or objects not supported by QuickTest.
- Use low-level recording for when you need to record the exact location of the operation on your application screen. While recording in normal mode, QuickTest performs the step on an object even if it has moved to a new location on the screen. If the location of the object is important to your test, switch to **Low Level Recording** to enable QuickTest to record the object in terms of its x- and y- coordinates on the screen. This way, the step will pass only if the object is in the correct position.
- While low-level recording, QuickTest records all parent level objects as Window test objects and all other objects as WinObject test objects. They are displayed in the Active Screen as standard Windows objects.
- Low-level recording supports the following methods for each test object:
 - WinObject test objects: Click, DblClick, Drag, Drop, Type
 - Window test objects: Click, DblClick, Drag, Drop, Type, Activate, Minimize, Restore, Maximize
- Each step recorded in **Low Level Recording** mode is shown in the Keyword View and Expert View. (Analog recording records only the one step that calls the external analog data file.)

Using Analog Recording

You can switch to **Analog Recording** mode only while recording. The option is not available while editing.

To record in Analog Recording mode:

- 1 If you are not already recording, click the **Record** button to begin a recording session.
- 2 Click the **Analog Recording** button or select **Automation > Analog Recording**. The Analog Recording Settings dialog box opens.



- 3 Select from the following options:

- **Record relative to the screen.** QuickTest records any mouse movement or keyboard input relative to the coordinates of your screen, regardless of which application(s) are open or which application(s) you specified in the Record and Run Settings dialog box.

Select **Record relative to the screen** if you perform your analog operations on objects located within more than one window or if the window itself may move while you are recording your analog operations.

Note: When you record in **Analog Recording** mode relative to the screen, the run session will fail if your screen resolution or the screen location on which you recorded your analog steps has changed from the time you recorded.

The analog tracking continues to record the movement of the mouse until the mouse reaches the QuickTest screen to turn off **Analog Recording** or to stop recording. Clicking on the QuickTest icon in the Windows taskbar is also recorded. This should not affect your test. The mouse movements and clicks on the QuickTest screen itself are not recorded.

- **Record relative to the following window.** QuickTest records any mouse movement or keyboard input relative to the coordinates of the specified window.

Select **Record relative to the following window** if all your operations are performed on objects within the same window and that window does not move during analog recording. This guarantees that the test will run the analog steps in the correct position within the window even if the window's screen location changes after recording.

Note: If you have selected to record in **Analog Recording** mode relative to a window, any operation performed outside the specified window is not recorded while in **Analog Recording** mode.

- 4 If you choose to **Record relative to the following window**, click the pointing hand and click anywhere in the window on which you want to record in **Analog Recording** mode. The title of the window you clicked is displayed in the window title box.

For more information on using the pointing hand feature, see “Tips for Using the Pointing Hand” on page 374.


- 5 Click **Start Analog Record**.

6 Perform the operations you want to record in **Analog Recording** mode.

All of your keyboard input, mouse movements, and clicks are recorded and saved in an external file. When QuickTest runs the test, the external data file is called. It tracks every movement and click of the mouse to replicate exactly the operations you recorded.


**7** When you are finished and want to return to normal recording mode, click the **Analog Recording** button or select **Automation > Analog Recording** to turn off the option.

If you chose to **Record relative to the screen**, QuickTest inserts the **RunAnalog** step for a Desktop item. For example:

Item	Operation	Value
 Desktop	RunAnalog	"Track1"

Desktop.RunAnalog "Track1"

If you chose to **Record relative to the following window**, QuickTest inserts the **RunAnalog** step for a Window item. For example:

Item	Operation	Value
 Microsoft Internet Explorer	RunAnalog	"Track1"

Window("Microsoft Internet Explorer").RunAnalog "Track1"

The track file called by the **RunAnalog** method contains all your analog data and is stored with the current action.

You can use this track file in more than one action in your test, and also in other tests, by saving the action containing the **RunAnalog** step as a reusable action. A reusable action can be called by other tests or actions. For more information on using actions, see Chapter 15, "Working with Actions" and Chapter 16, "Working with Advanced Action Features."

Note: When entering the RunAnalog method, you must use a valid and existing track file as the method argument.

Tip: To stop an analog step in the middle of a run session, press CTRL + ESC, then click **Stop** in the Testing toolbar.

Tips for Using the Pointing Hand

- You can hold the left CTRL key to change the pointing hand to a standard pointer. You can then change the window focus or perform operations in QuickTest or in your application, such as right-clicking, using the scroll bars, or moving the pointer over an object to display a context menu.
- If the window containing the object you want to select is partially hidden by another window, hold the pointing hand over the partially hidden window for a few seconds until it comes to the foreground. Then point to and click the required object. You can configure the length of time required to bring a window into the foreground using the General pane of the Options dialog box.
- If the window containing the object you want to select is fully hidden by another window, or if a dialog box is hidden behind a window, press the left CTRL key and arrange the windows as needed.
- If the window containing the object you want to select is minimized, you can display it by holding the left CTRL key, right-clicking the application in the Windows task bar, and choosing **Restore** from the context menu.
- If the object you want to select can be displayed only by performing an event (such as right-clicking or moving the pointer over an object to display a context menu), hold the left CTRL key. The pointing hand temporarily turns into a standard pointer and you can perform the event. When the object you want to select is displayed, release the left CTRL key. The pointer becomes a pointing hand again.

Using Low Level Recording

You can switch to **Low Level Recording** mode only while recording a test. The option is not available while editing a test.

To record in Low Level Recording mode:

- 1 If you are not already recording, click the **Record** button to begin a recording session.



- 2 Click the **Low Level Recording** button or select **Automation > Low Level Recording**.

The record mode changes to **Low Level Recording** and all of your keyboard input and mouse clicks are recorded based on mouse coordinates. When QuickTest runs the test, the cursor retraces the recorded clicks.



- 3 When you are finished and want to return to normal recording mode, click the **Low Level Recording** button or select **Automation > Low Level Recording** to turn off the option.

The following examples illustrate the difference between the same operations recorded using normal mode and **Low Level Recording** mode.

Suppose you type the word `tutorial` into a user name edit box and then press the TAB key while in normal recording mode. Your test is displayed as follows in the Keyword View and Expert View:

▼ Welcome: Mercury Tours			
▼ Welcome: Mercury Tours			
userName	Set	"tutorial"	Enter "tutorial" in the "userName" e

```
Browser("Welcome: Mercury Tours").Page("Welcome: Mercury Tours").
  WebEdit("userName").Set "tutorial"
```

If you perform the same action while in **Low Level Recording** mode, QuickTest records the click in the user name box, followed by the keyboard input, including the TAB key. Your test is displayed as follows in the Keyword View and Expert View:

Microsoft Internet Explorer			
Internet Explorer_Server	Click	564,263	Click the "Internet Explorer_Server" object.
Internet Explorer_Server	Type	"tutorial"	Type "tutorial" in the "Internet Explorer_Server" object.
Internet Explorer_Server	Type	micTab	Type micTab in the "Internet Explorer_Server" object.

```
Window("Microsoft Internet Explorer").WinObject("Internet Explorer_Server").  
    Click 564,263  
Window("Microsoft Internet Explorer").WinObject("Internet Explorer_Server").  
    Type "tutorial"  
Window("Microsoft Internet Explorer").WinObject("Internet Explorer_Server").  
    Type micTab
```

Working with the Active Screen



The Active Screen provides a snapshot of your application as it appeared when you performed the corresponding step during a recording session. An Active Screen can be captured for every step you record. Additionally, depending on the Active Screen capture options that you used while recording, the page displayed in the Active Screen can contain detailed property information on each object displayed on the page. To view the Active Screen pane, click the **Active Screen** button or select **View > Active Screen**. For information on setting Active Screen recording options, see “Enhancing Your Test” on page 315.

The Active Screen enables you to parameterize object values and insert checkpoints, methods, and output values for almost any object in the page after you finish your recording session, even if your application is not available or you do not have a step in your test corresponding to the selected object.

You can specify the level at which QuickTest captures and stores information on objects while recording tests. For example, you can instruct QuickTest to capture all properties for all test objects on the captured screen, or only the properties of the recorded objects and their parents. For more information, see “Increasing or Decreasing the Active Screen Information Saved with a Test” on page 378.

If QuickTest captured object information while recording your test, you can use the Active Screen to add these objects to the local object repository. For information on configuring the Active Screen capture settings, see “Setting Active Screen Options” on page 1240. For information on adding objects to the object repository from the Active Screen, see “Adding Test Objects to a Local or Shared Object Repository” on page 136.

When QuickTest creates an Active Screen page for a Web-based application, it stores the path to images and other resources on the page, rather than downloading and storing the images with your test. Therefore, you may need to provide login information to view password-protected resources. For information on accessing password-protected resources in the Active Screen of a Web-based application, see the section on accessing password-protected resources in the active screen in the *HP QuickTest Professional Add-ins Guide*.

When working with Web-based applications, you can specify Active Screen display criteria for captured Web pages. For example, you can specify whether QuickTest should load ActiveX controls or Java applets. For more information, see “Setting Active Screen Options” on page 1240.

Active Screen pages for non-Web-based applications are based on a single bitmap capture of the visible part of the application window (or other top-level object), with context-sensitive areas representing each object displayed in the Active Screen.

You can choose whether or not to save the content of the Active Screen with your test. Saving the content of the Active Screen with your test is especially useful if you want to be able to edit the saved test directly from the Active Screen. Later, if you need to conserve disk space after you finish editing the test, and you plan to use your test only for test runs, you can save the test without the content of the Active Screen. (Tests without Active Screen files use significantly less disk space.) For more information, see “Increasing or Decreasing the Active Screen Information Saved with a Test”, below.

Increasing or Decreasing the Active Screen Information Saved with a Test

You can decide if and how much information you want to capture and save in the Active Screen. The more information you capture, the easier it is to add steps to your test using the many Active Screen options. However, more captured information also leads to slower recording and editing times. Removing or decreasing Active Screen information can be especially useful for conserving disk space after you have finished designing the test and you are using the test only for test runs.

If you find that the information saved in the Active Screen after recording is not sufficient for your test editing needs, or if you no longer need Active Screen information, and you want to decrease the size of your test, you can change the amount of Active Screen information saved with your test.

To increase or decrease the Active Screen information saved with your test:

- 1** Confirm that the Active Screen capture preference in the Active Screen pane of the Options dialog box is set to capture the amount of information you need. For more information, see “Setting Active Screen Options” on page 1240.
- 2** Perform one of the following:
 - Perform an **Update Run Mode** operation to save the required amount of information in the Active Screen for all existing steps. For more information on the **Update Run Mode** options, see “Updating a Test Using the Update Run Mode Option” on page 1125.
 - Re-record the step(s) containing the object(s) you want to add to the Active Screen by performing one of the following:
 - Select the step after which you want to record your step, position your application to match the selected location in your test, and then begin recording.
 - Place a breakpoint in your test at the step before which you want to add a step and run your test to the breakpoint. This brings your application to the point from which to record the step. For more information on setting breakpoints, see “Setting Breakpoints” on page 1079.

To stop saving Active Screen information (and reduce the disk space used by your test):

- 1** Open the relevant test in QuickTest.
- 2** Select **File > Save As** and clear the **Save Active Screen files** check box.

Note: If you clear this check box, your Active Screen files will not be saved, and you will not be able to edit your test using the options that are normally available from the Active Screen.

- 3** Click **Save** to apply your changes. For more information, see “Saving a Test” on page 324.

Tip: If you need to recover Active Screen files after you save a test without Active Screen files, re-record the necessary steps or use the **Update Run Mode** option to recapture screens for all steps in your test. For more information, see “Updating a Test Using the Update Run Mode Option” on page 1125.

Updating a Single Active Screen Capture

As the content of your application changes, you can continue to use the Active Screen from tests that you recorded previously. To do this, you update the selected Active Screen display so that you can use the Active Screen to add new steps to your test rather than re-recording steps on new or modified objects.

For example, suppose that one of the pages in your Web site now includes a new object and you want to add a checkpoint that checks this object. You can use the **Change Active Screen** command to replace the page in your Active Screen pane and then proceed to create a checkpoint for this object.

Note: It is also possible to update all Active Screen captures saved with a test using the Update Run Mode. For more information, see “Updating a Test Using the Update Run Mode Option” on page 1125.

To update a selected Active Screen capture:

- 1** Make sure that your application is displaying the window or page that you want to use to replace what is currently displayed in the Active Screen pane.
- 2** In the Keyword View, click a step that you want to change. The window or page is displayed in the Active Screen pane.
- 3** Select **Tools > Change Active Screen**. The QuickTest window is hidden and the mouse pointer becomes a pointing hand. For information about using the pointing hand feature, see “Tips for Using the Pointing Hand” on page 381.
- 4** Click the window or page displayed in your application.
- 5** When a message prompts you to change your current Active Screen display, click **Yes**.

Tips for Using the Pointing Hand

- You can hold the left CTRL key to change the pointing hand to a standard pointer. You can then change the window focus or perform operations in QuickTest or in your application, such as right-clicking, using the scroll bars, or moving the pointer over an object to display a context menu.
- If the window containing the object you want to select is partially hidden by another window, hold the pointing hand over the partially hidden window for a few seconds until it comes to the foreground. Then point to and click the required object. You can configure the length of time required to bring a window into the foreground using the General pane of the Options dialog box.
- If the window containing the object you want to select is fully hidden by another window, or if a dialog box is hidden behind a window, press the left CTRL key and arrange the windows as needed.
- If the window containing the object you want to select is minimized, you can display it by holding the left CTRL key, right-clicking the application in the Windows task bar, and choosing **Restore** from the context menu.
- If the object you want to select can be displayed only by performing an event (such as right-clicking or moving the pointer over an object to display a context menu), hold the left CTRL key. The pointing hand temporarily turns into a standard pointer and you can perform the event. When the object you want to select is displayed, release the left CTRL key. The pointer becomes a pointing hand again.

Tips for Improving Active Screen Performance

You can choose from the following Active Screen options to improve performance:

- If you are testing Windows-based applications, you can choose to save all Active Screen information in every step, save information only in certain steps, or to disable Active Screen captures entirely. You set this preference in the Active Screen pane of the Options dialog box. The less information saved, the faster your recording times will be. For more information, see “Setting Active Screen Options” on page 1240.
- If you are testing Web-based applications, you can disable screen capture of all steps in the Active Screen. From the Active Screen pane, click **Custom Level** to open the Custom Active Screen Capture Settings dialog box. Select the **Disable Active Screen Capture** option. This will improve recording time. For more information on the Active Screen pane of the Options dialog box, see “Setting Active Screen Options” on page 1240.
- If you are testing an application using a QuickTest add-in, see the *HP QuickTest Professional Add-ins Guide* to determine whether special Active Screen screen capture options exist for that environment.
- When you save a new test, or when you save a test with a new name using **Save As**, you can choose not to save the captured Active Screen files with the test, as described in step 2 of the procedure describing how to stop saving Active Screen information on page 379. Tests without Active Screen files use significantly less disk space.

14

Working with the Keyword View

The Keyword View provides an easy way to create, view, and modify tests in a graphical easy-to-use format.

This chapter includes:

- About Working with the Keyword View on page 384
- The Keyword View on page 385
- Understanding the QuickTest Object Hierarchy on page 391
- Adding a Standard Step to Your Test on page 392
- Adding Other Types of Steps to Your Test on page 407
- Modifying the Parts of a Step on page 410
- Working with Comments on page 410
- Managing Action Steps on page 412
- Using Keyboard Commands in the Keyword View on page 415
- Defining Keyword View Display Options on page 416
- Viewing Properties of Step Elements in the Keyword View on page 422
- Working with Breakpoints in the Keyword View on page 423

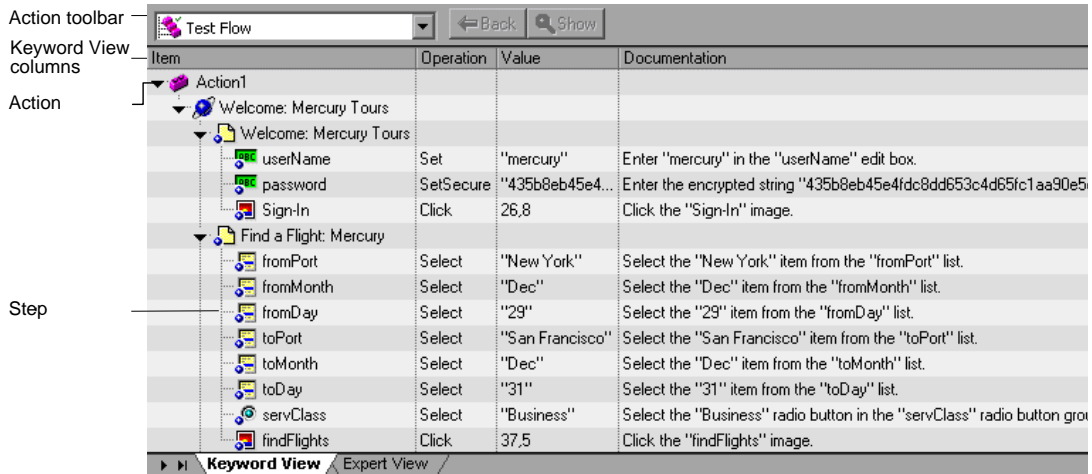
About Working with the Keyword View

The Keyword View enables you to create and view the steps of your test in a modular, table format. Each step is a row in the Keyword View that is comprised of individual, modifiable parts. You create and modify steps by selecting items and operations in the Keyword View and entering information as required. Each step is automatically documented as you complete it, enabling you to view a description of your test in understandable sentences. You can also use these descriptions as instructions for manual testing, if required.

You can use the Keyword View to add new steps to your test and to view and modify existing steps. When you add or modify a step, you select the test object or other step type you want for your step, select the method operation you want to perform, and define any necessary values for the selected operation or statement. Working in the Keyword View does not require any programming knowledge. The programming required to actually perform each test step is done automatically behind the scenes by QuickTest.

The Keyword View

The Keyword View enables you to create and view the steps of your test in a keyword-driven, modular, table format. The Keyword View is comprised of a table-like view, in which each step is a separate row in the table, and each column represents the different parts of the steps. The columns displayed vary according to your selection. For more information, see “Defining Keyword View Display Options” on page 416.



Actions are the highest level of the test hierarchy. They contain all the steps that are part of that action, and can include calls to other reusable actions. In the Keyword View, you can use the Action toolbar to view either the flow of all the top-level action calls in the test, or the content of a specific action. You can also display an action by double-clicking it in the Test Flow pane.

You can insert a new action, a call to an action, or a copy of an action, to your test. For more information on inserting and using actions in the Keyword View, see Chapter 15, “Working with Actions.”

Tip: You can copy and paste or drag and drop actions to move them to a different location within a test. For more information, see “Managing Action Steps” on page 412.

Each action is comprised of steps. Each step is inserted as a row in the Keyword View. For example, the Keyword View could contain the following rows:

Welcome: Mercury Tours			
userName	Set	"tutorial"	Enter "tutorial" in the "userName" e
password	SetSecure	"477cdb71935682eda"	Enter the encrypted string "477cdb7
Sign-In	Click	30,12	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:

- tutorial is entered in the **userName** edit box.
- An encrypted string is entered in the **password** edit box.
- The **Sign-In** image is clicked.
- The **Documentation** column translates each of the steps into understandable sentences.

For every step in the Keyword View, QuickTest displays a corresponding line of script in the Expert View. If you select a specific row in the Keyword View and switch to the Expert View, the cursor is located in the corresponding line of the script.

You can use the Keyword View to add steps at any point in your test. After you add steps, you can modify or delete them using standard editing commands and drag-and-drop functionality. You can print the contents of the Keyword View to your Windows default printer (and even preview the contents prior to printing). For more information, see “Printing a Test” on page 332.

In the Keyword View, you can also view properties for items such as checkpoints, output values, and actions, use conditional and loop statements, and insert breakpoints to assist you in debugging your test.

The Keyword View can contain any of the following columns: **Item**, **Operation**, **Value**, **Assignment**, **Comment**, and **Documentation**. A brief description of each column is provided below.

Item Column

The item on which you want to perform the step (test object, utility object, function call, or statement). This column displays a hierarchical icon-based tree. The highest level of the tree are actions, and all steps are contained within the relevant branch of the tree. Steps performed within the same parent object are displayed under that same object. Function calls, utility objects, and statements are placed in the tree hierarchy at the same level as the item above them (as a sibling).

You can collapse or expand an item in the item tree to change the level of detail that the tree displays.

- To collapse an item and its sub-items, click the arrow (▼) to the left of the item's icon, press the minus key (-) on your keyboard number pad, press the left arrow key on your keyboard, or right-click the item and select **Collapse Sub Tree**. The item tree hides all its sub-items and the collapse arrow changes to expand.
- To collapse all the items in the tree, select **View > Collapse All**.
- To expand an item one level or to its previously expanded state, select it and click the arrow (►) to the left of the item icon, press the plus key (+) on your keyboard number pad, press the right arrow key on your keyboard, or right-click the item and select **Expand Sub Tree**. The tree displays the details for the item and all its first-level sub-items and the expand arrows change to collapse.
- To expand an item and all its sub-items, select the item and press the asterisk (*) key on your keyboard number pad. The tree displays the details for the item and all its sub-items and the expand arrows change to collapse.
- To expand all the items in the tree, select **View > Expand All**.

Note: When you use the +, -, and * keys to expand and collapse the Item tree, make sure that the entire row is selected (by clicking to the left of the item's icon) and that a specific column is not selected, before pressing the required key. Otherwise, the keys will not work.

Operation Column

The operation to be performed on the item. This column contains a list of all available operations (methods, functions, or properties) that can be performed on the item selected in the **Item** column, for example, **Click** and **Select**. The default operation for the item selected in the **Item** column is displayed by default.

Value Column

The argument values for the selected operation, or the content of the statement. The **Value** cell is partitioned according to the number of arguments of the selected operation.

If an argument has a predefined list of values, QuickTest provides a drop-down list of possible values. If a list of values is provided, you cannot manually type a value in this box.

Assignment Column

The assignment of a value to or from a variable. For example, **Store in cCols** would store the return value of the current step in a variable called cCols, which you could then use later in the test.

You can select either **Store in** or **Get from**, depending on whether you want to retrieve the value from a variable or store the value in a variable. A **Store in X** value in the **Assignment** column is equivalent to an **X = <step>** line in the Expert View. A **Get From X** value in the Assignment column is equivalent to a **<step> = X** line in the Expert View. For more information on storing variables, see “Storing Return Values and Action Output Parameter Values” on page 794.

Comment Column

A free text edit box for any information you want to add regarding the step. These are also displayed as inline comments in the Expert View.

Note: You can also enter a comment on a new line below the currently selected step by choosing **Insert > Comment**. For more information, see “Adding Comments” on page 815.

Documentation Column

Read-only auto-documentation of what the step does in an easy-to-understand sentence, for example, Click the "Sign-in" image. or Select "San Francisco" in the "toPort" list. If you want to print or view only the steps, you can choose to display only this column. For example, you may want to print or view manual testing instructions.

Tips:

- You can display only the **Documentation** column of a test by right-clicking the column header row and choosing **Documentation Only** from the displayed menu.
 - You can also copy the documentation by selecting **Edit > Copy Documentation to Clipboard**, or right-clicking the column header row and choosing **Copy Documentation to Clipboard** from the displayed menu, and then paste it into a different application, as required.
-

Note: If you do not see one or more of these columns in the Keyword View, you can use the Keyword View Options dialog box to display them. For more information, see “Defining Keyword View Display Options” on page 416.

Tips for Working with the Keyword View

- You can use the left and right arrow keys to move the focus one cell to the left or right, with the following exceptions:
 - In the **Item** column, the left and right arrow keys collapse or expand the item (if possible). If not possible, the arrow keys behave as in any other column.
 - When a cell is in edit mode, for example, when modifying a value or comment, the left and right arrow keys move within the edited cell.
- When a **Value** cell is selected, press CTRL+F11 to open the Value Configuration Options dialog box.
- When the entire step is selected (by clicking to its left), use the + key (expands a specific branch), - key (collapses a specific branch), and * key (expands all branches) to expand and collapse the **Item** tree.
- When a row is selected (not a specific cell), you can type a letter to jump to the next row that starts with that letter.

Note: In addition to the above commands, you can also use QuickTest menu shortcuts. For more information, see “Performing QuickTest Commands” on page 46.

Understanding the QuickTest Object Hierarchy

The QuickTest test object hierarchy comprises one or more levels of test objects. The top level object may represent a window, dialog box, or browser type object, depending on the environment. The actual object on which you perform an operation may be learned as a top level object, a second level object, for example, `Window.WinToolbar`, or a third level object, for example, `Browser.Page.WebButton`.

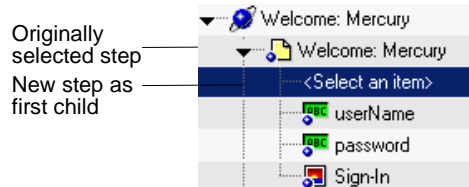
In some cases, even though the object in your application may be embedded in several levels of objects, the hierarchy does not include these objects. For example, if a `WebButton` object in your application is actually contained in several nested `WebTable` objects, which are all contained within a `Browser` and `Page`, the learned object hierarchy is only `Browser.Page.WebButton`.

An object that can potentially contain a lower-level object is called a container object. All top-level objects in the object hierarchy are container objects. If a second-level object contains third-level objects according to the QuickTest object hierarchy, then that object is also considered a container object. For example, in the step `Browser.Page.Edit.Set "David"`, `Browser` and `Page` are both container objects.

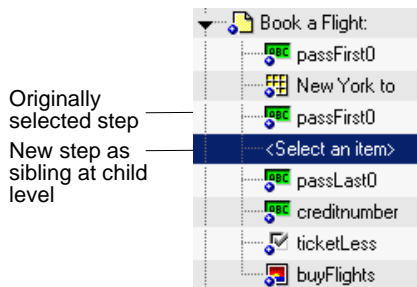
For information on the QuickTest object hierarchy for specific environments, see the relevant section in the *HP QuickTest Professional Add-ins Guide*.

When you add a step to your test in the Keyword View, the step is added as a sibling step or sub-step of the currently selected step, according to the QuickTest object hierarchy, as follows:

- If the selected step is a container object, the new step is inserted as the first sub-step of the container object.



- If the selected step is at the lowest level of the object hierarchy, the new step is inserted as a sibling step immediately after the selected step.



Adding a Standard Step to Your Test

You can use the Keyword View to add a step at any point in your test. You can add a step below the currently selected step, at the end of a test, or at the beginning of a new test. You can also add a new step immediately after a conditional or loop block, as described in “Adding a Standard Step After a Conditional or Loop Block” on page 409.

Tip: You can also add a step using the Step Generator. For more information, see “Inserting Steps Using the Step Generator” on page 777.

To add a standard step:**1** Perform one of the following:

- Click anywhere in the Keyword View (below the existing steps, if any) to add a step at the end of the test. If no steps are defined yet, this adds the first step to the test.
- Select **Insert > New Step** to add a new step after the existing steps (if any). If the test does not contain any steps, this adds the first step to the test.
- Select an existing step and select **Insert > New Step** to add a new step between existing steps. (If you select the last step, QuickTest adds a step at the end of the test.)
- Right-click an existing step and select **Insert New Step** from the context-sensitive menu.
- Drag and drop a test object from the Available Keywords pane to the Keyword or Expert view.

A new step is added to the Keyword View, either as a sibling step or a sub-step, according to the QuickTest object hierarchy, as described in “Understanding the QuickTest Object Hierarchy” on page 391.

Note: The **Select an item** list is generally expanded to display all applicable test objects, as well as the **Step Generator** and **Statement** items.

2 Define the step by clicking in the cell for the part of the step you want to modify and specifying its contents, as described below. Each cell in the step row represents a different part of the step. For each step, you can define the following:

- **Item.** A test object on which you perform a step. You must select an option from the **Item** column before you can add additional content to a step. For more information, see “Selecting an Item for Your Step” on page 395.
- **Operation.** The operation to be performed on the item. For more information, see “Selecting the Operation for Your Step” on page 403.

- **Value.** (If relevant.) The argument values for the selected operation. For more information, see “Defining Values for Your Step Arguments” on page 404.
- **Assignment.** (If relevant) The variable value. Double-click in the left part of the **Assignment** cell if you want to create or edit an assignment to or from a variable. Click the arrow button to select either **Get from** or **Store in**, depending on whether you want to retrieve the value from a variable or store the value in a variable. Click in the right part of the **Assignment** cell to specify or modify the name of the variable.

Note: The **Documentation** cell is read-only. This cell displays an explanation of what the step does in an easy-to-understand sentence, for example, Click the "Sign-in" image. or Select "San Francisco" in the "toPort" list. In most cases, QuickTest can generate the description displayed in this cell.

If you created a function library and associated it with the test, QuickTest can display documentation for it only if you defined the relevant text in the function library. For more information, see “Documenting the Function” on page 934 and “Working with User-Defined Functions and Function Libraries” on page 905.

Tip: You can use the standard editing commands (**Cut**, **Copy**, **Paste**, and **Delete**) in the **Edit** menu or in the context menu to make it easier to define or modify your steps. You can also drag and drop steps to move them to a different location within your action. For more information, see “Managing Action Steps” on page 412 and “Using Keyboard Commands in the Keyword View” on page 415.

- 3 After you make your changes, save the test. For more information, see “Saving a Test” on page 324.

Selecting an Item for Your Step

An item can be any of the following:

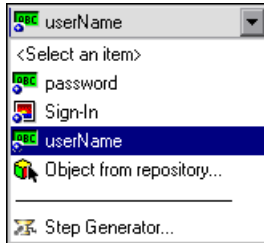
- A test object in the object repository.
 - You can either choose a test object from the list, or select **Object from repository** to open the Select Object for Step dialog box in which you can select a test object from the object repository or an object from your application. The test objects available in the list are the sibling and child test objects of the previous step's test object. The Select Object for Step dialog box contains all test objects in the object repository. You can select whether you want the operation for the step to be a test object operation or a run-time object operation. If you select a run-time object, an Object statement is added to the Keyword View.
 - You can drag and drop an object from the Available Keywords pane to your test. For more information, see “Understanding the Available Keywords Pane” on page 1165.
 - You can select an object directly from your application and add it to the object repository so that you can use it in the step.
- A statement, for example, a **Dim** statement.
- A step generated by the Step Generator. For more information, see “Inserting Steps Using the Step Generator” on page 777.

To select an item:

Click in the **Item** cell. Then click the down arrow and select the item on which you want to perform the step from the displayed list. When you insert a new step, the list is displayed automatically.

Selecting a Test Object from the Item List

The test objects available in the **Item** list are the sibling and child test objects of the previous step's test object, as defined in the shared object repository. The example below shows the objects available for the step following a **userName** test object.



To select a test object from the displayed Item list:

- 1 Click in the **Item** cell, then click the arrow button to display the **Item** list. If you have just created a new step, the list is displayed automatically as soon as you create the new step.
- 2 In the **Item** list, select the test object on which you want to perform the step. The item you select is displayed in the **Item** cell. You now need to specify an operation for the step. For more information, see “Selecting the Operation for Your Step” on page 403.

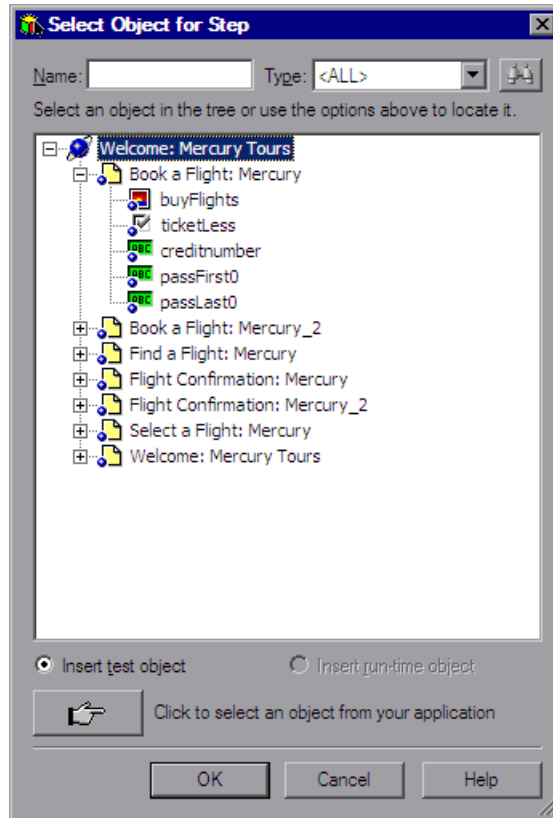
Selecting a Test Object from the Shared Object Repository

You can select any object in the object repository tree for your new step, or you can select the **Insert run-time object** option to enter an Object statement for the selected test object in your test. If the object repository is very large, you can search for the object. For example, you may want to add a password object that you know is an Edit box. You can search all the **Edit** type objects for one called **password**, or even one containing the letter **p**.

For more information on the object repository, see Chapter 5, “Managing Test Objects in Object Repositories.” For more information on Object statements, see “Accessing Native Properties and Operations” on page 887.

To select a test object from the shared object repository:

- 1** Click in the **Item** cell, then click the arrow button to display the **Item** list. If you have just created a new step, the list is displayed automatically as soon as you create the new step.
- 2** In the **Item** list, select **Object from repository**. The Select Object for Step dialog box opens.



- 3** Select an object from the object repository tree. If the object repository is very large, you can search for the object, as described below. If a search is not required, proceed to step 8.

- 4 In the **Name** box, enter the name of the object, or any part of the name. For example, you can enter p to search for all object names containing the letter p.

Note: If the **Name** box is left empty, all objects of the selected object type are considered matching criteria.

- 5 In the **Type** box, select the type of object for which to search, or select **<All>** to search for the object in all the object types.

Note: The object types in this list are a generic grouping of objects according to the general object characteristics. For example, the **List** type contains list and list view objects, as well as combo boxes; the **Table** type contains both tables and grids; and the **Miscellaneous** type contains a variety of other objects, such as WebElement and WinObject.



- 6 Click the **Find Next** button. The search starts at the currently selected node, and the number of objects that match your criteria is displayed. The first object in the list that matches your criteria is highlighted.
- 7 If required, click the **Find Next** button to navigate through all the objects that match your search criteria. The search continues to the end of the tree, then wraps to the beginning of the tree, and continues.

Tip: Press F3 to find the next object that matches your search criteria, or SHIFT+F3 to find the previous match.

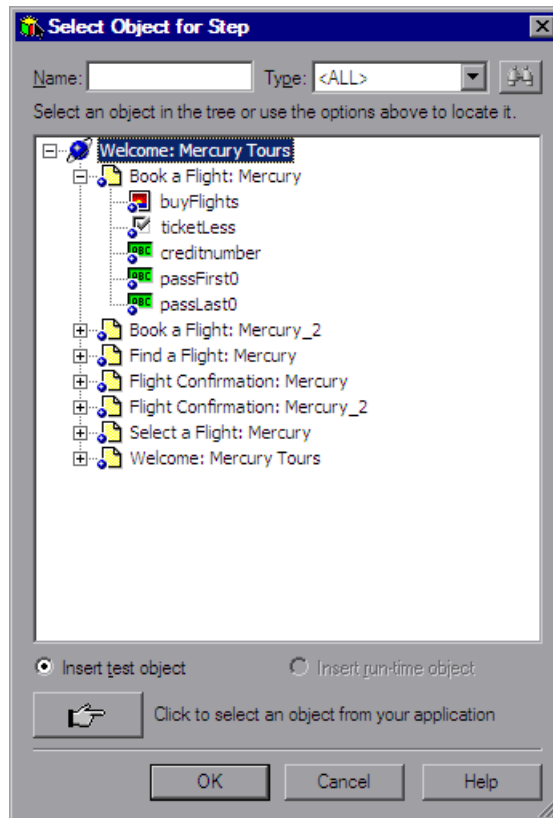
- 8 Click **OK**. The object is displayed in the **Item** column of the Keyword View, and is also added to the **Item** list. You can now specify the operation for the selected object. For more information, see “Selecting the Operation for Your Step” on page 403.

Selecting a Test Object from Your Application

If the shared object repository does not include the test object that you need for this step, you can select it directly from your application and add it to the shared object repository so that you can use it in this and other steps.

To add a test object from your application:

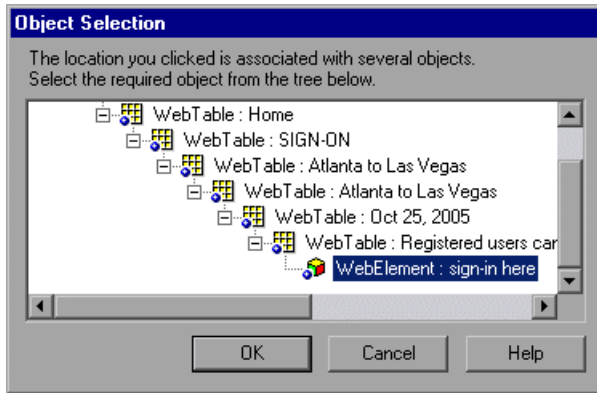
- 1** Click in the **Item** cell, then click the arrow button to display the **Item** list. If you have just created a new step, the list is displayed automatically as soon as you create the new step.
- 2** In the **Item** list, select **Object from repository**. The Select Object for Step dialog box opens.





- 3 Click the pointing hand button. QuickTest is hidden.
- 4 Use the pointing hand to click on the required object in your application. For more information about using the pointing hand feature, see “Tips for Using the Pointing Hand” on page 402.

If the location you clicked is associated with more than one object, the Object Selection dialog box opens.



- 5 Select the object for the new step and click **OK**. The object is displayed in the shared object repository tree in the Select Object for Step dialog box.

- 6 Click **OK**. The object is displayed in the **Item** column in the Keyword View. You can now specify the operation for the selected object. For more information, see “Selecting the Operation for Your Step” on page 403.

Tips:

- If you select an object in your application that is not in the shared object repository, a test object is added to the local object repository when you insert the new step. After you add a new test object to the local object repository, it is recommended to rename it, if its name does not clearly indicate its use. For example, you may want to rename a test object named **Edit** (that is used for entering a username) to **UserName**. This will enable other users to select the appropriate test object when adding steps using test objects located in this shared object repository.
 - After you add the required objects to the local object repository, you can use the Object Repository Merge Tool to update the shared object repository and make the new objects available to other tests. For more information, see “Updating a Shared Object Repository from Local Object Repositories” on page 269.
 - If you are adding a container test object, it is also recommended to specify its context, for example, if you are adding a confirmation message box from a Login page, you may want to name it **Login > Confirm**. For more information, see “Renaming Test Objects” on page 169.
-

Tips for Using the Pointing Hand

- You can hold the left CTRL key to change the pointing hand to a standard pointer. You can then change the window focus or perform operations in QuickTest or in your application, such as right-clicking, using the scroll bars, or moving the pointer over an object to display a context menu.
- If the window containing the object you want to select is partially hidden by another window, hold the pointing hand over the partially hidden window for a few seconds until it comes to the foreground. Then point to and click the required object. You can configure the length of time required to bring a window into the foreground using the General pane of the Options dialog box.
- If the window containing the object you want to select is fully hidden by another window, or if a dialog box is hidden behind a window, press the left CTRL key and arrange the windows as needed.
- If the window containing the object you want to select is minimized, you can display it by holding the left CTRL key, right-clicking the application in the Windows task bar, and choosing **Restore** from the context menu.
- If the object you want to select can be displayed only by performing an event (such as right-clicking or moving the pointer over an object to display a context menu), hold the left CTRL key. The pointing hand temporarily turns into a standard pointer and you can perform the event. When the object you want to select is displayed, release the left CTRL key. The pointer becomes a pointing hand again.

Selecting the Operation for Your Step

The **Operation** cell specifies the operation to be performed on the item listed in the **Item** column. The available operations vary according to the item selected in the **Item** column. When you select an item, all operations associated with that item are listed.

For example, if you selected a browser test object, such as a WebButton object, the list contains all of the available methods, such as Click or Exist.

To select an operation for the step:

Click in the **Operation** cell. Then click the down arrow button and select the operation to be performed on the item. The available operations vary according to the item selected in the **Item** column. For example, if you selected a browser test object, the list contains all of the methods and properties available for the browser object. If you selected a test object in the **Item** column, the default operation (most commonly-used operation) for the test object is automatically displayed in the **Operation** column. This cell is not applicable if you chose to insert a statement in the **Item** column.




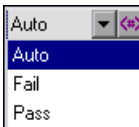
Note: Even if the **Item** column in the Keyword View is displayed to the right of the **Operation** column, you must still first select an item to view the list of available operations in the **Operation** column.

Defining Values for Your Step Arguments

The **Value** cell lists the values for each of the operation arguments. You can insert a constant value or a parameter for each argument.

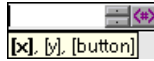
You can also encode password values. For more information, see “Inserting Encoded Passwords into Method Arguments and Data Table Cells” on page 406.

The **Value** cell is partitioned according to the number of possible arguments of the selected operation. Each partition contains different options, depending on the type of argument that can be entered in the partition, as follows:

Argument Partition	Argument Type	Instructions
	String	Enables you to enter a string containing English letters and numbers, enclosed by quotes. If you do not enter the quotes, QuickTest adds them automatically. If you modify a cell that contains a string enclosed by quotes by removing the quotes, QuickTest will not restore the quotes and the value will be treated as a variable name.
	Integer	Enables you to enter any number, or use the up and down arrows to select a number.
	Boolean	Enables you to select a True or False value from the list.
	Predefined Constant	Enables you to select a predefined value from the list. If a list of values is provided, you cannot manually type a value in this box.

To define or modify a value:

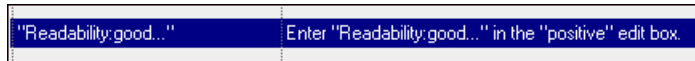
Click in each partition of the **Value** cell and enter the argument values for the selected operation. Note that when you click in the **Value** cell, a tooltip displays information for each argument. In the tooltip, the argument for the partition that is currently highlighted is displayed in bold, and any optional arguments are enclosed in square brackets.



Note: After you enter the initial value, you can edit the value at any time in the Keyword View for a test object, utility object, function call, conditional statement, or loop statement. You cannot edit the value of a regular statement, such as `x=10`, in the Keyword View after you define its initial value. You can edit the previously defined value of a regular statement only in the Expert View.


To add multi-line arguments:

Press SHIFT+ENTER to add line breaks to your argument value. After you enter a multi-line argument value, QuickTest automatically converts it to a string, and displays only the first line of the argument, followed by an ellipsis (...). This format for multi-line argument values is also displayed in the Documentation column of the Keyword View.



Tip: Select the cell to display the entire argument value to be used in the step. Note that the argument value is used during the run session exactly as it appears in the step. For example, if you enter quotation marks as part of the argument value, they are included in the argument value used during the run session. QuickTest automatically interprets a multi-line value as a string, so you do not need to add quotation marks for this purpose.

To parameterize the value for an argument:

Click the  button in the required **Value** cell. For more information, see “Parameterizing Values” on page 625.

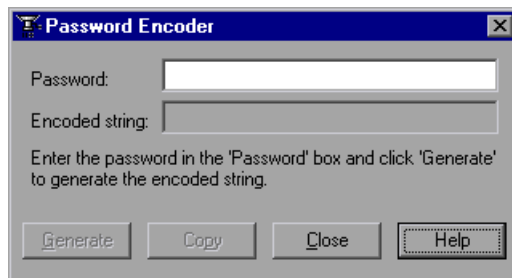
Inserting Encoded Passwords into Method Arguments and Data Table Cells

You can encode passwords to use the resulting strings as method arguments or Data Table parameter values. For example, your Web site may include a form in which the user must supply a password. You may want to test how your site responds to different passwords, but you also want to ensure the integrity of the passwords. The **Password Encoder** enables you to encode your passwords and place secure values into the Data Table.

Tip: You can also encrypt strings in Data Table cells using the Encrypt option in the Data Table menu. For more information, see “Data Menu” on page 1209.

To encode a password:

- 1 From the **Windows** menu, select **Start > Programs > QuickTest Professional > Tools > Password Encoder**. The Password Encoder dialog box opens.



- 2 Enter the password in the **Password** box.
- 3 Click **Generate**. The Password Encoder encrypts the password and displays it in the **Encoded String** box.

- 4 Use the **Copy** button to copy and paste the encoded value into the Data Table.
- 5 Repeat the process for each password you want to encode.
- 6 Click **Close** to close the Password Encoder.

Adding Other Types of Steps to Your Test








In addition to adding standard statement steps to your test using the Keyword View, you can also insert the following special types of steps using the relevant options from the **Insert** menu. Each step is entered as a row in the Keyword View, and you can then modify it as described in “Modifying the Parts of a Step” on page 410.

- You can insert a checkpoint step. For more information, see “Understanding Checkpoints” on page 495.
- You can insert an output value step. For more information, see “Outputting Values” on page 669.
- You can insert comments in steps to separate parts of an action or a test and to add details about a specific part. For more information, see “Adding Comments” on page 815.
- You can insert a step that sends information to the results, a step that puts a comment line in your test, a step that synchronizes your test with your application, or a step that measures a transaction in your test. For more information, see “Adding Steps Containing Programming Logic” on page 775.
- You can insert a step that calls a WinRunner test or function. For more information, see “Working with WinRunner” on page 1517.
- You can use conditional statements and loop statements in your test. For more information, see “Using Conditional and Loop Statements in the Keyword View” on page 408.

For information on adding a new step immediately after a conditional or loop block, see “Adding a Standard Step After a Conditional or Loop Block” on page 409.

Using Conditional and Loop Statements in the Keyword View

Using conditional statements, you can incorporate decision making into your tests. Using loop statements, you can run a group of steps repeatedly, either while or until a condition is true. You can also use loop statements to repeat a group of steps a specific number of times. Each statement type is indicated by one of the following icons in the Keyword View:

Icon	Type
	If...Then statement
	ElseIf...Then statement
	Else statement
	While...Wend statement
	For...Next statement
	Do...While statement
	Do...Until statement

After you insert a conditional or loop statement in the Keyword View, you can insert or record steps after the statement to include them in the conditional or loop block.

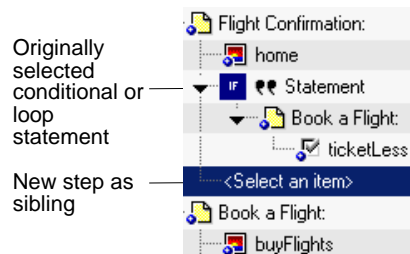
For information on including conditional and loop statements in your test, see Chapter 28, “Adding Steps Containing Programming Logic.”

Adding a Standard Step After a Conditional or Loop Block

After you add a conditional or loop statement to your test, all steps that you add or record are automatically inserted within the conditional or loop statement block. After you have finished adding steps to the block, you can add a step outside of the block, at a sibling level to the conditional or loop statement step, as described below. For more information on conditional and loop statements, see Chapter 28, “Adding Steps Containing Programming Logic.”

To add a standard step after a conditional or loop block:

- 1 Select the conditional or loop statement step after and outside of which you want to add the new step, and select **Insert > New Step After Block** or press **SHIFT+F8**. A new step is added to the Keyword View, at the end of the conditional or loop block, outside of the conditional or loop statement (as a sibling).



- 2 Specify the content of the step by modifying it, as described in “Adding a Standard Step to Your Test” on page 392.

Modifying the Parts of a Step

You can modify any part of a step in the Keyword View. For example, you can change the test object on which the step is performed, change the operation to be performed in the step, or add information regarding a step in the **Comment** column.

When working in the Keyword View, you can use the standard editing commands (**Cut**, **Copy**, **Paste** and **Delete**) in the **Edit** menu or in the context menu to make it easier to modify your steps.

Tip: You can copy and paste, or drag and drop steps to move them to a different location in an action. For more information, see “Managing Action Steps” on page 412.

To modify a step, click in the cell containing the part of the step you want to modify and specify the content of the cell. Each cell in the step row represents a different part of the step. For more information, see “Adding a Standard Step to Your Test” on page 392.

Working with Comments

A **Comment** is free text entry. You can insert a comment in the **Comment** cell of a step, or you can add a comment in a separate step. Using comments can help improve readability and make a test easier to update. For example, you may want to add a comment step at the beginning of each action to specify what that section includes.

After you add a comment, it is always visible as long as one or more columns are displayed. For information on selecting columns to display, see “Defining Keyword View Display Options” on page 416. QuickTest does not process comments when it runs a test.

To add a comment to an existing step:

Select the step and type your comment in the **Comment** column.

Note: You can also insert a comment step. For more information, see “Adding Comments” on page 815.

To modify an existing comment:

Double-click the comment in the **Comment** column. The cell becomes a free text field.

Managing Action Steps

You can move an action step before or after any other step in an action. You can also delete it if it is no longer required.

Note: You can also change the run order of actions in the test from the Test Flow pane. For more information, see “Using the Test Flow Pane” on page 431.

Moving an Action Step

You can move an action to a different location within a test, as needed, and you can move a step to a different location within an action.

To move an action or a step in the Keyword View:

- In the **Item** column, drag the step up or down and drop it at the required location within the action. When you drag a selected step, a line is displayed, enabling you to see the location to which the step will be moved. If you drag a step within its parent object, the step is displayed in the new position under its parent. If you move the step to a different parent object, the parent is duplicated, and the step is moved below it.

To move a top-level action to a different location in the test, use the Action toolbar to display the Test Flow and then drag the action up or down to the required location.

- Copy or cut the step to the Clipboard and then paste it in the required location. You can use **Edit > Copy** or CTRL + C to copy the step, **Edit > Cut** or CTRL + X to cut the step, and **Edit > Paste** or CTRL + V to paste the step. When you move, copy, or cut an action or step, you also move, copy, or cut all of its sub-steps, if any.

Notes:

- Conditional and loop blocks can only be copied or cut in their entirety. QuickTest does not enable you to copy or cut only the child nodes of conditional or loop blocks. After you copy or cut conditional or loop blocks to the Clipboard, QuickTest enables you to paste them only in valid locations.
 - You cannot copy or cut a parent object together with only some of its child objects. You must either select only the parent (which automatically includes all its child objects) or the parent object together with all of its children.
 - If you copy an action (**Insert > Call to Copy of Action**, right-click an action icon and select **Insert Call to Copy of Action**, or right-click any step and select **Action > Insert Call to Copy**), the Select Action dialog opens, which enables you to insert a call to a copy of an action. For more information on inserting a call to a copy of an action, see “Inserting Calls to Copies of Actions” on page 466.
-

Deleting an Action Step

You can delete an action step, if required. Before you delete a step, make sure that removing it will not prevent the action from running correctly. When an item has both an operation and sub-steps defined for it, as in the example below, you can choose whether to delete only the operation of the item, or to delete the item and all of its sub-steps.

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 Select **Edit > Delete** or press the DELETE key. One of the following messages is displayed, depending on the type of step you select:
 - If you select an item with either an operation (or checkpoint or output value) or sub-steps (but not both), a message opens asking if you want to delete the selected item and all of its sub-steps (if any).
 - If you select an item with both an operation (or checkpoint or output value) and sub-steps, a message opens asking whether you want to delete the selected item and all of its sub-steps, or delete only the item's operation (and leave the item and sub-steps).
- 3 Click **Delete Item** to delete the selected item (and any sub-steps), or click **Delete Operation** to delete only the operation for the selected item (and not delete the item).

Using Keyboard Commands in the Keyword View

If you prefer to use your keyboard, you can use the following keyboard commands to navigate within the Keyword View:

- Press F8 to add a new step below the currently selected step.
- Press SHIFT+F8 to add a new step after a conditional or loop block.
- Press F7 to use the Step Generator to add a new step below the selected step.
- The TAB and SHIFT+TAB keys move the focus left or right within a single row, unless you are in a cell that is in edit mode. If so, press ENTER to exit edit mode, and then you can use the TAB keys.
- When a cell containing a list is selected:
 - You can press SHIFT+F4 to open the list for that cell.
 - You can change the selected item by using the up and down arrow keys. In the **Item** column, the list must be open before you can use the arrow keys.
 - You can type a letter or sequence of letters to move to a value that starts with the typed letters. The typed sequence is highlighted in white.

Defining Keyword View Display Options

You can choose how you want to display the information in the Keyword View using the Keyword View Options dialog box. You can customize the display of the Keyword View columns, fonts, and colors. The options you set remain in effect for all tests in all subsequent sessions on your computer.

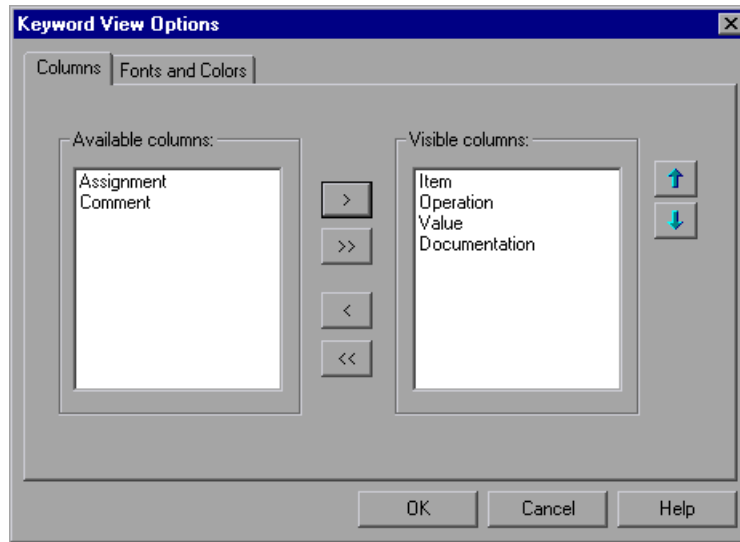
Displaying Keyword View Columns

You can use the Columns tab of the Keyword View Options dialog box to specify which columns you want to display in the Keyword View. You can also specify the order in which the columns are displayed.

Tip: You can display only the **Documentation** column by right-clicking the column header row and choosing **Documentation Only** from the displayed menu. You can then print the Keyword View for use as instructions for manual testing. For more information on printing from the Keyword View, see “Printing a Test” on page 332.

To specify the Keyword View columns to display:

- 1 Select **Tools > View Options**. The Keyword View Options dialog box opens.



The **Available columns** list shows columns not currently displayed in the Keyword View. The **Visible columns** list shows columns currently displayed in the Keyword View.

- 2 Double-click column names or choose column names and click the arrow buttons (> and <) to move them between the **Available columns** and **Visible columns** lists.

Tip: Click the double arrow buttons (>> and <<) to move all the column names from one list to the other. Select multiple column names (using the SHIFT and/or CONTROL keys) and click the arrow buttons (> and <) to move only the selected column names from one list to the other.



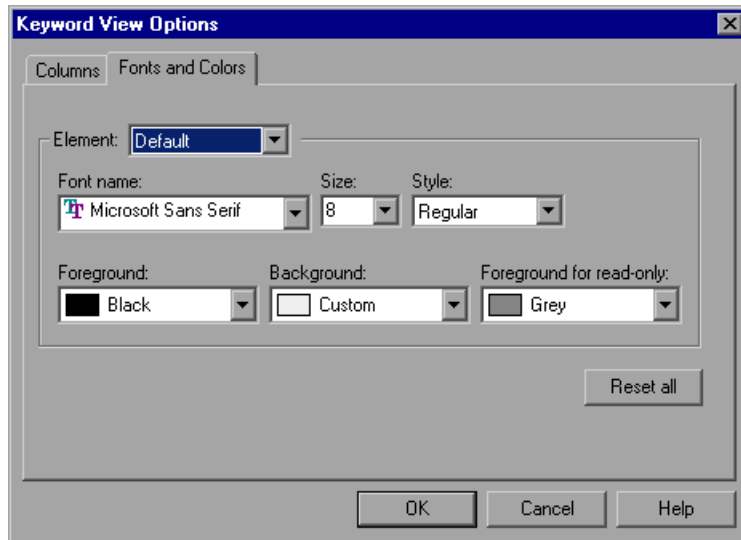
- 3 In the **Visible columns** list, set the order in which columns appear in the Keyword View by selecting one or more columns and then using the up and down arrow buttons.

Note: The order of the columns in the Keyword View does not affect the order in which the cells need to be completed for each step. For example, if you choose to display the **Operation** column to the left of the **Item** column, you still need to select the item first, and only then is the **Operation** column list refreshed to match the selection you made in the **Item** column.

- 4 Click **OK** to close the dialog box and apply the new column display.

Setting Keyword View Fonts and Colors

You can use the Fonts and Colors tab of the Keyword View Options dialog box to specify different text and color display options for different elements in the Keyword View.



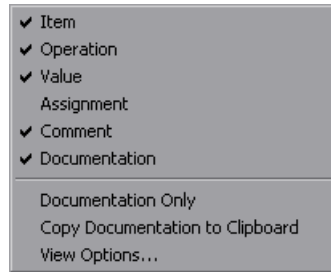
The Fonts and Colors tab includes the following options:

Option	Description
Element	<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.

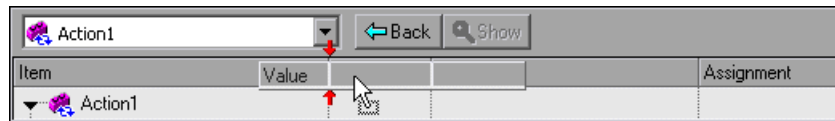
Tips for Working with the Keyword View

- 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.



For example, you can display only the **Documentation** column if you want to print the steps for use as instructions for manual testing, by selecting **Documentation Only**.

- You can rearrange columns by dragging a column header to its new location in the Keyword View. Red arrows are displayed when the column header is dragged to an available location.



Viewing Properties of Step Elements in the Keyword View

You can view properties for different parts of a step in the Keyword View. For example, you can view object properties, action properties, action call properties, checkpoint properties, and output value properties. Right-click the item whose properties you want to view, and select the relevant option from the displayed menu.

The property options available in the **Step** menu or the context (right-click) menu change according to the currently selected step. For example, if you right-click a step that contains a checkpoint or output value on a test object, you can view object properties and checkpoint or output value properties for the current object and checkpoint or output value. If you right-click an action, you can choose to view action properties or action call properties for the current action.

Working with Breakpoints in the Keyword View

You can insert and remove breakpoints in the Keyword View. When you place a breakpoint in a step in the Keyword View, it is also displayed in the Expert View, and vice versa.

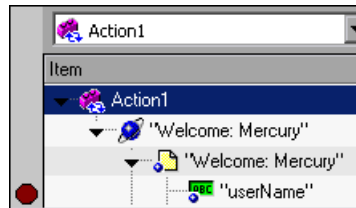
To insert a breakpoint in the Keyword View:

- Click in the left margin at the point where you want to insert the breakpoint.
- Select a step and press F9.
- Select **Debug > Insert/Remove Breakpoint**.

A red breakpoint icon  is displayed.

To remove a breakpoint from the Keyword View:

- Click the breakpoint icon.
- Select a step and press F9.
- Select **Debug > Insert/Remove Breakpoint**.



Note: QuickTest automatically places the breakpoint next to the appropriate item for the step. In the example shown above, even if you click next to the **Welcome: Mercury** browser or page item, the breakpoint is automatically inserted next to the **userName** edit item, on which the step is actually performed. When you collapse items, the breakpoint icons remain in the left margin next to the closest visible item, so you can see that the test contains breakpoints.

For more information on breakpoints, see “Using Breakpoints” on page 1078.

15

Working with Actions

You can divide your test into actions to streamline the process of testing your application. This chapter covers the basic use of actions in your test. Using advanced action-related features is described in Chapter 16, “Working with Advanced Action Features.”

This chapter includes:

- About Working with Actions on page 426
- Using Global and Action Data Sheets on page 429
- Using the Test Flow Pane on page 431
- Using the Action Toolbar in the Keyword View on page 435
- Creating New Actions on page 436
- Guidelines for Working with Actions on page 439
- Setting Action Properties on page 441
- Nesting Actions on page 453
- Splitting Actions on page 455
- Renaming Actions on page 457
- Removing Actions from a Test on page 460
- Creating an Action Template on page 462

About Working with Actions

Actions help divide your test into logical units, such as the main sections of a Web site, or specific activities that you perform in your application.

A test comprises calls to actions. When you create a new test, it contains a call to a single action. By creating tests that call multiple actions, you can design tests that are more modular and efficient.

An action consists of its own test script, including all of the steps in that action, and any objects in its local object repository.

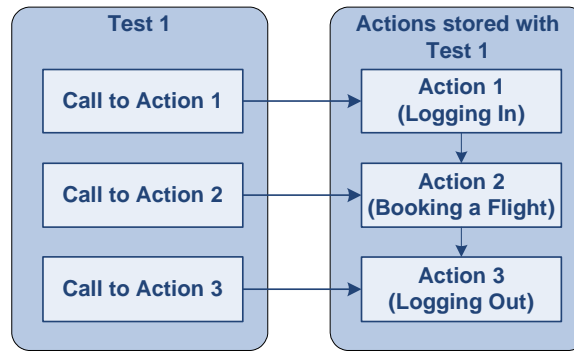
Each action is stored together with the test in which you created it. You can insert a call to an action that is stored with the test and, depending on the properties of the action, you may also be able to call an action stored with another test.

When you open a test, you can choose to view the test flow (calls to actions) or you can view and edit the individual actions stored with your test.

If you work with tests that include many steps or lines of script, it is recommended that you use actions to divide your test steps. Actions should ideally contain no more than a few dozen test steps.

For example, suppose you want to test several features of a flight reservation system. You plan several tests to test various business processes, but each one requires the same login and logout steps. You can create one action that contains the steps required for the login process, another for the logout steps, and other actions for the main steps in your test. After you create the login and logout actions, you can insert those actions into other tests.

If you create a test in which you log into the system, book one flight, and then log out of the system, your test might be structured as shown—one test calling three separate actions:



Actions enable you to parameterize and iterate over specific elements of a test. They can also make it easier to modify steps in one action when part of your application changes.

For every action called in your test, QuickTest creates a corresponding action sheet in the Data Table so that you can enter Data Table parameters that are specific to that action only. For more information on global and action data sheets, see “Using Global and Action Data Sheets” on page 429. For information on parameterizing tests, see Chapter 24, “Parameterizing Values,” and Chapter 25, “Outputting Values.”

Using Multiple Actions in a Test

When you create a test, it includes one action. All the steps you add and all the modifications you make while editing your test are part of a single action.

You can divide your test into multiple actions by creating new actions and inserting calls to them, by inserting calls to existing actions, or by splitting existing actions. The actions used in the test, and the order in which they are run, are displayed in the Test Flow pane.

There are three kinds of actions:

- **Reusable action.** An action that can be called multiple times by the test with which it is stored (the local test), as well as by other tests.
- **Non-reusable action.** An action that can be called only in the test with which it is stored, and can be called only once.
- **External action.** A reusable action stored with another test. External actions are read-only in the calling test, but you can choose to use a local, editable copy of the Data Table information for the external action.

For more information on creating and calling new actions, see “Creating New Actions” on page 436. For more information on inserting calls to existing actions, see “Nesting Actions” on page 453.

By default, new actions are reusable. You can mark each action you create in a test as reusable or non-reusable. Only reusable actions can be called multiple times from the current test or from another test. You can store a copy of a non-reusable action with your test and then insert a call to the copy, but you cannot directly insert a call to a non-reusable action saved with another test. Inserting calls to reusable actions makes it easier to maintain your tests, because when an object or procedure in your application changes, it needs to be updated only one time, in the original action.

Two or more tests can call the same action and one action can call another action (this is known as nesting an action, described in “Nesting Actions” on page 453). Complex tests may have many actions and may share actions with other tests.

When you run a test with multiple actions, the test results are divided by actions within each test iteration so that you can see the outcome of each action, and you can view the detailed results for each action individually. For more information on the Test Results window, see Chapter 33, “Viewing Run Session Results.”

Using Global and Action Data Sheets

When you output a value to the Data Table or add a Data Table parameter to your test, you can specify whether to store the data in the **Global** data sheet or in the **action** data sheet.

- Choosing **Global sheet** enables you to create a new column or select an existing column in the **Global** sheet in the Data Table. When you run your test, QuickTest inserts or outputs a value from or to the current row of the Global data sheet during each global iteration. You can use the columns in the Global data sheet for Data Table output values or Data Table parameters in any action. This enables you to pass information between actions.
- Each action also has its own sheet in the Data Table so that you can insert data that applies only to that action. Choosing **Current action sheet (local)** enables you to create a new column or select an existing column in the corresponding action sheet in the Data Table. Note that the name of the action sheet is the same as the name of the relevant action. When you run your test, QuickTest inserts or outputs a value from or to the current row of the current action (local) data sheet during each action iteration.

When there are parameters or output value steps in the current action's sheet, you can set QuickTest to run one or more iterations on that action before continuing with the current global iteration of the test. When you set your action call properties to run iterations on all rows, QuickTest inserts the next value from or to the corresponding action parameter or output value during each action iteration, while the values of the global parameters stay constant.

Note: If you create Data Table parameters or output value steps in your action and select to use the **Current action sheet (local)** option, be sure that the run settings for your action are set correctly in the Run tab of the Action Call Properties dialog box. You can set your action to run without iterations, to run iterations on all rows in the action's data sheet, or to run iterations only on the rows you specify. For more information on setting action iteration preferences, see "Inserting a Call to an Existing Action" on page 468.

For example, suppose you want to test how a flight reservation system handles multiple bookings. You may want to parameterize the test to check how your site responds to multiple sets of customer flight itineraries. When you plan your test, you plan the following procedures:

- 1** The travel agent logs into the flight reservation system.
- 2** The travel agent books five sets of customer flight itineraries.
- 3** The travel agent logs out of the flight reservation site.

When you consider these procedures, you realize that it is necessary to parameterize only the second step—the travel agent logs into the flight reservation system only once, at the beginning, and logs out of the system only once, at the end. Therefore, it is not necessary to parameterize the login and logout procedures in your test.

By creating three separate actions within your test—one for logging in, another for booking a flight, and a third for logging out—you can parameterize the second action in your test without parameterizing the others.

For more information on the Data Table, see Chapter 42, “Working with Data Tables.” For more information on parameterization, see Chapter 24, “Parameterizing Values.” For more information on output values, see Chapter 25, “Outputting Values.”

Using the Test Flow Pane

The Test Flow pane enables you to view all the calls to actions in the current test and the order in which they are run. From the Test Flow pane, you can display test, action, and action call properties, manage actions and change their order in the test, work with the object repository, and run specific actions.

For more information, see:

- “Understanding the Test Flow Pane” on page 432
- “Working with Actions in the Test Flow Pane” on page 433















Note: The Test Flow pane is displayed by default when you start QuickTest Professional. To hide or show the pane, select **View > Test Flow** or click the **Test Flow Pane** toolbar button.

When you double-click an action in the Test Flow pane, the Keyword View and Expert View show only the selected action. In the Keyword View and Expert View, you can view and edit the individual steps of an action stored in this test, and view the steps for each selected external action.

- The Keyword View displays the steps of your test in a modular, table format. For more information on the Keyword View, see Chapter 14, “Working with the Keyword View.”
- The Expert View displays the script for the selected action. For more information on the Expert View, see Chapter 29, “Working in the Expert View and Function Library Windows.”

Understanding the Test Flow Pane

The Test Flow pane uses the following icons to indicate the different types of item in the hierarchy:

Icon	Description
	A test
	A call to a non-reusable action
	A call to an external action
	A call to a conditional, external action
	A call to a reusable action
	A call to a conditional, reusable action
	A call to a missing action (an action whose path is not saved with the test)
	A call to a conditional, missing action
	A call to a looped, reusable action
	A call to a conditional, looped, reusable action
	A call to an external, looped action
	A call to a conditional, external, looped action

Tips:

- You can right-click in the Test Flow pane title bar to view available display options and decide how to display the Test Flow pane. For example, you can auto hide the pane, dock it, or close it.
- You can click the **Test Flow Pane** toolbar button to hide or show the Test Flow pane view.



Working with Actions in the Test Flow Pane

You can perform the following operations in the Test Flow pane:

- **Display an action in the Keyword View and Expert View.** Double-click an action in the Test Flow pane to show only that action in the Keyword View and Expert View.
- **View or hide the sub-nodes in the test.** Right-click the **Test** node in the tree and select **Expand All** or **Collapse All** to view or hide the sub-nodes in the tree. You can also select the **Test** node and press + or * on the keyboard to expand all the nodes in the test, and - to collapse the nodes.
- **Display the test properties.** Right-click the **Test** node in the tree and then select **Settings** to display the Test Settings dialog box. Details of the test and its path are displayed. For more information on the Test Settings dialog box, see “Using the Test Settings Dialog Box” on page 1262.
- **View or hide the sub-nodes of an action.** Right-click an action in the tree and then select **Expand Sub Tree** or **Collapse Sub Tree** to view or hide the sub-nodes in the action. You can also select a sub-node and press + or * on the keyboard to expand the node and - to collapse the node.
- **Display the action properties.** Right-click an action in the tree and then select **Action Properties** to display the Action Properties dialog box. The name of the action and its path are displayed. For more information on the Action Properties dialog box, see “Setting Action Properties” on page 441.
- **Display the action call properties.** Right-click an action in the tree and then select **Action Call Properties** to display the Action Call Properties dialog box. For more information on the Action Call Properties dialog box, see “Setting Action Call Properties” on page 481.
- **Work with the Object Repository.** Right-click an action in the tree and then select **Object Repository** to open the Object Repository window, which displays a tree containing all objects in the current test. For more information, see Chapter 7, “Managing Object Repositories.”

- **Manage Actions.** Right-click an action in the tree and then select **Copy** or **Delete**.
 - Select **Copy** to open the Select Action dialog box and create a copy of the action in your test. For more information, see “Inserting Calls to Copies of Actions” on page 466.
 - Select **Delete** to remove the action from your test. For more information, see “Removing Actions from a Test” on page 460.
- **Run the test.** Right-click an action in the tree and then select **Run from Action** or **Run to Action** to start a run session from the beginning of the selected action, or to run the test until the beginning of the selected action and then pause the run session.
- **Debug your test.** Right-click an action in the tree and then select **Debug from Action** to begin (and pause) a debug session at the beginning of the selected action.
- **Change the run order of actions.** You can perform either of the following steps to move a top-level action (a direct child of the test) in the Test Flow Pane tree, and change the run order of the test accordingly. The action and any sub-actions are moved.
 - Right-click a top-level action in the tree and then select **Move Up** or **Move Down**. You can also press CTRL+UP arrow or CTRL+DOWN arrow to move an action and its sub-actions.
 - Drag a top-level action in the tree up or down to the required location. When you drag a selected action, a line is displayed, enabling you to see the location in the tree to which the action will be moved. You can only drag top-level actions. Selecting the parent action automatically includes all its sub-actions. You cannot drag a sub-action, nor can you drag a parent action together with only some of its sub-actions.

For more information on moving actions, see “Managing Action Steps” on page 412.

If a test contains a call to an action that does not exist or cannot be found, the action still appears in the tree in the Test Flow pane, and QuickTest lists the action in the Missing Resources pane.

Using the Action Toolbar in the Keyword View

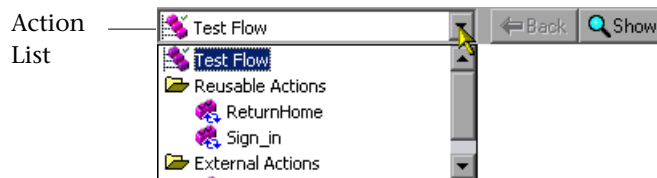
The Action toolbar contains options that enable you to view the top-level actions in the test flow or to view any action stored with your test (whether or not the action is actually called in the test). The Action toolbar is automatically displayed above the Keyword View when a reusable or external action is included in test.



Tip: You can display or hide the Action toolbar in the Keyword View by choosing **View > Toolbars > Action**. For more information, see Chapter 2, “QuickTest at a Glance.”

In the Expert View, the Action List is always visible and the Expert View always displays the steps for the selected action. For more information on the Expert View, see Chapter 29, “Working in the Expert View and Function Library Windows.”

The **Action List** enables you to view either the test flow (the calls to the top-level actions in the test) or you can view the steps for a selected reusable or external action. Selecting **Test Flow** in the Action List displays the overall flow of your test with all the calls to the top-level actions in your test. The test flow also enables you to view and edit the individual steps of non-reusable actions. An action view displays all the details of the selected reusable or external action.



In the test flow, reusable actions are not expandable. You can view the expanded steps of a reusable action by selecting the action from the Action List. For more information on reusable actions, see “Setting General Action Properties” on page 443.

There are several ways to open the action view for a reusable or external action in the Keyword View:

- In the Test Flow pane, double-click the action you want to view.
- Use the Action toolbar to display the top-level Test Flow and then double-click the call to the action you want to view.
- Use the Action toolbar to display the top-level Test Flow and then highlight the call to the action you want to view and click the **Show** button.
- Select the name of the action from the Action List.

You may have actions that are stored with your test, but are not currently called from your test. (They may be called by other tests, and you can insert calls to these actions from within your test, if needed).

Actions that are not called in your test are not displayed in the Test Flow pane, but they are displayed in the Action List. You can select these actions to view or edit their contents.

If an action is stored with your test but is not called by the test, and you are sure that you do not need the action for this test or any other test, you can delete the action from the test. For more information, see “Removing Actions from a Test” on page 460.

Creating New Actions

You can create new actions and add calls to them, as needed.

You can call the new action from your test flow as a top-level action, or you can call the new action from within another action in your test as a sub-action (or nested action). For more information, see “Nesting Actions” on page 453.

You can also split an existing action into two actions. For more information on splitting actions, see “Splitting Actions” on page 455.

To create a new action in your test:

- 1 If you want to insert a call to the new action from an existing action in your test, click the step after which you want to insert the new action. To insert a call to the new action from the test flow as a top-level action, click any step.



- 2 Select **Insert > Call to New Action** or click the **Insert Call to New Action** button on the **Insert** toolbar. The Insert Call to New Action dialog box opens.

- 3 In the **Name** box, type a new action name or accept the default name. If you rename the action, make sure that the action name is unique (within the test), does not exceed 1023 characters, does not begin or end with a space, and does not contain the following characters:
`\ / : * ? " < > | % ' ! { }`
- 4 In the **Description** box, add a description of the action. You can also add an action description at a later time using the Action Properties dialog box.

Tip: Descriptions of actions are displayed in the Select Action dialog box. The description makes it easier for you to choose an existing action you want to call. For more information, see “Setting General Action Properties” on page 443.

- 5 Ensure **Reusable Action** is selected if you want to be able to call the action from other tests or multiple times from within this test. By default, this option is selected. You can also set or modify this setting at a later time using the Action Properties dialog box.

For more information on reusable actions, see “Using Multiple Actions in a Test” on page 427. For more information on the Action Properties dialog box, see “Setting Action Properties” on page 441.

- 6 Decide where to insert the call to the action by selecting **At the end of the test** or **After the current step**. Choosing **At the end of the test** creates a call from the test flow to a top-level action. Choosing **After the current step** inserts the call to the action from within the current action (nests the action).

Note: If the currently selected step is a reusable action from another test, the new action is added automatically to the end of the test (the location options are disabled).

For more information on inserting action calls within actions, see “Nesting Actions” on page 453.

- 7 Click **OK**. A new action is stored with your test and the call to it is displayed at the bottom of the test or after the current step. You can move your action call to another location at a parallel (sibling) level within your test by dragging it to the desired location. For more information on moving actions, see “Using the Test Flow Pane” on page 431 and “Managing Action Steps” on page 412.
- 8 If you inserted the call to the new action while editing your test, make sure your new action is selected before adding steps to it.

Guidelines for Working with Actions

Consider the following guidelines when working with actions:

- If your action runs more than one iteration, the action must end at the same point in your application as it started, so that it can run another iteration without interruption. For example, suppose you are testing a sample flight reservation site. If the action starts with a blank flight reservation form, it should conclude with a blank flight reservation form.
- A single test may include both global Data Table parameters and action (local) Data Table parameters. For example, you can create a test in which a travel agent logs into the flight reservation system, books three flights, and logs out; the next travel agent logs into the flight reservation system, books three flights, logs out, and so forth.

To parameterize the 'book a flight' action, you select **Current action sheet (local)** in the parameterization dialog box and enter the three flights into the relevant **Action** tab in the Data Table. To parameterize the entire test, you select **Global** in the parameterization dialog box and enter the login names and passwords for the different agents into the **Global** tab in the Data Table.

Your entire test runs one time for each row in the Global data sheet. Within each test, each parameterized action is repeated according to the number of rows in its data sheet and the run settings selected in the Run tab of the Action Properties dialog box.

- You may want to rename the actions in your test with descriptive names to help you identify them. It is also a good idea to add detailed action descriptions. This facilitates inserting actions from one test to another. You can rename an action by choosing **Edit > Action > Rename Action**. (Make sure you follow the naming conventions for actions. For more information, see “Creating New Actions” on page 436.)

- If you plan to use an identical or virtually identical procedure in more than one test, you should consider inserting a call to an action from another test.
 - If you want to make slight modifications to the action in only one test, you should use the **Insert Call to Copy of Action** option to create a copy of the action.
 - If you want modifications to affect all tests containing the action, you should use the **Insert Call to Existing Action** option to insert a link to the action from the original test.
 - If you want modifications to the action to affect all tests containing the action, but you want to edit data in a specific test's Data Table, use the **Insert Call to Existing Action** option and, in the External tab of the Action Properties dialog box, select **Use a local, editable copy**.
- Reusable actions help you to maintain your tests, but it is important to consider the effects of including reusable actions in tests. Be sure to consider how changes to an action could potentially affect other tests that call that action.
- If you expect other users to open your tests and all actions in your tests are stored in the same drive, you should use relative paths for your reusable actions so that other users will be able to open your tests even if they have mapped their network drives differently.

Note: If you are working with the Resources and Dependencies model with Quality Center 10.00, specify an absolute Quality Center path. For more information, see “Considerations for Working with Relative Paths in Quality Center” on page 1450.

- If you expect certain elements of your application to change regularly, it is a good idea to divide the steps related to changeable elements into a separate action so that it will be easy to change the required steps, if necessary, after the application is modified.
- If you decide to remove an action, consider how that might affect your test or another test that contains a call to that action. For example, will it prevent a later action in the same test from running correctly? Will it cause the test containing a call to that action to fail?

- When you insert a call to an external action, the action is inserted in read-only format, and the **Record** button is disabled. If you want to record, you first need to insert a call to a reusable or non-reusable action into your test, or select a step from a reusable or non-reusable action that already exists in your test.

Setting Action Properties

The Action Properties dialog box enables you to define options for the stored action. These settings apply each time the action is called. You can modify an action name, add or modify an action description, and set an action as reusable or non-reusable. For an external action, you can set the Data Table definitions.

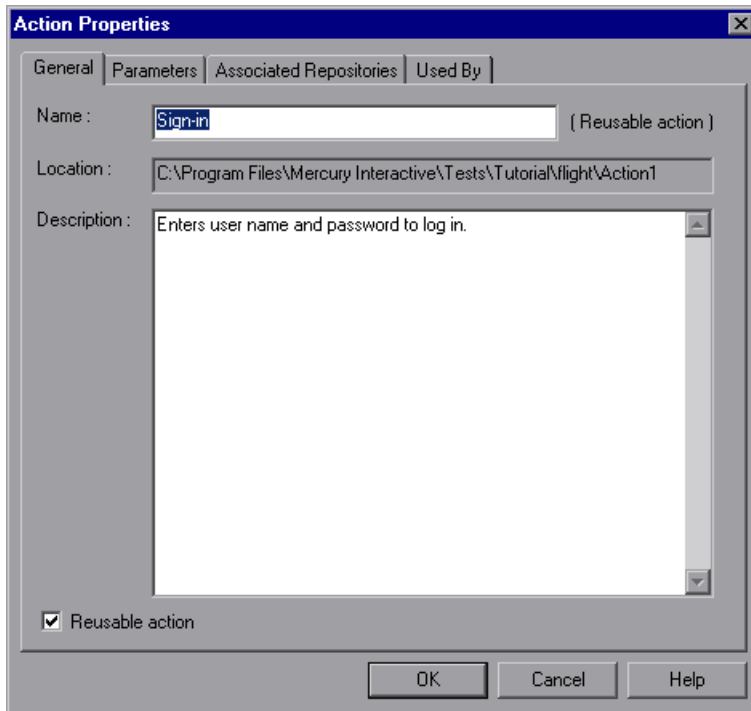
The Action Properties dialog box also enables you to define input and output parameters to be used by the action, and specify the object repositories that are associated with the action. For more information, see “Setting Action Parameters” on page 472 and “Associating Object Repositories with Actions” on page 446.

Note: The following sections describe how to define action properties using the Action Properties dialog box. You can also define actions and action parameters in the Expert View. For more information, see “Understanding Action Syntax in the Expert View” on page 488.

You can open the Action Properties dialog box while working with your test by:

- Right-clicking an action node in the Test Flow pane and selecting **Action Properties**.
- Choosing **Edit > Action > Action Properties** when an action node is highlighted in the Keyword View or displayed in the Expert View.
- Right-clicking an action node in the Keyword View and selecting **Action Properties**.

The Action Properties dialog box always contains the General tab, the Parameters tab (described in “Setting Action Parameters” on page 472), the Associated Repositories tab, and the Used By tab:



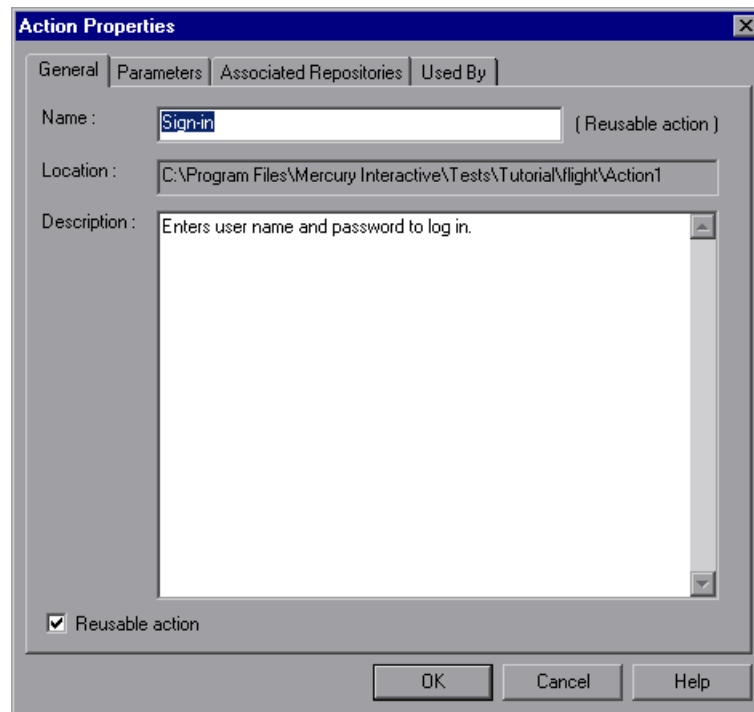
Note: In addition to the tabs shown above, the Action Properties dialog box for a called external action also contains an External Action tab, and the other tabs are read-only. For more information, see “Setting Properties for an External Action” on page 450.

For more information, see:

- “Setting General Action Properties” on page 443
- “Setting Action Parameters” on page 472
- “Associating Object Repositories with Actions” on page 446
- “Viewing a List of the Tests and Actions Using this Action” on page 452
- “Setting Properties for an External Action” on page 450

Setting General Action Properties



You can use the General tab of the Action Properties dialog box to modify the name of an action, add or edit an action’s description, or change the reusability status of the action.



Note: The name of the action and its path are displayed in the tab. If it was defined with a relative path in QuickTest, then the path is displayed as <test name>\<action name>. If the action is a reusable action or an external action, then **Reusable action** or **External Action** is displayed next to the action name.

The General tab includes the following options:

Option	Description
Name	<p>The name of the action. By default, the action name is the internal name provided by QuickTest, such as Action 1. This number is incremented by 1 for each new action that is added to the test.</p> <p>You can rename the action, as needed. The action name must be unique (within the test), cannot begin or end with a space, cannot exceed 1023 characters, and cannot contain the following characters: \\ : * ? " < > % ' ! { }</p>
Location	<p>The folder or Quality Center path where the action is stored.</p>
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 453.</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 436.</p>

Option	Description
Reusable action	<p>Indicates whether the action is a reusable action. By default, this check box is selected. A reusable action can be called multiple times within a test and can be called from other tests. Non-reusable actions can be copied and inserted as independent actions, but cannot be inserted as calls to the original action.</p> <p>When you change this setting, the action icon changes to a non-reusable action icon  or reusable action icon  as appropriate. If the steps of the action were expanded, they collapse after changing a non-reusable action to a reusable action. You can view the steps of the reusable action by selecting the action name in the Test Flow pane.</p>

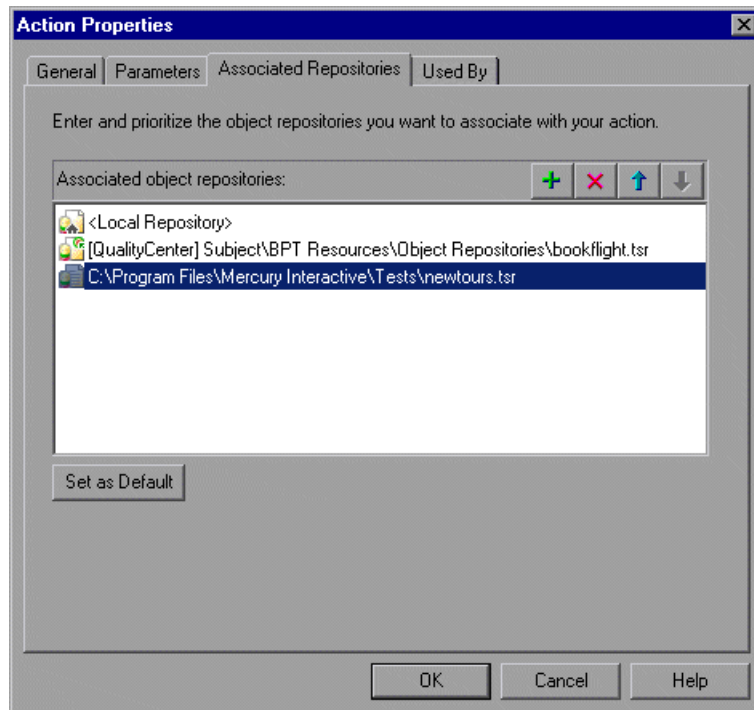
Notes:

- 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 435.
-

Associating Object Repositories with Actions

You can associate object repositories with actions in several ways:


- ▶ You can associate a single action with an object repository by right-clicking the action in the Resources pane and choosing **Associate repository with action** from the context menu. This opens the Open Shared Object Repository dialog box, enabling you to associate an object repository with the selected action.
- ▶ You can use the Associated Repositories tab of the Action Properties dialog box to associate one or more object repositories with the current action. (Right-click an action in the Test Flow pane and select **Action Properties**, or select **Edit > Action > Action Properties** to open the Action Properties dialog box.)



Tip: You can associate shared object repositories with multiple actions simultaneously, using the Associate Repositories dialog box. For more information, see “Managing Shared Object Repository Associations” on page 199.

QuickTest searches these files to locate test object descriptions when identifying objects in your application. You can associate object repositories that are saved in your file system or in a Quality Center project.

Note: QuickTest uses associated object repositories from Quality Center project folders only when you are connected to the corresponding Quality Center project. If you are not connected to the relevant Quality Center project, all associated object repositories that are stored in your Quality Center project are listed as missing in the Missing Resources pane. (QuickTest always lists any associated object repository that cannot be found in the Missing Resources pane.)

In addition, if an object repository cannot be found, QuickTest displays a warning message when you click the Associated Repositories tab in the Action Properties dialog box. QuickTest also adds a question mark to the missing object repository icon  to the left of the missing object repository in the **Associated object repositories** list.

For more information on missing resources, see Chapter 41, “Handling Missing Resources.”

You can associate as many object repositories as needed with an action, and the same object repository can be associated with different actions as needed. You can also set the default object repositories to be associated with all new actions in all tests.





The order of the object repositories in the list determines the order in which QuickTest searches for a test object description. If there are test objects in different object repositories with the same name, object class, and parent hierarchy, QuickTest uses the first one it finds based on the priority order defined in the Associated Repositories tab. The local object repository is always listed first and cannot be moved down the priority list or deleted.

You can enter an associated object repository as a relative path. During the run session, QuickTest searches for the file in the folders listed in the Folders pane of the Options dialog box, in the order in which the folders are listed. For more information, see “Setting Folder Testing Options” on page 1237.

Note: If you want other users or HP products to be able to run an action on other computers, and the action’s associated object repositories are stored in the file system, you can set the file path as a relative path (click the path once to highlight it, and then click it again to enter edit mode). Any users who want to run this action should then specify the drive letter and folder in which QuickTest should search for the relative path in the Folders pane of the Options dialog box (**Tools > Options > Folders** node). For more information, see “Setting Folder Testing Options” on page 1237, and “Using Relative Paths in QuickTest” on page 316.

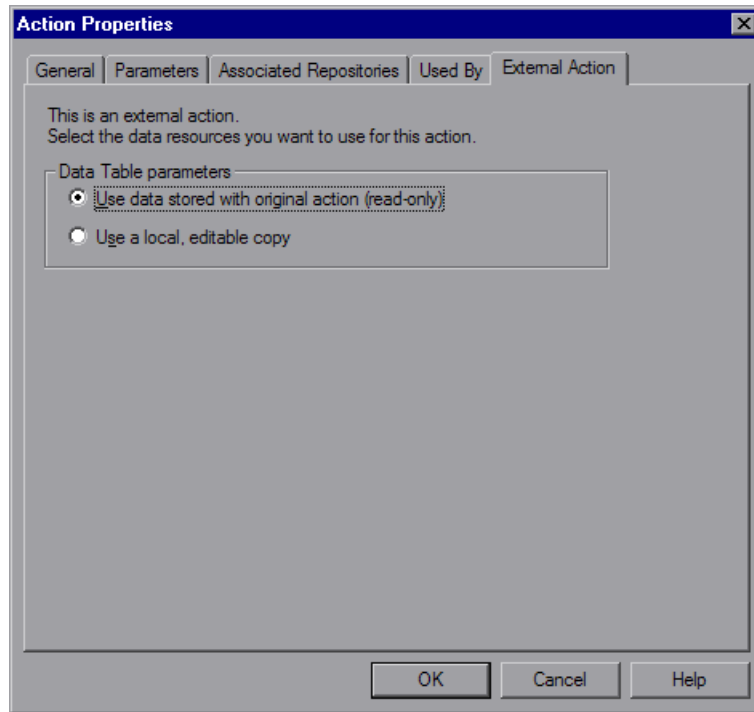
Important: If you are working with the Resources and Dependencies model with Quality Center 10.00, you should store the action’s associated object repositories in the Quality Center Test Resources module and specify an absolute Quality Center path in the Folders pane. For more information, see “Considerations for Working with Relative Paths in Quality Center” on page 1450.

You can add, delete and prioritize the object repositories associated with the action using the following buttons:

Option	Description
	<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: To use the Resources and Dependencies model with Quality Center 10.00, specify an absolute Quality Center path. For more information, see “Considerations for Working with Relative Paths in Quality Center” on page 1450.</p> <p>Tips:</p> <ul style="list-style-type: none"> ➤ To add a Quality Center path when connected to Quality Center, click this button. QuickTest adds [Quality Center], and displays a browse button so that you can locate the Quality Center path. ➤ When not connected to Quality Center, hold the SHIFT key and click this button. QuickTest adds [Quality Center], and you enter the path. You can also type the entire Quality Center path manually. If you do, you must add a space after [Quality Center]. For example: [Quality Center] Subject\ObjectRepositories\flight.tsr.
	<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 from the default for all actions.</p> <p>Caution: If the default object repository is moved or renamed, QuickTest will not be able to locate it. The object repository will be displayed in the Missing Resources pane when new actions or tests are created. For information on resolving missing resources, see Chapter 41, “Handling Missing Resources.”</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.



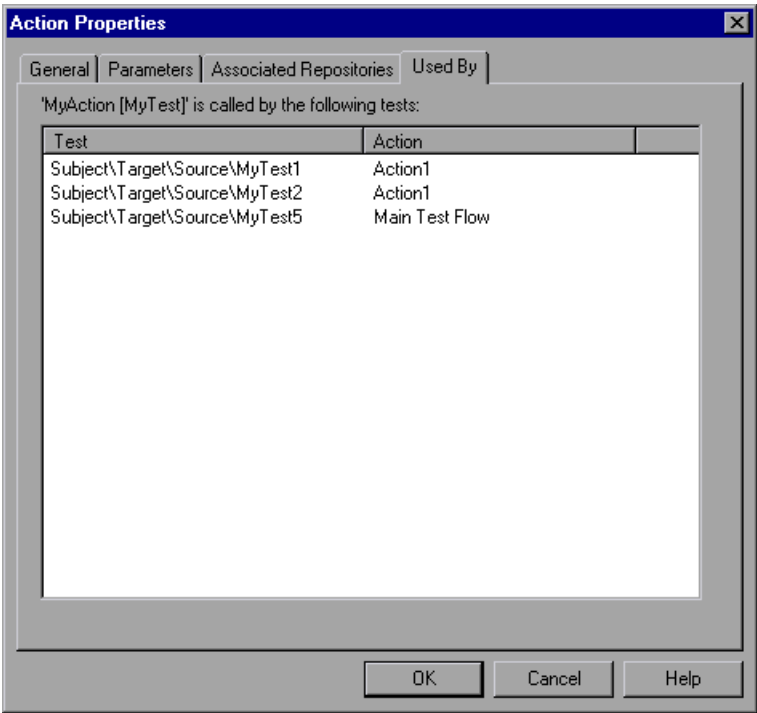
The External Action tab includes the following options:

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). If you select this option, the data is read-only when viewed from the calling test, and all changes to the original action's data sheet apply when the action runs in the calling test. ➤ To use an editable copy of the data in the test's Data Table, select Use a local, editable copy. If you select this option, a copy of the called action's data sheet is added to the calling test's Data Table and is independent of the original action. <p>Changes to the original action's data sheet do not affect the calling test even if you insert another call to this action after the action's data sheet is modified.</p> <p>If the called action has parameterized steps that rely on new information in the original action's data sheet, enter the relevant column names and required data to the action sheet in the calling test manually.</p> <p>Note: When you call an external action, the global data sheet columns and data from the called action's test are always imported as a local, editable copy in the calling test's global data sheet.</p> <p>Changes to the original action's global data sheet do not affect the calling test even if you insert another call to this action after the called action's global data sheet is modified.</p> <p>If the called action has parameterized steps that rely on new information in the global data sheet, enter the relevant column names and required data to the calling test's global data sheet manually.</p>

For more information on calls to external actions, see “Inserting a Call to an Existing Action” on page 468.

Viewing a List of the Tests and Actions Using this Action

If your tests are stored in Quality Center and are using the resources and dependencies model, the Action Properties dialog box displays the Used By tab. This enables you to view a list of the tests and actions that contain calls to this particular action. This is the same list that is displayed in the Dependencies tab of the Test Plan module in Quality Center. For more information, see “Using the Resources and Dependencies Model” on page 1447.



The Used By tab includes the following options:

Option	Description
Test	Indicates the Quality Center path of the test containing a call to this action.
Action	Indicates the internal name of the action containing a call to this action. The internal name is the name that QuickTest applies to an action by default when the action is created, for example, Action 1. The internal name of the action calling this action is displayed even if the calling action was renamed.

Nesting Actions

Sometimes you may want to call an action from within an action. This is called **nesting**. By nesting actions, you can:

- Maintain the modularity of your test.
- Run one or more actions based on the results of a conditional statement.

For example, suppose you have parameterized a step where a user selects one of three membership types as part of a registration process. When the user selects a membership type, the page that opens depends on the membership type selected in the previous page. You can create one action for each type of membership. Then you can use If statements to determine which membership type was selected in a particular iteration of the test and run the appropriate action for that selection.

In the Keyword View, your test might look something like this:

Item	Operation	Value	Documentation
Demographics Info			Call the Demographics Info action.
Membership Preferences			Call the Membership Preferences action.
Membership Preference			
Membership Preference			
MemType	Select	DataTable("memty...	Select radio button <the value of the specified Data Table c...
MemType	GetROProperty	selected	Retrieve the current value of the selected property for the "...
IF Statement		Mem_Type = "paid"	Check whether (Mem_Type = "paid") is true. If so:
Paid_Mem			Call the Paid_Mem action.
ELSE 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 797.

To nest an action within an existing action:

- 1 Highlight the step after which you would like to insert the call to the action.
- 2 Follow the instructions for inserting a call to a new action as described in “Creating New Actions” on page 436, or for inserting a call to a copy of an action or a call to an existing action as described in “Inserting Calls to Existing Actions” on page 464.

Splitting Actions

You can split an action that is stored with your test into two sibling actions or into parent-child nested actions. When you split an action, the second action starts with the step that is selected when you perform the split action operation.


You cannot split an action, and the option is disabled when:

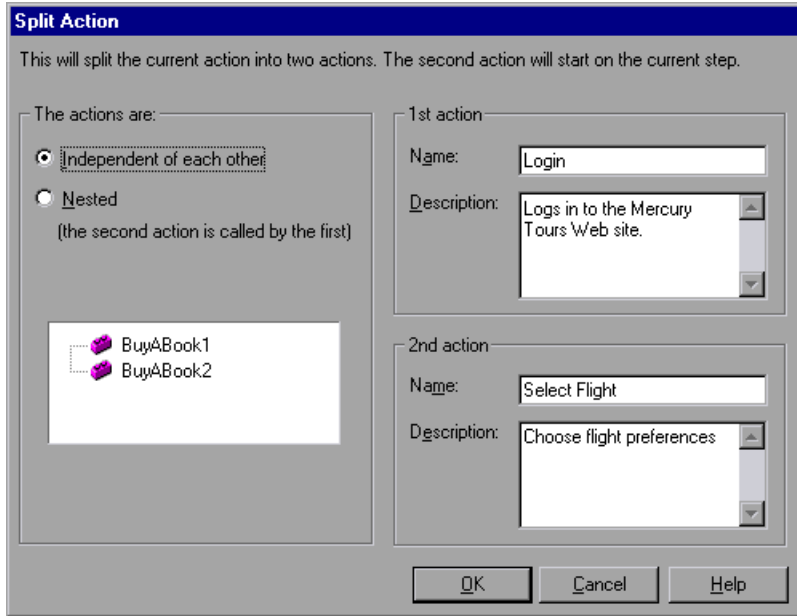
- an external action is selected
- the first step of an action is selected
- you are working with a read-only test
- recording a test
- running a test

When you split an action in your test that uses a local object repository:

- QuickTest makes a copy of the local object repository.
- The two actions have identical local object repositories containing all of the objects that were in the original local object repository.
- If you add objects to one of the split actions, the new objects are added only to the corresponding local object repository.

To split an action:

- 1 Select the step before which you want the new (second) action to begin.
- 2  Select **Edit > Action > Split Action**, click the **Split Action** button, or right-click the step and select **Action > Split**. The Split Action dialog box opens.



- 3 Select one of the following options:
 - **Independent of each other.** Splits the selected action into two sibling actions.
 - **Nested (the second action is called by the first).** Splits the selected action into a parent action whose last step calls the second, child action.
- 4 If you want, modify the name and description of the two actions in the **Name** and **Description** boxes.

Note: If a reusable action is called more than once in a test and you split the action into two independent actions, each call to the action within the test will be followed by a call to the new (reusable) action. If a reusable action is called from another test, however, splitting it may cause the calling test to fail.

Renaming Actions

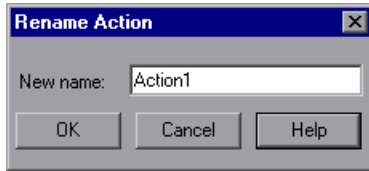
You can rename an action from the Keyword View or Expert View using the Action Properties dialog box or the Rename Action dialog box. When you rename an action, consider how it will affect your test and any tests that call this action. For example, if you rename an action that is used by another test, future run sessions may fail because the test cannot locate the specified action.

Note: If you are working with the Resources and Dependencies model, and the test containing the action you are renaming is stored in the Test Plan module in Quality Center, the internal (default) action name is always displayed in the Used By tab in the Action Properties dialog box. This is true even if you rename the action. For more information, see “Viewing a List of the Tests and Actions Using this Action” on page 452.

Important: You must use the **Rename Action** option in QuickTest if you want to save an action under another name. You cannot change the name of an action directly in the file system or in Quality Center.

To rename an action in the Rename Action dialog box:

- 1 In the Keyword View, select the call to the action you want to rename and select **Edit > Action > Rename Action**. In the Expert View, display the action that you want to rename and select **Edit > Action > Rename Action**. The Rename Action dialog box opens.

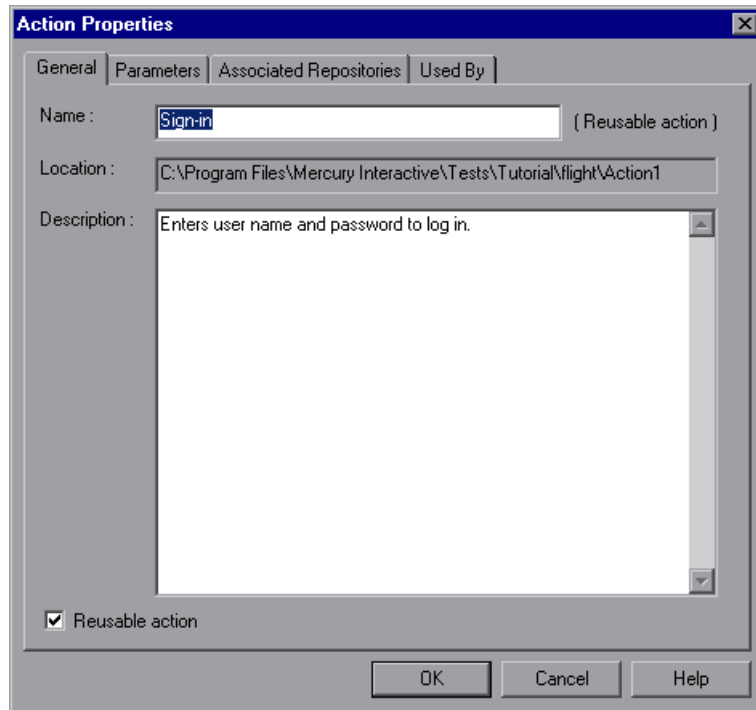


- 2 Enter a new name for the action in the **New name** box. Make sure that action name is unique within the test, does not begin or end with a space, does not exceed 1023 characters, and does not contain the following characters:
\\ : * ? " < > | % ' ! { }
- 3 Click **OK** to save the change.

Tip: You can also press SHIFT + F2 to open the Rename Action dialog box.

To rename an action in the Action Properties dialog box:

- 1 In the Test Flow pane or in the Keyword View, right-click the action and select **Action Properties**. Alternatively, in the Keyword View or in the Expert View, select an action and select **Edit > Action > Action Properties**. The Action Properties dialog box opens.



- 2 Enter a new action name in the **Name** box of the General tab. Each action name within a test must be unique. Make sure that action name is unique within the test, does not begin or end with a space, does not exceed 1023 characters, and does not contain the following characters:
 \ / : * ? " < > | % ' ! { }
- 3 Click **OK** to save the change.

Removing Actions from a Test

If an action is no longer needed, you can remove it from your test. If the action is stored with your test (reusable or non-reusable action) and is called only once in the test, then removing the action deletes it entirely. Alternatively, if the action is stored in another test (external action), or is called more than once in this test (reusable action), removing the action deletes the selected call to the action, without affecting the source action.

The following table illustrates what happens when you delete an action:

Action Type	How deleting the action affects the test:
Reusable action (action stored in the current test)	<ul style="list-style-type: none">➤ If multiple action calls exist in the current test, QuickTest removes only the call to this action. Additional calls to the action in this test remain unchanged. The corresponding action sheet in the Data Table remains unchanged.➤ If this is only call to this action in the current test, QuickTest deletes the action in its entirety, including its corresponding action sheet in the Data Table. <p>Important: Be careful when deleting a local reusable action. If the action is called by other tests, deleting the action may cause the other tests to fail.</p>
Non-reusable action (action stored in the current test)	Deletes the action in its entirety, including its corresponding action sheet in the Data Table.
External action (action stored in a different test)	Removes the call to the action from the current test without affecting the action in the source test. The original action remains stored with the test in which it was created.

Tips for Removing Action Calls and Deleting Actions

- QuickTest provides several locations from which you can remove calls to actions:

Resources pane. Use to simultaneously remove all calls to a specific action.

- If you remove a reusable or non-reusable local action, QuickTest removes all calls to the action in this test and deletes the action in its entirety.
- If you remove an external action, QuickTest removes all calls to the action from the test, but does not affect the source action in any way.

Test Flow pane or the **Keyword View.** Use to remove specific calls to an action.

- If a test contains multiple calls to a single reusable action, and you remove some—but not all—of the calls, QuickTest removes the calls to the action in the specified locations, but does not delete the action itself. This means that the action can continue to be called by this test and by other tests, as needed.
- If you remove all calls to an action, the result is the same as removing the action from the Resources pane. For reusable and non-reusable actions, QuickTest removes all calls to the action in this test and deletes the action in its entirety. For external actions, QuickTest removes all calls to the action from the test, but does not affect the source action in any way.
- When QuickTest deletes an action in its entirety, the corresponding action sheet is removed from the Data Table, but columns related to this action that are located in the Global sheet are not removed.
- If you open a test containing a call to an action you removed, QuickTest informs you that the action is missing. For more information, see “Handling Missing Resources” on page 1179.

To remove a call to an action or delete an entire action:

- 1** In the Resources pane, the Test Flow pane, or the Keyword View:
 - Right-click the action you want to remove and select **Delete**.
 - Select the action you want to remove and press the **Delete** key on your keyboard.
 - Select the action you want to remove and select **Edit > Delete**.
- 2** Click **Yes** in the confirmation message box.

Note: If an action stored in this test is called by other tests, deleting the action in this test may cause other tests to fail.

Creating an Action Template

If you want to include one or more statements in every new action in your test, you can create an action template. For example, if you always enter your name as the author of an action, you can add this comment line to your action template. An action template applies only to actions created on your computer.

To create an action template:

- 1** Create a text file containing the comments, function calls, and other statements that you want to include in your action template. The text file must be in the structure and format used in the Expert View.
- 2** Save the text file as **ActionTemplate.mst** in your **<QuickTest Installation Folder>\dat** folder. All new actions you create contain the script lines from the action template.

Note: Only the file name **ActionTemplate.mst** is recognized as an action template.

16

Working with Advanced Action Features

You can divide your test into actions to streamline the process of testing your application. This chapter covers the advanced use of actions in your test. Using basic action-related features is described in Chapter 15, “Working with Actions.”

This chapter includes:

- About Working with Advanced Action Features on page 464
- Inserting Calls to Existing Actions on page 464
- Setting Action Parameters on page 472
- Using Action Parameters on page 476
- Setting Action Call Properties on page 481
- Sharing Action Information on page 486
- Understanding Action Syntax in the Expert View on page 488
- Exiting an Action on page 491

About Working with Advanced Action Features

Actions help divide your test into logical units, such as the main sections of a Web site, or specific activities that you perform in your application.

A test is comprised of calls to actions. When you create a new test, it contains a call to a single action. By creating tests that call multiple actions, you can design tests that are more modular and efficient.

You can pass information between actions in several ways. You can also specify input parameters for actions, so that steps in an action can use values supplied from elsewhere in the test. You can also output values from actions to be used in steps later in the test, or to be passed back to the application that ran the test. For more information, see “Using Action Parameters” on page 476.

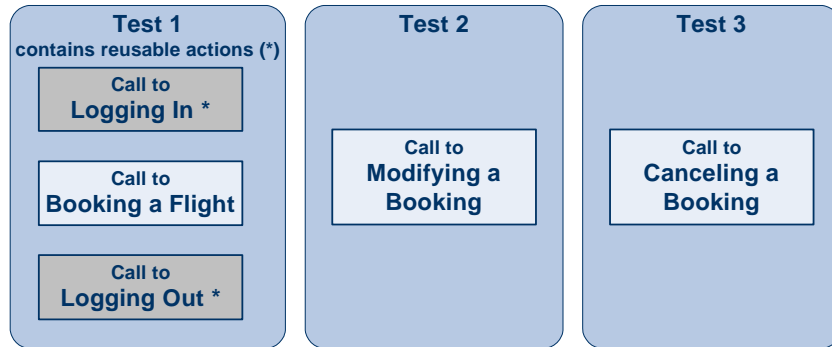
Inserting Calls to Existing Actions

When you plan a suite of tests, you may realize that each test requires some identical activities, such as logging in. Rather than inserting all of the login steps three times in three separate tests and enhancing this part of the script (with checkpoints, parameterization, and programming statements) separately for each test, you can create an action that logs into a flight reservation system and store it with one test. After you are satisfied with the action you created, you can insert calls to the existing action into other tests.

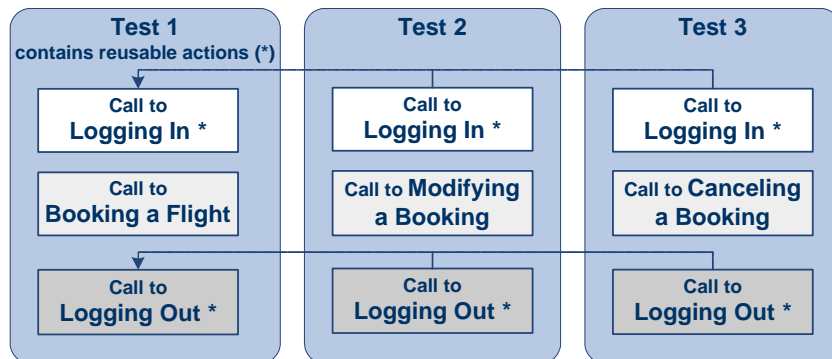
You can insert calls to an existing action by inserting a call to a copy of the action, or by inserting a call to the original action.

For example, suppose you want to create the following three tests for the Mercury Tours site—booking a flight, modifying a reservation, and deleting a reservation. While planning your tests, you realize that for each test, you need to log in and log out of the site, giving a total of five actions for all three tests.

You would initially create three tests with five actions. Test 1 would contain two reusable actions (Logging In and Logging Out). These actions can later be called by Test 2 and Test 3.



You would then finish creating Test 2 and Test 3 by inserting calls to the reusable actions you created in Test 1.



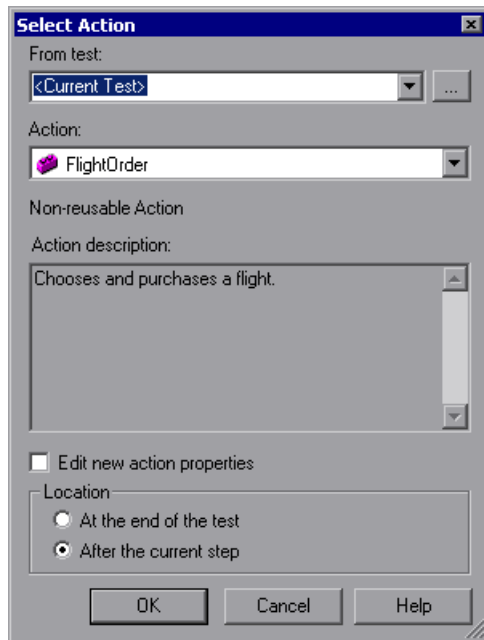
Inserting Calls to Copies of Actions

When you insert a call to a copy of an action into a test, the original action is copied in its entirety, including checkpoints, parameterization, the corresponding action tab in the Data Table, plus any defined action parameters. If the test you are copying has objects in the local object repository, the copied action's local object repository is also copied together with the action.

The action is inserted into the test as an independent, non-reusable action (even if the original action was reusable). After the action is copied into your test, you can add to, delete from, or modify the action just as you would with any other non-reusable action. Any changes you make to this action after you insert it affect only this action, and changes you make to the original action do not affect the copied action.

To create a copy of an action and call the copy in your test:

- 1 In your test, select **Insert > Call to Copy of Action**, right-click an action icon and select **Insert Call to Copy of Action**, or right-click any step and select **Action > Insert Call to Copy**. The Select Action dialog box opens.



- 2 Use the **From test** browse button to find the test containing the action you want to copy. The **Action** box displays all local actions (actions that are stored with the test you selected).

Note: You can enter a Quality Center folder or a relative path in the **From test** box.

- If you enter a relative path, QuickTest searches for the test in the folders listed in the Folders pane of the Options dialog box. For more information, see “Setting Folder Testing Options” on page 1237 and “Using Relative Paths in QuickTest” on page 316.
- If you are working with the Resources and Dependencies model with Quality Center 10.00, specify an absolute Quality Center path. For more information, see “Considerations for Working with Relative Paths in Quality Center” on page 1450.

-
- 3 In the **Action** list, select the action you want to insert. When you select an action, its type (**Non-reusable Action** or **Reusable Action**) and description, if one exists, are displayed. This helps you identify the action you want to copy. For more information on action descriptions see “Setting General Action Properties” on page 443.
 - 4 If you want to modify the copied action’s properties, select the **Edit new action properties** check box. If you select this option, the Action Properties dialog box is displayed when you click **OK**. You can then modify the action properties as described in “Setting Action Call Properties” on page 481.

Note: If you do not select this option, you can modify the action’s properties later by right-clicking the action icon in the Keyword View and selecting **Action Properties**.

- 5 Decide where to insert the call to the copy of the action and select **At the end of the test** or **After the current step**.

For more information on inserting actions within actions, see “Using Action Parameters” on page 476.

Note: If the currently selected step is a reusable action from another test, the call to the copy of the action is added automatically to the end of the test (the **After the current step** option is disabled).

- 6 Click **OK**. The action is inserted into the test as a call to an independent, non-reusable action. You can move your action call to another location in your test by dragging it to the desired location. For more information on moving actions, see “Managing Action Steps” on page 412.

Inserting a Call to an Existing Action

You can insert a call to a reusable action that is stored in your current test (local action), or in any other test (external action). Inserting a call to an existing action is similar to linking to it. You can view the steps of the action in the action view, but you cannot modify them. The called action’s local object repository (if it has one) is also read-only.

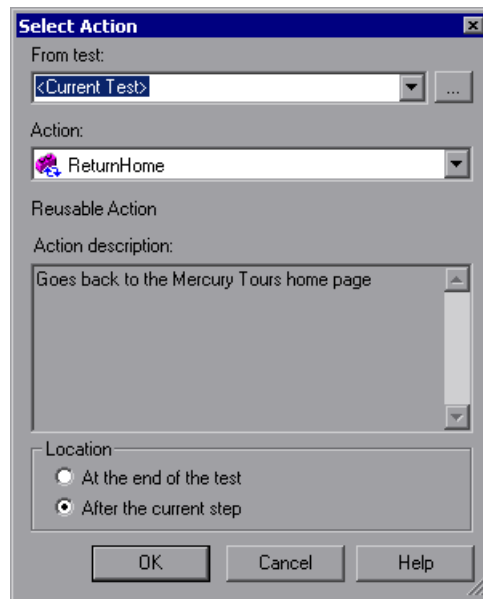
If the called external action has data in the Data Table, however, you can choose whether you want the data from the action’s data sheet to be imported as a local, editable copy, or whether you want to use the (read-only) data from the original action. (Columns and data from the called action’s global data sheet is always imported into the calling test as a local, editable copy.) For more information, see “Setting Properties for an External Action” on page 450.

To modify a called, external action, you must open the test with which the action is stored and make your modifications there. The modifications apply to all tests that call that action. If you chose to use the original action's data when you call an external action, then changes to the original action's data are applied as well.

Tip: You can view the location of the original action in the General tab of the Action Properties dialog box.

To insert a call to an existing action:

- 1 Select **Insert > Call to Existing Action**, right-click an action icon and select **Insert Call to Existing Action**, or right-click any step and select **Action > Insert Call to Existing**. The Select Action dialog box opens.



- 2 Use the **From test** browse button to find the test that contains the action you want to call. The **Action** box displays all reusable actions in the test you selected.

Note: You can enter a Quality Center folder or a relative path in the **From test** box.

- If you enter a relative path, QuickTest searches for the test in the folders listed in the Folders pane of the Options dialog box. For more information, see “Setting Folder Testing Options” on page 1237 and “Using Relative Paths in QuickTest” on page 316.
- If you are working with the Resources and Dependencies model with Quality Center 10.00, specify an absolute Quality Center path. For more information, see “Considerations for Working with Relative Paths in Quality Center” on page 1450.

-
- 3 In the **Action** list, select the action you want to call. When you select an action, its type (**Reusable Action**) and description, if one exists, are displayed. This helps you identify the action you want to call. For more information on action descriptions, see “Setting General Action Properties” on page 443.


Tip: External actions that the test calls are also displayed in the list. If the action you want to call is already called from within the selected test, you can select it from the list of actions. This creates another call to the original action.

Note: QuickTest disables the **Action** list if the selected test does not contain any reusable or external actions.

- 4 Decide where to insert the call to the action and select **At the end of the test** or **After the current step**.

Note: If the currently selected step is a reusable action from another test, the call to the action is added automatically to the end of the test (the **After the current step** is disabled).

For more information on inserting actions within actions, see “Using Action Parameters” on page 476.

- 5 Click **OK**. A call to the action  is inserted into the test flow. You can move your action call to another location in your test by dragging it to the desired location. For more information on moving actions, see “Managing Action Steps” on page 412.

Tip: You can create an additional call to any reusable or external action in your test by pressing CTRL while you drag and drop the action to another location at a parallel (sibling) level within your test.

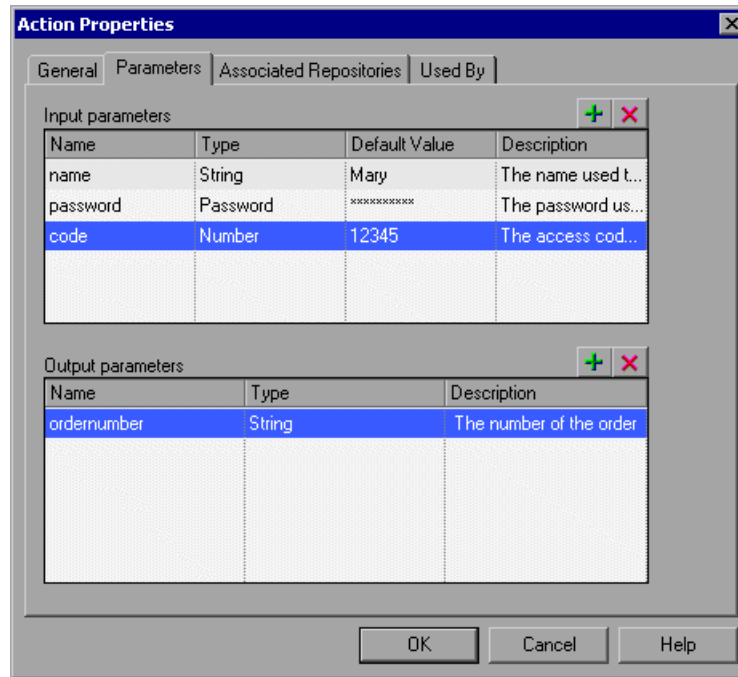
Setting Action Parameters

You can specify input parameters for an action so that steps in the action can use values supplied from elsewhere in the test. Input values for an action parameter can be retrieved from the test (for a top-level action) or from the parameters of the parent action that calls it (for a nested action), or from the output of a previous action call (for a sibling action).

You can specify output parameters for an action, so that it can return values for use later in the test. For example, you can output a parameter value to a parent action so that a later nested action can use the value.

For each input or output action parameter, you define a name (case sensitive), a type, and optionally, a description. You can also specify a default value for each action input parameter, or you can use the default value that QuickTest provides for the parameter value type that you choose. The default value is saved with the action and is used by the action if a value is not defined for a parameter in the action call. You can define, modify, and delete input and output parameters in the Parameters tab of the Action Properties dialog box (**Edit > Action > Action Properties** or right-click an action and select **Action Properties**).

For more information on using action parameters, see “Using Action Parameters” on page 476 and “Guidelines for Working with Action Parameters” on page 479.



The screenshot shows the 'Action Properties' dialog box with the 'Parameters' tab selected. It contains two tables: 'Input parameters' and 'Output parameters'.

Input parameters table:

Name	Type	Default Value	Description
name	String	Mary	The name used t...
password	Password	xxxxxxxxxx	The password us...
code	Number	12345	The access cod...

Output parameters table:

Name	Type	Description
ordernumber	String	The number of the order

Buttons at the bottom: OK, Cancel, Help.

To add a new input or output action parameter:



- 1 Click the **Add** button above the **Input parameters** or **Output parameters** lists to add a new parameter to the appropriate list. A row for the new parameter is added to the relevant list.
- 2 Click in the **Name** box and enter a name for the parameter. (Action parameter names are case sensitive.)

- 3 Select the value type for the parameter in the **Type** box. You can select one of the following types:
 - **String.** A character string enclosed within a pair of quotation marks, for example, "New York". If you enter a value and do not include the quotation marks, QuickTest adds them automatically when the value is inserted in the script during the test run. The default value is an empty string.
 - **Boolean.** A true or false value. If you select a **Boolean** value type, you can click in the **Default Value** column and click the arrow to select a **True** or **False** value. The default value is **True**.
 - **Date.** A date string, for example, 3/2/2005. If you select a **Date** value type, you can click in the **Default Value** column and click the arrow to open a calendar from which you can select a date. The default value is today's date.
 - **Number.** Any number. The default value is 0.
 - **Password.** An encrypted password value. If you select a **Password** value type, the password characters are masked when you enter the password in the **Default Value** field. In the action, however, the value appears encrypted. The default value is an empty string, which also appears as an encrypted value in the actual action.
 - **Any.** A variant value type, which accepts any of the above value types. Note that if you select the **Any** value type, you must specify the value in the format that is required in the location where you intend to use the value. For example, if you intend to use the value later as a string, you must enclose it in quotation marks. When you specify a value of **Any** type, QuickTest checks whether it is a number. If the value is not a number, QuickTest automatically encloses it in quotation marks. If you are editing an existing value, QuickTest automatically encloses it in quotation marks if the previous value had quotation marks. The default value is an empty string.

- 4 If you are defining an input action parameter, click in the **Default Value** box and enter a default value for the parameter. Alternatively, you can leave the default value provided by QuickTest for the parameter value type. The default value is required so that you can run the action without receiving parameter values from elsewhere in the test.
- 5 (Optional) Click in the **Description** box and enter a description of the parameter, for example, the purpose of the parameter in the action. QuickTest displays this description together with the name of the parameter in any dialog box in which you can choose an action parameter, including the Output Options, Parameter Options, and Value Configuration Options dialog boxes.

To modify an existing action parameter:

- 1 Select the parameter you want to modify from the **Input parameters** or **Output parameters** list.
- 2 Modify the values as necessary in the edit boxes of the parameter row.

To delete an existing action parameter:

- 1 Select the parameter you want to delete from the **Input parameters** or **Output parameters** list.
- 2 Click the **Delete** button. The parameter is removed from the list.



Note: When you delete an action parameter, make sure that you also delete any steps that use the action parameter.

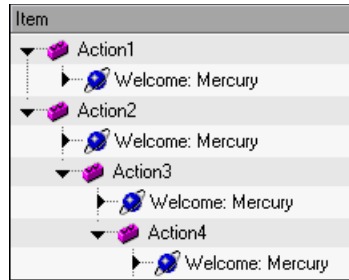
Using Action Parameters

Action parameters enable you to transfer input values from your test to a top-level action, from a parent action to a nested action, or from an action to a sibling action that occurs later in the test. Action parameters also enable you to transfer output values from a step in an action to its parent action, or from a top-level action back to the script or application that ran (called) your test. For example, you can output a value from a step in a nested action and store it in an output action parameter, and then use that value as input in a later step in the calling parent action.

You can use action parameters in any step in your action (including function calls). You define the parameters that an action can receive and the output values that it can return in the Parameters tab of the Action Properties dialog box (**Edit > Action > Action Properties** or right-click an action and select **Action Properties**). You specify the actual values that are provided to these parameters and the locations in which the output values are stored using the Parameter Values tab in the Action Call Properties dialog box (opened by right-clicking an action and choosing **Action Call Properties**).

You can specify input parameters for an action so it can receive input values from elsewhere in the test. Input values for an action parameter can be retrieved from the test (for a top-level action), from the parameters of the parent action that calls it (for a nested action), or from the output of a previous action call (for a sibling action). You can also specify output parameters for an action, so that it can output values for use later in the test, or pass values back to the application that ran (called) the test.

For example, suppose you want to take a value from the external application that runs (calls) your test and use it in an action within your test. In the test below, you would need to pass the input test parameter from the external application through Action2 and Action3 to the required step in Action4.




You would do this as follows:

- 1 Define the input test parameter (**File > Settings > Parameters** node) with the value that you want to use later in the test.
- 2 Define an input action parameter for Action2 (**Edit > Action > Action Properties > Parameters** tab) with the same value type as the input test parameter.
- 3 Parameterize the input action parameter value (**Edit > Action > Action Call Properties > Parameter Values** tab) using the input test parameter value you specified above.
- 4 Define an input action parameter for Action3 (**Edit > Action > Action Properties > Parameters** tab) with the same value type as the input test parameter.
- 5 Parameterize the input action parameter value.
 - Select **Edit > Action > Action Call Properties > Parameter Values** tab and select the input action parameter value you specified for Action2.
 - Use the Parameter utility object to specify the action parameter as the *Parameters* argument for the RunAction statement in the Expert View. For more information, see “Calling Actions with Parameters” on page 489.
- 6 Define an input action parameter for Action4 (**Edit > Action > Action Properties > Parameters** tab) with the same value type as the input test parameter.

7 Parameterize the input action parameter value.

- Select **Edit > Action > Action Call Properties > Parameter Values** tab and select the input action parameter value you specified for Action3.
- Use the Parameter utility object to specify the action parameter as the *Parameters* argument for the RunAction statement in the Expert View. For more information, see “Calling Actions with Parameters” on page 489.

8 Parameterize the value in the required step in Action4.

- Click the parameterization icon  and specify the parameter in the Value Configuration Options dialog box using the input action parameter you specified for Action 4.
- Use the Parameter utility object in the Expert View to specify the value to use for the step. For more information, see “Using Action Parameters in Steps in the Expert View” on page 638.

An action’s parameters are stored with the action and are the same for all calls to that action. If you modify an action parameter’s name, type, or description, and then view the action properties for a call to that same action in a different part of the test, you will see that the action parameter has changed.

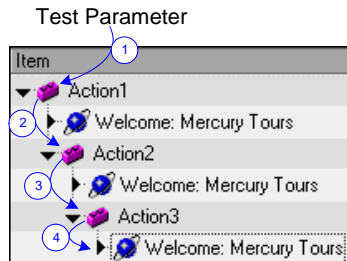
The actual value specified for an input action parameter and the location specified for action output parameter can be different for each call to the action. When you insert a call to a copy of an action, the copy of the action is inserted with the action parameters and action call parameter values that were defined for the action you copied. When you split an action, the action parameters are copied to both actions. The action call values for the second action are taken from the default values of that action’s parameters.

For information on defining action parameters and the values used in action calls, see “Setting Action Parameters” on page 472, and “Setting Action Call Parameter Values” on page 483.

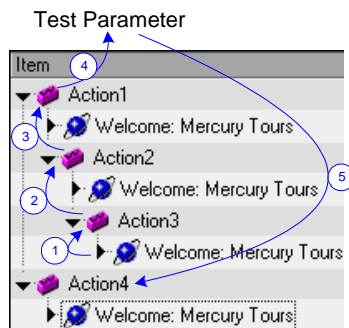
Guidelines for Working with Action Parameters

Consider the following guidelines when working with action parameters:

- Input action parameter values can be used only within the steps of the current action. You can use an action input value from another action (or from the test) only if you pass the value from action to action down the test hierarchy to the action in which you want to use it. For example:
Test -> Action1 -> Action2 -> Action3 -> (Action3) Step 1

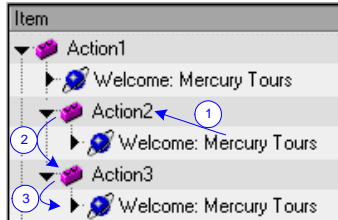


- Output action parameter values can be retrieved from a previous action at the same hierarchical level, from a parent action, or from the current action. You can use an action output value from one action within the step of another action if:
 - You pass the value from action to action up the test hierarchy to the action in which you want to use it. For example:
(Action3) Step 1-> Action3 -> Action2 -> Action1 -> Test -> Action4



In this example, any step in Action 1, Action 2, or Action 3 can potentially use the output value from (Action3) Step 1, even though the example shows that the output value is used by steps in Action4.

- You pass the value from a previous action to the sibling action in which you want to use it. For example:
(Action2) Step 1 -> Action2 -> Action3 -> (Action3) Step 1



In this example, any step in Action 2 or Action 3 can potentially use the output value from (Action2) Step 1, even though the example shows that the output value is used by (Action3) Step 1.

- In subsequent steps of a calling action, you can use any type of action output value as a variable, if the value was retrieved from the called action. For example, if ActionA calls ActionB and specifies MyBVar as the variable in which to store ActionB's output parameter, then steps in ActionA after the call to ActionB can use the MyBVar as a value (just as you would use any other variable).

Setting Action Call Properties

The Action Call Properties dialog box controls the way the action behaves in a specific call to the action. It enables you to specify how many times QuickTest should run the called action (according to the number of rows in the Data Table), and also to specify the initial value for any input action parameters and the location in which you want to store the values of any output action parameters.

Note: The following sections describe how to define action call properties using the Action Call Properties dialog box. You can also define action calls and action call parameters in the Expert View. For more information, see “Understanding Action Syntax in the Expert View” on page 488.

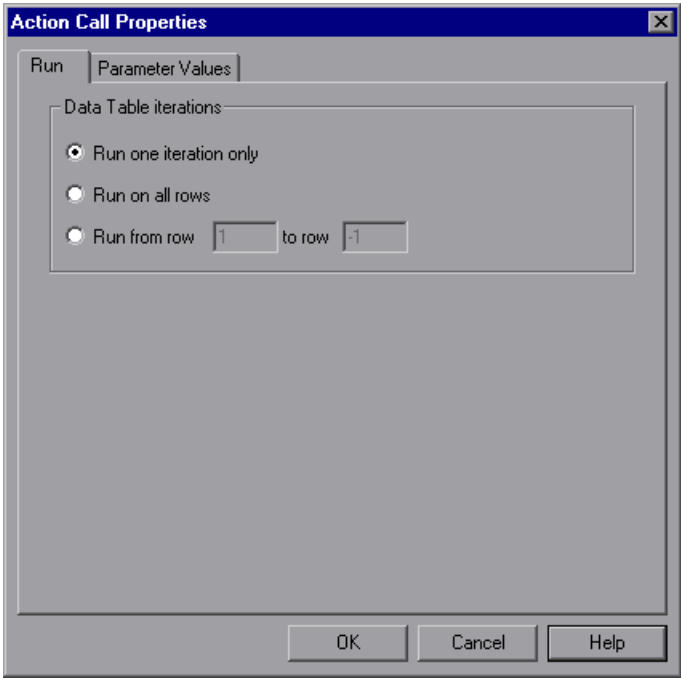
You can open the Action Call Properties dialog box by:

- Right-clicking an action node in the Test Flow pane and selecting **Action Call Properties**.
- Right-clicking an action node in the Keyword View and selecting **Action Call Properties**.
- Choosing **Edit > Action > Action Call Properties** from the Keyword View when an action node is highlighted.

The Action Call Properties dialog box enables you to set options that apply only to a specific action call. The dialog box contains both the Run tab and the Parameter Values tab.

Setting the Run Properties for an Action

You can use the Run tab of the Action Call Properties dialog box to instruct QuickTest to run only one iteration on the called action, to run iterations on all rows in the Data Table, or to run iterations only for a certain row range in the Data Table.



The Run tab includes the following options:

Option	Description
Run one iteration only	Runs the called action only once, using the first row in the action's data sheet.
Run on all rows	Runs the called action with the number of iterations according to the number of rows in the action's Data Table.
Run from row __ to row __	Runs the called action with the number of iterations according to the specified row range.

Notes:

- If you run multiple iterations on an action, the action must begin and end at the same point in the application, so that the application is in the proper location and state to run the next iteration of the action.
 - The Run tab of the Action Call Properties dialog box applies to individual action calls and refers to the rows in the action's data sheet. You can set the Run properties for an entire test (setting iterations for rows on the Global data sheet) from the Run pane in the Test Settings dialog box. For more information, see Chapter 45, "Setting Options for Individual Tests."
-

Setting Action Call Parameter Values

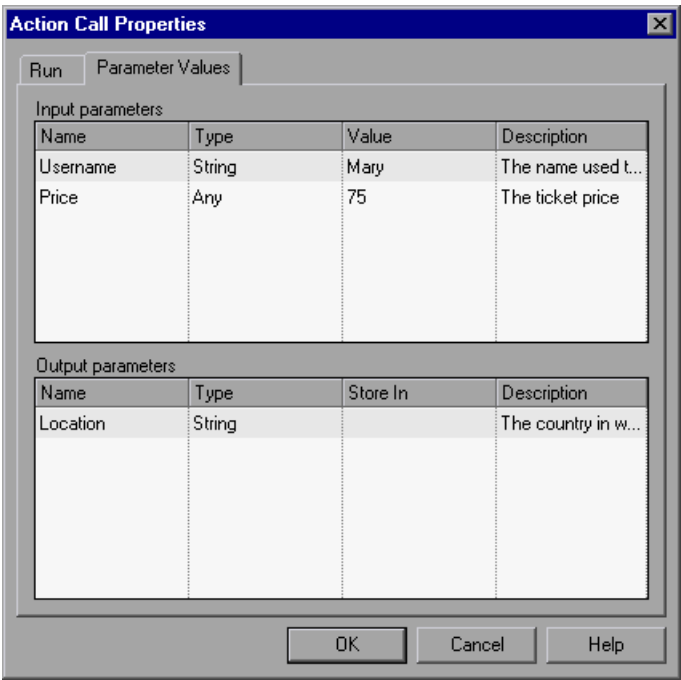
You use the Parameter Values tab of the Action Call Properties dialog box to specify the values of input action parameters used by the called action and to specify the locations in which you want to store output action parameter values. You can also parameterize the value used for a particular input action parameter using any available parameter type.

Note: Specifying input and output parameter values in action calls is optional.

If you do not set a value for an input action parameter, the default value that is specified in the Action Properties dialog box is used.

If you do not define a storage location for an output parameter value, the calling action still has access to the output parameter data generated by the actions it calls. However, specifying a storage location can make your action call statements more readable.


The actual input and output action parameters that an action can receive or return, and their types, are defined in the Action Properties dialog box.



For more information on defining input and output action parameters, see “Setting Action Call Properties” on page 481. For general information on using action parameters, see “Using Action Parameters” on page 476.

To specify the value for an input action parameter:


- 1 In the **Input parameters** area, click in the **Value** box for the parameter and enter a value. For a description of the different options available for each value type, see the definitions included in “Setting Action Parameters” on page 472.

Alternatively, you can click the parameterization button  in the **Value** box to open the Value Configuration Options dialog box in which you can parameterize the value. You can parameterize the value using a test or action parameter (test parameter for a top-level action, or action parameter for a nested or sibling action), Data Table parameter, environment parameter, or random number parameter. For more information, see Chapter 24, “Parameterizing Values.”

- 2 Repeat this procedure for any additional input action parameter values you want to set.

To specify a location in which to store an output action parameter value:

- 1 In the **Output parameters** area, click in the **Store In** box for the parameter and enter a variable name.

Alternatively, you can click the output storage button  in the **Store In** box to open the Storage Location Options dialog box in which you can specify a location for storing the output value. You can select to store the value in a test parameter, the calling action parameter, a Data Table parameter, or an environment parameter. For more information, see “Sharing Action Information” on page 486 and “Storing Return Values and Action Output Parameter Values” on page 794.

- 2 Repeat this procedure for each output action parameter value in the list.

Sharing Action Information

There are several ways to share or pass values from one action to other actions:

- Store values in the output action parameters of a called action and use those values in steps that are performed after the action call within the calling action, or in steps within sibling actions. For more information, see “Storing Values in Test and Action Parameters” on page 673.
- Store values from one action in the global Data Table and use these values as Data Table parameters in other actions. For more information, see “Sharing Values via the Global Data Table” on page 486.
- Set a value from one action as a user-defined environment variable and then use the environment variable in other actions. For more information, see “Sharing Values Using Environment Variables” on page 487.
- Add values to a Dictionary object in one action and retrieve the values in other actions. For more information, see “Sharing Values Using the Dictionary Object” on page 487.

Sharing Values via the Global Data Table

You can share a value that is generated in one action with other actions in your test by storing the value in the global Data Table. Other actions can then use the value in the Data Table as an input parameter. You can store a value in the Data Table by outputting the value to the global Data Table or by using `DataTable`, `Sheet` and `Parameter` objects and methods in the Expert View to add or modify a value.

For example, suppose you are testing a flight reservation application. When a user logs into the application, his or her full name is displayed on the top of the page. Later, when the user chooses to purchase the tickets, the user must enter the name that is listed on his or her credit card. Suppose your test contains three actions—`Login`, `SelectFlight`, and `PurchaseTickets` and the test is set to run multiple iterations with a different login name for each iteration. In the `Login` action, you can create a text output value to store the displayed name of the user. In the `PurchaseTickets` action, you can parameterize the value that is set in the Credit Card Owner edit box using the Data Table column containing the user’s full name.

For more information on output values, see Chapter 25, “Outputting Values.” For more information on parameterization, see Chapter 24, “Parameterizing Values.” For more information on DataTable objects and methods, see Chapter 42, “Working with Data Tables,” and the *HP QuickTest Professional Object Model Reference*.

Sharing Values Using Environment Variables

If you don’t need to run multiple iterations of your test or you want the value you are sharing to stay constant for all iterations, you can use an internal, user-defined environment variable that can be accessed by all local actions in your test.

For example, suppose you want to test that your flight reservation application correctly checks the credit card expiration date that the user enters. The application should request a different credit card if the expiration date that was entered is earlier than the scheduled flight departure date. In the SelectFlight action, you can store the value entered in the departure date edit box in an environment variable. In the PurchaseTickets action, you can compare the value of the expiration date edit box with the value stored in your environment variable.

For more information on environment variables, see Chapter 24, “Parameterizing Values.” For information on the Environment object, see the *HP QuickTest Professional Object Model Reference*.

Sharing Values Using the Dictionary Object

As an alternative to using environment variables to share values between actions as described above, you can use the Dictionary object. The Dictionary object enables you to assign values to variables that are accessible from all actions (local and external) called in the test in which the Dictionary object is created.

To use the Dictionary object, you must first add a reserved object to the registry (in **HKEY_CURRENT_USER\Software\Mercury Interactive\QuickTest Professional\MicTest\ReservedObjects**) with ProgID = "Scripting.Dictionary". For example:

```
HKEY_CURRENT_USER\Software\Mercury  
Interactive\QuickTest Professional\MicTest\ReservedObjects\GlobalDictionary
```

After you have added the reserved Dictionary object to the registry and restarted QuickTest, you can add and remove values to the Dictionary in one action and retrieve the values in another action from the same test.

For example, if you want to access the departure date set in the SelectFlight action from the PurchaseTickets action, you can add the value of the DepartDate WebEdit object to the dictionary in the SelectFlight action as follows:

```
GlobalDictionary.RemoveAll  
GlobalDictionary.Add "DateCheck", DepartDate
```

Then you can retrieve the date from the PurchaseTickets action as follows:

```
Dim CompareDate  
CompareDate=GlobalDictionary("DateCheck")
```

For more information on the Dictionary object, see the VBScript Reference documentation (**Help > QuickTest Professional Help > VBScript Reference > Script Runtime**).

Understanding Action Syntax in the Expert View

An action call in the Expert View can define the action iterations, input parameter values, output parameter storage locations, and an action return values.

Calling Actions Using Basic Syntax

In the Expert View, a call to an action with no parameters is displayed within the calling action with the following basic syntax:

```
RunAction ActionName, IterationQuantity
```

For example, to call the **Select Flight** action and run it one iteration:

```
RunAction "Select Flight", oneIteration
```

For example, to call the **Select Flight** action and run it as many iterations as there are rows in the Data Table:

```
RunAction "Select Flight", allIterations
```

For example, to call the **Select Flight** action and run it four iterations (for the first four rows of the Data Table):

```
RunAction "Select Flight", "1 - 4"
```

Calling Actions with Parameters

If the action you are calling has input and/or output parameters defined, you can also supply the values for the input parameters and the storage location of the output parameters as arguments of the RunAction statement. Input parameters are listed before output parameters.

For an input parameter, you can specify either a fixed value or you can specify the name of another defined parameter (Data Table parameter, environment parameter, or an action input parameter of the calling action) from which the argument should take its value.

For an output parameter, you can specify either a variable in which you want to store the value or the name of a defined parameter (Data Table parameter, environment parameter, or an action output parameter of the calling action).

An action call with parameters has the following syntax:

RunAction *ActionName*, *IterationQuantity*, *Parameters*

For example, suppose you call Action2 from Action1, and Action2 has one input and one output parameter defined.

The following statement supplies a string value of MyValue for the input parameter and stores the resulting value of the output parameter in a variable called MyVariable.

```
RunAction "Action2", oneIteration, "MyValue", MyVariable
```

The following statement uses the value defined for Action1's Axn1_In input action parameter as the value for the input parameter, and stores the resulting value of the output parameter in Action1's Data Table sheet in a column called Column1_out.

```
RunAction "Action2", oneliteration, Parameter("Axn1_In"),  
    DataTable("Column1_out", dtLocalSheet)
```

In the following example, the first statement calls Action2 using its default input parameter value. The second statement uses the value defined for Action2's Axn2_out output action parameter as the value for the call to Action 3's input parameter, and stores the resulting value of the output parameter in Action1's Axn1_out so that the output value is available at the parent action level.

```
RunAction "Action2", oneliteration  
RunAction "Action3", oneliteration, Parameter("Action2","Axn2_out"),  
    Parameter("Axn1_out")
```

Note that the Action2 output parameter is available for use in the call to Action3, even though no storage location is specified in the call to Action2.

Storing Action Return Values

If the action called by the RunAction statement includes an ExitAction statement, the RunAction statement can return the value of the ExitAction's *RetVal* argument. Note that this return value is a return value of the action call itself and is independent of any values returned by specific output parameters of the action call.

To store the return value of an action call, use the syntax:

```
MyRetVal=RunAction (ActionName, IterationQuantity, Parameters)
```

For more information on the Expert View, see Chapter 29, "Working in the Expert View and Function Library Windows." For more information on the RunAction statement, see the *HP QuickTest Professional Object Model Reference*.

Exiting an Action

You can add a line in your script in the Expert View to exit an action before it runs in its entirety. You may want to use this option to return the current value of the action to the value at a specific point in the run or based on the result of a conditional statement. There are four types of exit action statements you can use:

- **ExitAction.** Exits the current action, regardless of its iteration attributes.
- **ExitActionIteration.** Exits the current iteration of the action.
- **ExitRun.** Exits the test, regardless of its iteration attributes.
- **ExitGlobalIteration.** Exits the current global iteration.

You can view the exit action node in the Test Results tree. If your exit action statement returns a value, the value is displayed in the action, iteration, or test summary, as applicable.

For more information on these functions, see the *HP QuickTest Professional Object Model Reference*. For more information on the Test Results, see Chapter 33, “Viewing Run Session Results.”

Part IV

Enhancing Tests

17

Understanding Checkpoints

You can check objects in your application to ensure that they function properly.

This chapter includes:

- About Understanding Checkpoints on page 495
- Adding New Checkpoints to a Test on page 496
- Adding Existing Checkpoints to a Test on page 498
- Understanding Types of Checkpoints on page 501

About Understanding Checkpoints

QuickTest enables you to add checks to your test. A **checkpoint** is a verification point that compares the current value for specified properties with the expected value for those properties. This enables you to identify whether your application is functioning correctly.

When you add a checkpoint, QuickTest inserts a checkpoint step to the current row in the Keyword View and adds a **Check CheckPoint** statement in the Expert View. By default, QuickTest names the checkpoint using the name of the test object on which the checkpoint was created. You can choose to specify a different name for the checkpoint or accept the default name.

When you run the test, QuickTest compares the expected results of the checkpoint to the current results. If the results do not match, the checkpoint fails. You can view the results of the checkpoint in the Test Results window.

Tip: You can also use the **CheckProperty** method and the **CheckItemProperty** method to check specific property or item property values. For more information, see the *HP QuickTest Professional Object Model Reference*.

Note: If you want to retrieve the return value of a checkpoint (a boolean value that indicates whether the checkpoint passed or failed), you must add parentheses around the checkpoint argument in the statement in the Expert View. For example:

```
a = Browser("MyBrowser").Page("MyPage").Check (CheckPoint("MyProperty"))
```

For more information on Expert View syntax, see “Understanding Basic VBScript Syntax” on page 853.

Adding New Checkpoints to a Test

You can add checkpoints while creating or editing your test. It is generally more convenient to define checkpoints after creating the initial test.

Note: You can also add an existing checkpoint to your test. For more information, see “Adding Existing Checkpoints to a Test” on page 498.

To add new checkpoints while editing or recording your test:

Use the commands in the **Insert > Checkpoint** menu, or click the **Insert Checkpoint** button in the toolbar. This displays a menu of checkpoint options that are relevant to the selected step.

To add new checkpoints while editing only:

- Right-click the step where you want to add the checkpoint and select the relevant checkpoint option.
- Select the step where you want to add the checkpoint, select **Insert > Checkpoint**, and then select the relevant checkpoint option.
- Right-click any object in the Active Screen and select the relevant checkpoint option. These options can be used to create checkpoints for any object in the Active Screen (even if the object is not part of any step in the Keyword View).

Notes:

- If you use the Active Screen option, ensure that the Active Screen contains sufficient data for the object you want to check. For more information, see “Setting Active Screen Options” on page 1240.
 - Throughout this guide, procedures for creating checkpoints may be described using only one of the above methods. However, you can choose any of the methods described above.
-

Adding Existing Checkpoints to a Test

QuickTest enables you to reuse the existing checkpoints in your test. When you insert checkpoints into your test, consider which checkpoints can be reused in multiple locations in your test. For example:

- Checkpoints that check generic content or the state of your application may be useful in multiple locations.
- Checkpoints that check the content of a specific area of your application are generally useful in only one particular place in your test.

The following examples illustrate situations in which inserting an existing checkpoint may be useful:

- If each page of your application contains your organization's logo, you can reuse a bitmap checkpoint to verify each occurrence in the application.
- If your application contains multiple edit boxes, you can reuse a checkpoint to confirm the enabled status of these edit boxes throughout your test.

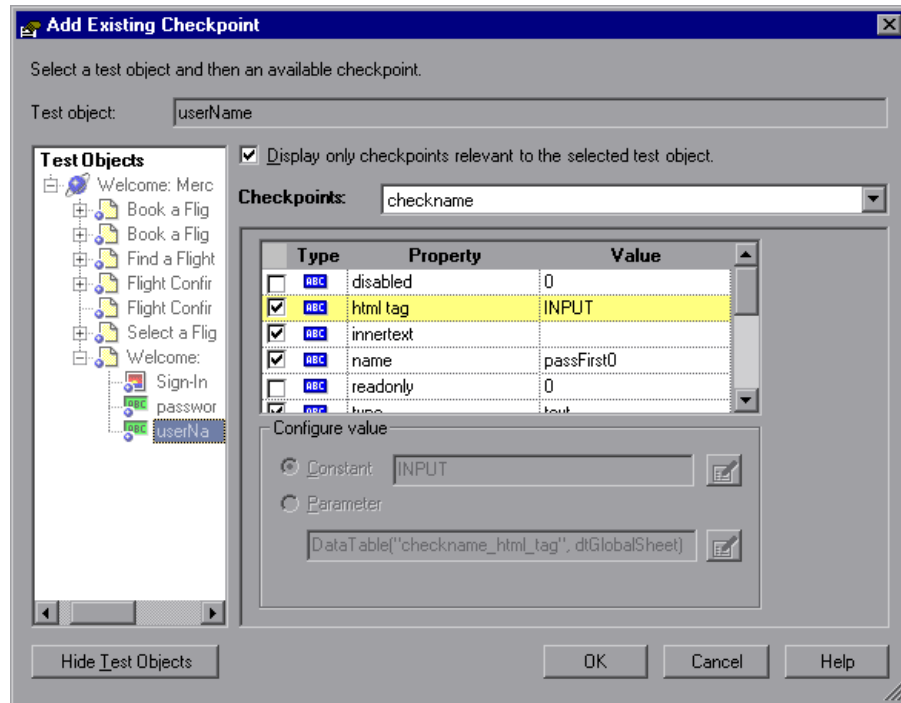
Understanding the Add Existing Checkpoint Dialog Box

You open the Add Existing Checkpoint dialog box by selecting **Insert > Checkpoint > Existing Checkpoint**. This option is available only if at least one of the object repositories associated with the current action (including the local object repository) contains at least one checkpoint.

If a test object step is highlighted in the Keyword View or the cursor is located in a step in the Expert View, the Add Existing Checkpoint dialog box opens with the **TestObjects** tree hidden.

The test object displayed in the **Test object** box is the object from the highlighted step in the Keyword View or the specific object where the cursor is located in the Expert View.

You can display or hide the **TestObjects** tree by clicking the **Show/Hide Test Objects** button.



The Add Existing Checkpoint dialog box contains the following items:

Item	Description
Test object	Specifies the test object for which you are adding a checkpoint.
TestObjects tree	Displays every object in the current test.
Show/Hide Test Objects	Shows or hides the TestObjects tree.
Display only checkpoints relevant to the selected test object	<p>When selected, QuickTest determines which checkpoints from the current action's object repositories are relevant for the selected object (based on the checkpoint type and the properties selected in the checkpoint) and displays only those checkpoints in the Checkpoints list.</p> <p>When using this option, it is recommended to open your application and display the selected object so that QuickTest can accurately determine all of the checkpoints that can apply to that object.</p>
Checkpoints	<p>Lists the checkpoints available for insertion.</p> <p>If the Display only checkpoints relevant to the selected test object option is cleared, this list includes all checkpoints from all object repositories associated with the current action.</p> <p>If the Display only checkpoints relevant to the selected test object option is selected, this list displays only the relevant checkpoints as described above.</p>
Properties Area	Displays the checkpoint properties for the selected checkpoint in read-only format.

To insert an existing checkpoint in your test:

- 1** Display the action in which you want to insert the checkpoint and select the step after which you want to insert the checkpoint.
- 2** Select **Insert > Checkpoint > Existing Checkpoint**. The Add Existing Checkpoint dialog box opens.
- 3** If the **TestObjects** tree is displayed, select the object for which you want to insert a checkpoint. Otherwise proceed to step 4.
- 4** From the **Checkpoints** list, select the checkpoint that you want to insert for the object displayed in the **Test object** box.
- 5** Click **OK**. The checkpoint is inserted after the current step.

Understanding Types of Checkpoints

You can insert the following checkpoint types to check various objects in an application.

- **Standard Checkpoint** checks the property value of an object in your application. The standard checkpoint checks a variety of objects such as buttons, radio buttons, combo boxes, lists, and so forth. For example, you can check that a radio button is activated after it is selected or you can check the value of an edit box.

Standard checkpoints are supported for all add-in environments (see “Supported Checkpoints” on page 504).

For more information on standard checkpoints, see Chapter 18, “Checking Object Property Values Using Standard Checkpoints.”

- **Image Checkpoint** checks the value of an image in your application. For example, you can check that a selected image’s source file is correct.

Note: You create an image checkpoint by inserting a standard checkpoint on an image object.

Image checkpoints are supported for the Web add-in environment (see “Supported Checkpoints” on page 504).

For more information on image checkpoints, see Chapter 18, “Checking Object Property Values Using Standard Checkpoints.”

- **Bitmap Checkpoint** checks an area of your application as a bitmap. For example, suppose you have a Web site that can display a map of a city the user specifies. The map has control keys for zooming. You can record the new map that is displayed after one click on the control key that zooms in the map. Using the bitmap checkpoint, you can check that the map zooms in correctly.

You can create a bitmap checkpoint for any area in your application, including buttons, text boxes, and tables.

Bitmap checkpoints are supported for all add-in environments (see “Supported Checkpoints” on page 504).

For more information on bitmap checkpoints, see Chapter 19, “Checking Bitmaps.”

- **Table Checkpoint** checks information within a table. For example, suppose your application contains a table listing all available flights from New York to San Francisco. You can add a table checkpoint to check that the time of the first flight in the table is correct.

Note: You create a table checkpoint by inserting a standard checkpoint on a table object.

Table checkpoints are supported for Web, ActiveX, Java, Oracle, and .NET Windows Forms environments, as well as other add-in environments (see “Supported Checkpoints” on page 504). Table checkpoints are also supported for some list view objects, such as WinListView and VbListView, as well as other list view objects in add-in environments.

For more information on table checkpoints, see “Checking Tables” on page 529.

- **Text Checkpoint** checks that a text string is displayed in the appropriate place on a Web page or application. For example, suppose a Web page displays the sentence Flight departing from New York to San Francisco. You can create a text checkpoint that checks that the words "New York" are displayed between "Flight departing from" and "to San Francisco".

Text checkpoints are supported for most add-in environments (see “Supported Checkpoints” on page 504).

For more information on text checkpoints, see Chapter 21, “Checking Text.”

- **Text Area Checkpoint** checks that a text string is displayed within a defined area in a Windows-based application, according to specified criteria. For example, suppose your Visual Basic application has a button that says View Doc <Num>, where <Num> is replaced by the four digit code entered in a form elsewhere in the application. You can create a text area checkpoint to confirm that the number displayed on the button is the same as the number entered in the form.

Text area checkpoints are supported for all Windows-based environments, such as Standard Windows, Visual Basic, and ActiveX add-in environments (see “Supported Checkpoints” on page 504). Text area checkpoints are also supported for some other add-in environments, such as Java.

For more information on text area checkpoints, see Chapter 21, “Checking Text.”

- **Accessibility Checkpoint** identifies areas of your Web site that may not conform to the World Wide Web Consortium (W3C) Web Content Accessibility Guidelines. For example, guideline 1.1 of the W3C Web Content Accessibility Guidelines requires you to provide a text equivalent for every non-text element. You can add an **Alt** property check to check whether objects that require the **Alt** property under this guideline, do in fact have this tag.

Accessibility checkpoints are supported for the Web add-in environment (see “Supported Checkpoints” on page 504).

For more information on accessibility checkpoints, see the section on testing Web objects in the *HP QuickTest Professional Add-ins Guide*.

- **Page Checkpoint** checks the characteristics of a Web page. For example, you can check how long a Web page takes to load or whether a Web page contains broken links.

Note: You create a page checkpoint by inserting a standard checkpoint on a page object.

Page checkpoints are supported for the Web add-in environment (see “Supported Checkpoints” on page 504).

For more information on page checkpoints, see the section on testing Web objects in the *HP QuickTest Professional Add-ins Guide*.

- **Database Checkpoint** checks the contents of a database accessed by your application. For example, you can use a database checkpoint to check the contents of a database containing flight information for your Web site.

Database checkpoints are supported for all add-in environments (see “Supported Checkpoints” on page 504).

For more information on database checkpoints, see Chapter 22, “Checking Databases.”

- **XML Checkpoint** checks the data content of XML documents in XML files or XML documents in Web pages and frames. For more information on XML checkpoints, see Chapter 23, “Checking XML.”

The **XML Checkpoint (Web Page/Frame)** option is supported for the Web add-in environment. The **XML Checkpoint** option is supported for all add-in environments (see “Supported Checkpoints” on page 504).

Supported Checkpoints

QuickTest add-ins help you to create and run tests and components on applications in a variety of development environments. For information about using checkpoints for each add-in environment installed with QuickTest Professional, see “Supported Checkpoints” on page 1546.

18

Checking Object Property Values Using Standard Checkpoints

By adding standard checkpoints to your tests, you can compare object property values in your application with the expected values.

This chapter includes:

- About Checking Object Property Values on page 505
- Creating Standard Checkpoints on page 506
- Understanding the Checkpoint Properties Dialog Box on page 508
- Understanding the Image Checkpoint Properties Dialog Box on page 512
- Modifying Checkpoints on page 514

About Checking Object Property Values

You can check the object property values in your application using standard checkpoints. Standard checkpoints compare the expected values of object properties to the object's current values during a run session. You can create standard checkpoints for all supported testing environments (as long as the appropriate add-ins are loaded).

You use standard checkpoints to perform checks on images, tables, Web page properties, and other objects within your application.

Creating Standard Checkpoints

You can check that a specified object in your application has the property values you expect, by adding a standard checkpoint step to your test while recording or editing the test. To set the options for a standard checkpoint, you use the Checkpoint Properties dialog box.

To add a standard checkpoint while recording:



- 1** While in a recording session, select **Insert > Checkpoint > Standard Checkpoint**, or click the **Insert Checkpoint or Output Value** toolbar button.

The QuickTest window is hidden, and the pointer changes into a pointing hand.

Tips:

- If the window on which you want to spy is partially hidden by another window, hold the pointing hand over the partially hidden window for a few seconds until it comes to the foreground. Then point to and click the required object.
 - You can hold the left CTRL key to change the pointing hand to a standard pointer, and then change the window focus or perform operations, such as right-clicking or moving the pointer over an object to display a context menu.
 - If the window containing the object you want to select is minimized, you can display it by holding the left CTRL key, right-clicking the application in the Windows task bar, and choosing **Restore** from the context menu.
-

- 2** Click the object you want to check. The Object Selection - Checkpoint Properties dialog box opens.
- 3** Select the item you want to check from the displayed object tree. The tree item name corresponds to the object's class.
- 4** Click **OK**. The Checkpoint Properties dialog box opens.

- 5 Specify the settings for the checkpoint. For more information, see “Understanding the Checkpoint Properties Dialog Box” on page 508.
- 6 Click **OK** to close the dialog box. A checkpoint statement is added for the selected object in the Keyword View and Expert View.

To add a standard checkpoint while editing:

- 1 Perform one of the following:
 - Right-click the step on which you want to perform a checkpoint and select **Insert Standard Checkpoint**.
 - Select the step where you want to add the checkpoint and select **Insert > Checkpoint > Standard Checkpoint**.
 - Right-click any object in the Active Screen and select **Insert Standard Checkpoint**.

The Checkpoint Properties dialog box opens.

Note: To add a standard checkpoint while editing, one of the following must be true:

- Active Screen information exists for the step. For more information, see “Working with the Active Screen” on page 376.
- The object for which you want to create a checkpoint is currently displayed in the application.

Depending on the object and environment, it may be necessary for both of these conditions to be true. For more information, see the chapter for your environment in the *HP QuickTest Professional Add-ins Guide*.

- 2 Specify the settings for the checkpoint. For more information, see “Understanding the Checkpoint Properties Dialog Box” on page 508.
- 3 Click **OK** to close the dialog box. A checkpoint statement is added for the selected object in the Keyword View and Expert View.

Understanding the Checkpoint Properties Dialog Box

In the Checkpoint Properties dialog box, you can specify which properties of the object to check, and edit the values of these properties. While the specific elements vary slightly depending on the type of object you are checking, the Checkpoint Properties dialog box generally includes the following basic elements:

The 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 512
Page checkpoint properties	The section on checking Web pages in the <i>HP QuickTest Professional Add-ins Guide</i>
Table checkpoint properties	“Understanding the Table Checkpoint Properties Dialog Box” on page 535

Identifying the Checkpoint






The top part of the dialog box displays information on the checkpoint:

Information	Description
Name	<p>The name of the checkpoint. By default, the checkpoint name is the same as the name of the test object on which the checkpoint was created. You can specify a different name for the checkpoint or accept the default name.</p> <p>If you rename the checkpoint, make sure that the name:</p> <ul style="list-style-type: none"> ➤ is unique ➤ does not begin or end with a space ➤ does not contain " (double quotation mark) ➤ does not contain the following character combinations: <pre>:= @@</pre>
Class	The type of object. In this example, the WebEdit class indicates that the object is an edit box.

Information	Description
Find in Repository button 	Displays the checkpoint in its object repository. Note: This option is available only when editing an existing checkpoint. It is not available when creating a new checkpoint. When available, it is located to the right of the Name box.

Selecting the Object Property to Check

The properties for the object are listed in the Properties pane of the dialog box. The pane includes the properties, their values, and their types:

Pane Element	Description
Check box	For each object class, QuickTest recommends default property checks. You can accept the default checks or modify them accordingly. To check a property, select the corresponding check box. To exclude a property check, clear the corresponding check box.
Type	The  icon indicates that the value of the property is currently a constant. The  icon indicates that the value of the property is currently a test or action parameter. The  icon indicates that the value of the property is currently a Data Table parameter. The  icon indicates that the value of the property is currently an environment variable parameter. The  icon indicates that the value of the property is currently a random number parameter.
Property	The name of the property.
Value	The expected value of the property. For more information on modifying the value of a property, see “Setting Values in the Configure Value Area” on page 757.

Editing the Expected Value of an Object Property

In the **Configure value** area, you can define the expected value of the property to check as a **Constant** or **Parameter**. For information on modifying property values, see “Setting Values in the Configure Value Area” on page 757.

Setting the Checkpoint Timeout Option

Checkpoint timeout. Specifies the time interval (in seconds) during which QuickTest attempts to perform the checkpoint successfully. QuickTest continues to perform the checkpoint until it passes or until the timeout occurs. If the checkpoint does not pass before the timeout occurs, the checkpoint fails.

For example, suppose it takes some time for an object to achieve an expected state. Increasing the checkpoint timeout value in this case can help ensure that the object has sufficient time to achieve that state, enabling the checkpoint to pass (if the data matches) before the maximum timeout is reached.

If you specify a checkpoint timeout other than **0**, and the checkpoint fails, the Test Results window displays information on the checkpoint timeout.

Inserting the Checkpoint in Your Test

The **Insert statement** option specifies when to perform the checkpoint in the test.

- Select **Before current step** if you want to check the value of the object property before the highlighted step is performed.
- Select **After current step** if you want to check the value of the property after the highlighted step is performed.

Note: The **Insert statement** option is not available when adding a checkpoint during recording or when modifying an existing object checkpoint. It is available only when adding a new checkpoint to an existing test while editing it.

Understanding the Image Checkpoint Properties Dialog Box

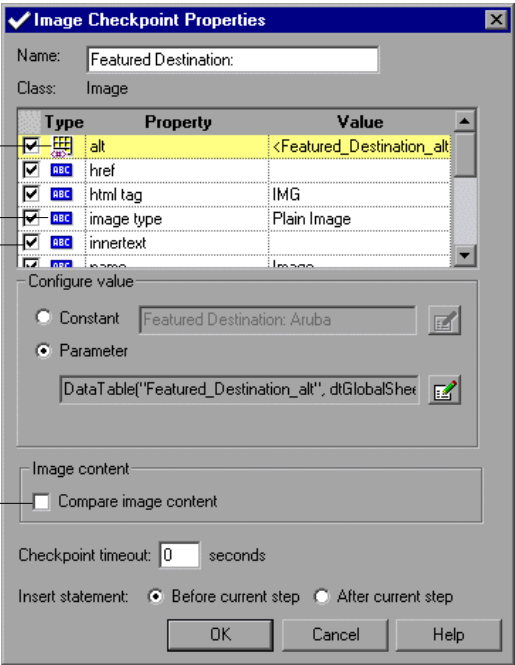
Image checkpoints enable you to check the properties of a Web image. In the Image Checkpoint Properties dialog box, you can specify which properties of the image to check and edit the values of those properties. This dialog box is similar to the standard Checkpoint Properties dialog box, except that it contains the **Compare image content** option. This option enables you to compare the expected image source file with the actual image source file.

This icon indicates that the value of the property to check is a Data Table parameter.

The ABC icon indicates that the value of the property to check is a constant.


The selected check box indicates that this property will be checked.

Instructs QuickTest to compare the expected image with the graphic of the actual image.



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:</p> <ul style="list-style-type: none"> ➤ is unique ➤ does not begin or end with a space ➤ does not contain " (double quotation mark) ➤ does not contain the following character combinations: <pre>:= @@</pre>
Class	The type of object. This is always Image .
Find in Repository button  (Located to the right of the Name box)	<p>Displays the checkpoint in its object repository.</p> <p>Note: This option is available only when editing an existing checkpoint. It is not available when creating a new checkpoint.</p>

Selecting the Image Property to Check

The default properties for the image are listed in the Properties pane of the dialog box. This pane includes the properties, their values, and their types. It is identical to the Properties pane in the Checkpoint Properties dialog box for standard checkpoints. For more information, see “Selecting the Object Property to Check” on page 510.

Editing the Expected Value of an Image Property

The middle part of the Image Checkpoint Properties dialog box contains the following:

- **Configure value.** Enables you to define the expected value of the property as a **Constant** or **Parameter**. For information on modifying property values, see “Setting Values in the Configure Value Area” on page 757.
- **Compare image content.** Compares the expected image source file with the graphic of the actual image source file. If the expected and actual images are different, QuickTest displays them both in the Test Results. If the images are identical, only one graphic is displayed.

Setting General Image Checkpoint Options

The bottom part of the Image Checkpoint Properties dialog box contains the **Checkpoint timeout** and **Insert statement** options. These options are identical to those in the Checkpoint Properties dialog box for standard checkpoints. For more information, see “Setting the Checkpoint Timeout Option” on page 511 and “Inserting the Checkpoint in Your Test” on page 511.

Modifying Checkpoints

You can modify the settings of existing checkpoints.

To modify a checkpoint:

- 1** In the Keyword View or Expert View, right-click the checkpoint that you want to modify and select **Checkpoint Properties**. Alternatively, select the step containing the checkpoint and select **Edit > Step Properties > Checkpoint Properties**. The relevant checkpoint dialog box opens.
- 2** Modify the properties and click **OK**. For more information, see “Understanding the Checkpoint Properties Dialog Box” on page 508.

19

Checking Bitmaps

QuickTest enables you to check that the visible parts of your application are displayed correctly by comparing bitmaps of objects in your application to bitmaps captured previously and stored with the test.

This chapter includes:

- About Checking Bitmaps on page 515
- Fine-Tuning the Bitmap Comparison on page 516
- Creating and Modifying Bitmap Checkpoints on page 518
- The Bitmap Checkpoint Properties Dialog Box on page 522

About Checking Bitmaps

You can check an area of an application as a bitmap. You can check an entire object or any area within an object. For example, suppose you have a Web site that can display a map of a city the user specifies. The map has control keys for zooming. You can record the new map that is displayed after one click on the control key that zooms in the map. Using the bitmap checkpoint, you can check that the map zooms in correctly.

You can create bitmap checkpoints for all supported testing environments (as long as the appropriate add-ins are loaded).

The results of bitmap checkpoints may be affected by factors such as operating system, screen resolution, and color settings.

When you create a bitmap checkpoint, QuickTest captures the **visible** part of the specified object as a bitmap (QuickTest does not capture any part that is scrolled off the screen, or hidden by another object, for example), and inserts a checkpoint in the test.

When you run the test, QuickTest captures a bitmap of the actual object in the application and compares this bitmap (or a selected area within it) with the bitmap stored in the checkpoint.

If there are differences, QuickTest saves the bitmap of the actual object and displays it next to the expected bitmap in the details pane of the Test Results window. In the Test Results window you can also view a bitmap that reflects the difference between the two bitmaps, to assist you in identifying the nature of the discrepancy. You can configure QuickTest not to save the bitmaps in the test results, or to save them even if the checkpoint passes (**Tools > Options > Run > Screen Capture** pane). For more information on test results of a checkpoint, see “Viewing Checkpoint Results” on page 1028.

Fine-Tuning the Bitmap Comparison

When running a bitmap checkpoint, QuickTest compares the area that you are checking in the application with the bitmap stored in the checkpoint, pixel by pixel. By default, if any pixels are different, the checkpoint fails. The Bitmap Checkpoint Properties dialog box (described on page 522) provides options for fine-tuning the bitmap comparison.

You can adjust the comparison to enable the checkpoint to pass even if the bitmaps are not identical by setting the **RGB tolerance** and **Pixel tolerance** options described below.

In addition, QuickTest enables you to use **custom comparers** for bitmap checkpoints. A custom comparer is a COM object that you or a third party developed to run the bitmap comparison in the checkpoint according to a more specific algorithm. If one or more custom comparers are installed and registered on the QuickTest computer, the Bitmap Checkpoint Properties dialog box includes a **Comparer** option.

This option enables you to select the QuickTest default comparer or a custom comparer that performs the bitmap comparison according to your testing requirements. For an example on when it can be useful to create a custom comparer, see “Use-Case Scenario: Handling Images Whose Location in the Application Changes” on page 1577. For more information on developing custom comparers, see Appendix D, “Bitmap Checkpoint Customization.”

If you select a custom comparer, some of the options in the Bitmap Checkpoint Properties dialog box are different. For more information, see “The Bitmap Checkpoint Properties Dialog Box” on page 522.

Bitmap Checkpoint Tolerance Options

- **RGB tolerance.** The RGB (Red, Green, Blue) tolerance determines the percent by which the RGB values of the pixels in the actual bitmap can differ from those of the expected bitmap and allow the checkpoint to pass. (The RGB tolerance option is limited to bitmaps with a color depth of 24 bits.)

For example, a bitmap checkpoint on identical bitmaps could fail if different display drivers are used when you create your checkpoint and when you run your test. Suppose one display driver displays the color white as RGB (255, 255, 255) and another driver displays the color white as RGB (231, 231, 231). The difference between these two values is about 9.4%. By setting the **RGB tolerance** to 10%, your checkpoint will pass when running your test with either of these drivers.

Note: QuickTest applies the RGB tolerance settings when comparing each pixel in the actual and expected bitmaps. The Red, Green, and Blue values for each pixel are compared separately. If any of the values differs more than the tolerance allows, the pixel fails the comparison.

- **Pixel tolerance.** The pixel tolerance determines the number or percentage of pixels in the actual bitmap that can differ from those in the expected bitmap and allow the checkpoint to pass.

For example, suppose the expected bitmap has 4000 pixels. If you define the pixel tolerance to be 50 and select the **Pixels** radio button, up to 50 pixels in the actual bitmap can be different from those in the expected bitmap and the checkpoint passes. If you define the pixel tolerance to be 5 and select the **Percent** radio button, up to 200 pixels (5 percent of 4000) in the actual bitmap can be different from those in the expected bitmap and the checkpoint passes.

Using both RGB and Pixel Tolerances

If you define both RGB and pixel tolerances, the RGB tolerance is calculated first. The pixel tolerance then defines the maximum number of pixels that can fail the RGB criteria and allow the checkpoint to pass.

For example, suppose you define an RGB tolerance of 10 percent and a pixel tolerance of 5 percent for a bitmap that has 4000 pixels.

For the checkpoint to pass, each pixel in the actual bitmap must have RGB values that are no greater than or no less than 10 percent of the RGB values of the expected bitmap. If that criterion fails, QuickTest checks that the number of pixels that failed are less than 200. If that criterion passes, the checkpoint passes.

Creating and Modifying Bitmap Checkpoints

You insert a bitmap checkpoint while recording or editing a test. You can also modify an existing bitmap checkpoint.

Bitmap checkpoints can capture only the visible part of an object. Therefore, confirm that the object to capture is always fully visible on the screen before a bitmap checkpoint step is performed. One way to do this is to insert a `MakeVisible` statement (for relevant environments) prior to your bitmap checkpoint step. For more information on the `MakeVisible` method, see the *QuickTest Object Model Reference*.

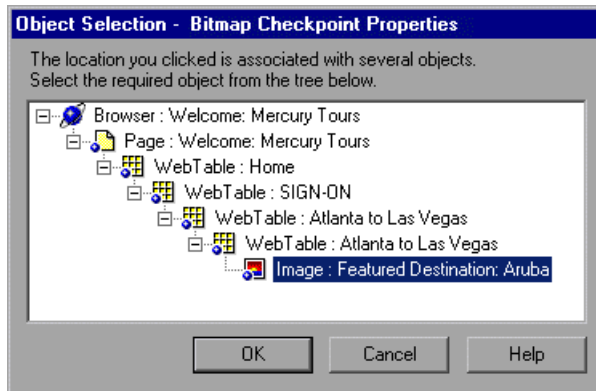
To create a bitmap checkpoint while recording:



- 1 Select **Insert > Checkpoint > Bitmap Checkpoint**, or click the **Insert Checkpoint or Output Value** button and select **Bitmap Checkpoint**.

The QuickTest window is hidden, and the pointer turns into a pointing hand. For more information about using the pointing hand feature, see “Tips for Using the Pointing Hand” on page 521.

- 2 Click an object to check in your application. If the location you click is associated with more than one object, the Object Selection - Bitmap Checkpoint Properties dialog box opens.



- 3 Select an object from the tree on which to create the bitmap checkpoint.

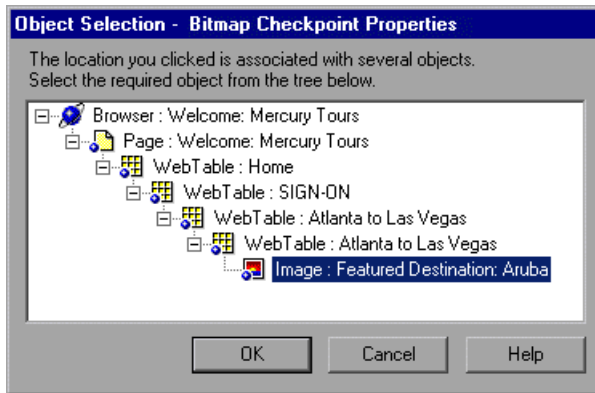
Tip: If you want to create a bitmap checkpoint that contains multiple objects, you should select the highest level object that includes all the objects to include in the bitmap checkpoint.

- 4 Click **OK**. The Bitmap Checkpoint Properties dialog box opens. Create the Bitmap checkpoint using the options in the dialog box. For more information, see “The Bitmap Checkpoint Properties Dialog Box” on page 522.

To create a bitmap checkpoint while editing:



- 1** Make sure the **Active Screen** button is selected.
- 2** Click the step in the Keyword View for which you want to add a checkpoint. The Active Screen displays the area of the application corresponding to the highlighted step.
- 3** Right-click an object in the Active Screen and select **Insert Bitmap Checkpoint**. If the location you click is associated with more than one object, the Object Selection - Bitmap Checkpoint Properties dialog box opens.



- 4** Select an object from the tree on which to create a bitmap checkpoint.

Tips:

- Ensure that the object you select is completely visible. If another application is overlapping the object, it is also captured.
 - To create a bitmap checkpoint that contains multiple objects, select the highest level object that includes all the objects to include in the bitmap checkpoint.
-

- 5 Click **OK**. The Bitmap Checkpoint Properties dialog box opens. Create the Bitmap checkpoint using the options in the dialog box. For more information, see “The Bitmap Checkpoint Properties Dialog Box” on page 522.

To modify a bitmap checkpoint:



- 1 Select the step containing the checkpoint and select **Edit > Step Properties > Checkpoint Properties**, or select the **Value** cell in the step and click the **Checkpoint Properties** button. Alternatively, in the Keyword View or Expert View, right-click the checkpoint that you want to modify and select **Checkpoint Properties**.
- 2 The Bitmap Checkpoint Properties dialog box opens and displays the object or area you saved with the checkpoint. Modify the Bitmap checkpoint using the options in the dialog box. For more information, see “The Bitmap Checkpoint Properties Dialog Box” on page 522.

Tips for Using the Pointing Hand

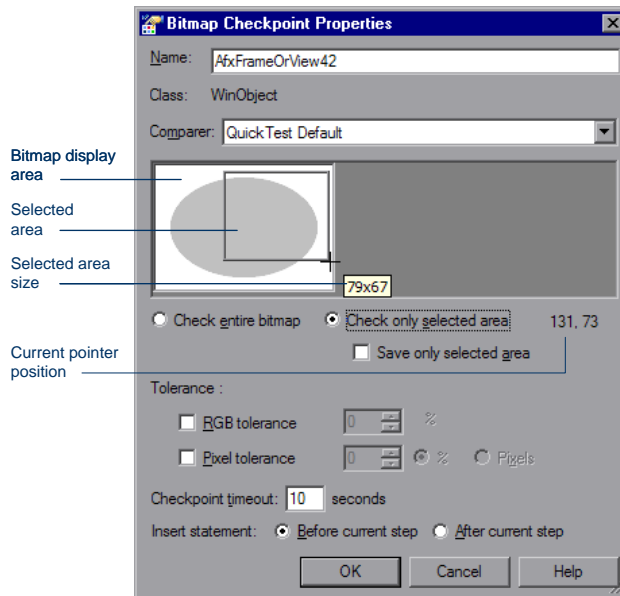
- You can hold the left CTRL key to change the pointing hand to a standard pointer. You can then change the window focus or perform operations in QuickTest or in your application, such as right-clicking, using the scroll bars, or moving the pointer over an object to display a context menu.
- If the window containing the object you want to select is partially hidden by another window, hold the pointing hand over the partially hidden window for a few seconds until it comes to the foreground. Then point to and click the required object. You can configure the length of time required to bring a window into the foreground using the General pane of the Options dialog box.
- If the window containing the object you want to select is fully hidden by another window, or if a dialog box is hidden behind a window, press the left CTRL key and arrange the windows as needed.
- If the window containing the object you want to select is minimized, you can display it by holding the left CTRL key, right-clicking the application in the Windows task bar, and choosing **Restore** from the context menu.

- If the object you want to select can be displayed only by performing an event (such as right-clicking or moving the pointer over an object to display a context menu), hold the left CTRL key. The pointing hand temporarily turns into a standard pointer and you can perform the event. When the object you want to select is displayed, release the left CTRL key. The pointer becomes a pointing hand again.

The Bitmap Checkpoint Properties Dialog Box

Description	Enables you to create or modify a bitmap checkpoint.
How to Access	See: “Creating and Modifying Bitmap Checkpoints” on page 518
Learn More	Conceptual overview: “About Checking Bitmaps” on page 515

The following image is an example. It shows the options that are available when selecting an area to check in the bitmap, in a new checkpoint being created while editing a test.



Bitmap Checkpoint Properties Dialog Box Details

This dialog box includes several groups of options, described in the following sections:

- “Descriptive Information” on page 524
- “Options for Selecting the Area to Check” on page 525
- “Tolerance Options” on page 526
- “Checkpoint Timeout and Statement Location Options” on page 527

Descriptive Information

The descriptive information is displayed in the top part of the Checkpoint Properties dialog box.

- **Name.** By default, the checkpoint name is the same as the name of the test object on which the checkpoint was created. Accept the name that QuickTest assigns to the checkpoint or specify another name for it.

If you rename the checkpoint, make sure that the name:

- is unique
- does not begin or end with a space
- does not contain " (double quotation mark)
- does not contain the following character combinations:
 - :=
 - @ @
- **Class.** The type of test object on which the checkpoint was created. (Read-only)
- **Comparer.** Enables you to select the comparer for QuickTest to use to run the checkpoint. You can select the QuickTest default comparer or a custom comparer. If you select a custom comparer, some of the options in this dialog box are different. For more information, see “Custom Comparer Options in the Bitmap Checkpoint Properties Dialog Box” on page 527.

This option is available only if any custom comparers are installed and registered on the QuickTest computer. Otherwise, the QuickTest default comparer is used. For more information, see “Fine-Tuning the Bitmap Comparison” on page 516.

- **Bitmap display area.** Displays a bitmap of the object you selected.



- **Find in Repository.** To view the checkpoint in its repository, click the **Find in Repository** button located to the right of the **Name** box.

This option is not available when creating a new checkpoint. It is available only when editing an existing checkpoint.

Options for Selecting the Area to Check

The options for selecting the area to check are displayed beneath the bitmap display area.

- **Check entire bitmap / Check only selected area.** Enables you to specify whether the checkpoint compares the entire bitmap or only a specific area of the bitmap. If you select **Check only selected area**, the cursor turns into a crosshairs pointer when you hover over the bitmap display area. Use the crosshairs pointer to draw a rectangle specifying the area that you want to select. To remove the rectangle, click again.

While the crosshairs pointer is visible, QuickTest displays the coordinates of the pointer's current position beneath the bottom-right corner of the bitmap display area. As you draw the rectangle using the crosshairs, QuickTest displays a tooltip with the current selected area size near the crosshairs pointer.

If you define the checkpoint to compare only a specific area of the bitmap, the selected area is highlighted also in the actual and expected bitmaps displayed in the Test Results window.

- **Save only selected area.** Enables you to save only the selected area of the object with your test (to save disk space). The bitmap stored in the checkpoint is cropped when you click **OK**. The Test Results window displays only the selected area of the bitmap.

This option is available only after you select **Check only selected area** and draw the rectangle that specifies the area.

Note: If you select the **Save only selected area** check box, you can later modify the checkpoint by selecting a smaller area within the selected area, but you cannot return the bitmap to its former size. The **Update Run Mode** option (**Automation > Update Run Mode**) only updates the saved area of the bitmap. It does not update the original, full size object. To include more of the object in the checkpoint, create a new checkpoint.

Tolerance Options

The tolerance options are displayed beneath the options for selecting the area to check.

- **RGB tolerance.** Enables you to define the percent by which the RGB values of the pixels in the actual bitmap can differ from those of the expected bitmap and allow the checkpoint to pass.

Select the check box and modify the percentage manually or by using the up and down arrows. For more information, see “Fine-Tuning the Bitmap Comparison” on page 516.

This option is limited to bitmaps with a color depth of 24 bits.

- **Pixel tolerance.** Enables you to define the number or percentage of pixels in the actual bitmap that can differ from those in the expected bitmap and allow the checkpoint to pass.

Select the check box, select either the **Percent** or **Pixels** radio button, and modify the value manually or by using the up and down arrows. If you switch between the **Percent** and **Pixels** radio buttons after entering the tolerance value, the value is recalculated based on your selection. (100% is the total number of pixels in the expected bitmap or selected area.)

For more information, see “Fine-Tuning the Bitmap Comparison” on page 516.

Checkpoint Timeout and Statement Location Options

The checkpoint timeout and statement location options are displayed in the bottom part of the Checkpoint Properties dialog box.

- **Checkpoint timeout.** Enables you to define the time interval (in seconds) during which QuickTest attempts to perform the checkpoint successfully. QuickTest continues to perform the checkpoint until it passes or until the timeout occurs. If the checkpoint does not pass before the timeout occurs, the checkpoint fails.

For example, suppose it takes some time for an object to achieve an expected state. Increasing the checkpoint timeout value in this case can help ensure that the object has sufficient time to achieve that state, enabling the checkpoint to pass (if the data matches) before the maximum timeout is reached.

If you specify a checkpoint timeout other than **0**, and the checkpoint fails, the Test Results window displays information on the checkpoint timeout.

- **Before current step / After current step.** Enables you to insert the bitmap checkpoint before or after the highlighted step.

This option is available only when creating a checkpoint while editing a test.

Custom Comparer Options in the Bitmap Checkpoint Properties Dialog Box

In the Bitmap Checkpoint Properties dialog box, if you select a custom comparer to run the bitmap comparison, the options for selecting an area of the bitmap and for setting tolerance levels are not available.

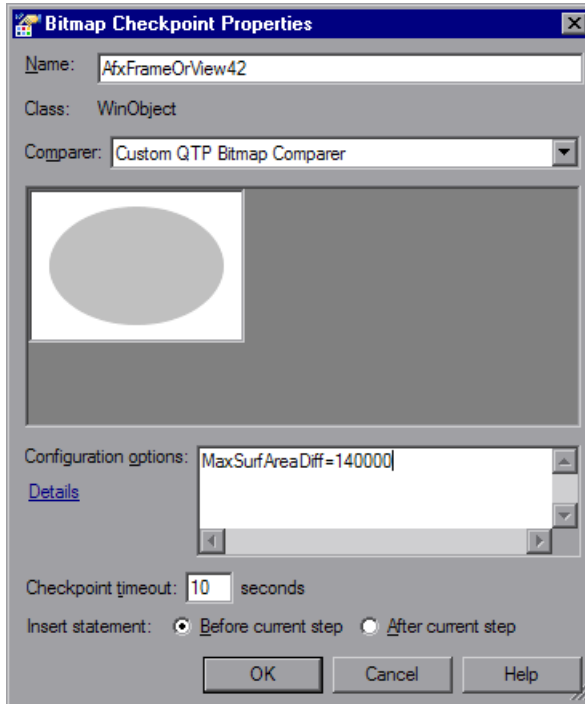
Instead, the following options are available (as supported by the custom comparer):

- **Configuration options.** Enables you to provide input (in string format) to the custom comparer, for any configuration options it supports. For example, you might be able to specify tolerance levels, an acceptable deviation in size or location of the bitmap, and so on.

By default, this box displays a configuration string provided by the custom comparer (if available).

- **Details.** Opens help information provided by the custom comparer (if available). This help can include instructions for providing configuration input to the comparer, information about the algorithm that the custom comparer uses to compare the bitmaps, an explanation about when to use this custom comparer, and so on.

Below is an image of the Bitmap Checkpoint Properties dialog box with a custom comparer selected:



20

Checking Tables

You can add table checkpoints to check the content of tables displayed in your application.

This chapter includes:

- About Checking Tables on page 529
- Creating a Table Checkpoint on page 530
- Understanding the Table Checkpoint Properties Dialog Box on page 535
- Checking Table Content on page 536
- Checking Table Properties on page 546
- Modifying a Table Checkpoint on page 548

About Checking Tables

By adding table checkpoints to your test, you can check the content of tables displayed in your application. For example, you can check that a specified value is displayed in a certain cell. For some environments, you can also check the properties of the table object. For example, you can check that a table has the expected number of rows and columns.

When you run the test, the table checkpoint compares the actual data to the expected data, as defined in the checkpoint. If the results match, the checkpoint passes. You can view the results of the checkpoint in the Test Results window. For more information, see Chapter 33, “Viewing Run Session Results.”

Table checkpoints are supported for table objects in a variety of add-in environments, such as Web, ActiveX, and Java. Table checkpoints are also supported for some list view objects, such as WinListView and VbListView.

Creating a Table Checkpoint

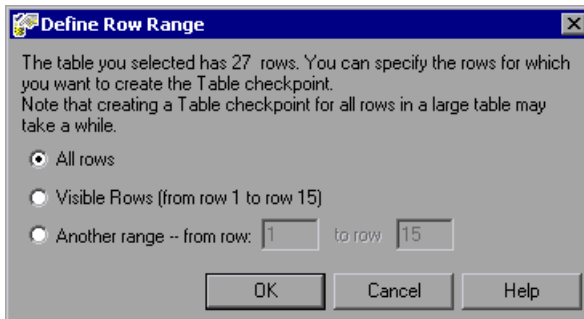
You can add a table checkpoint while recording or editing your test. To add a table checkpoint, you use the Table Checkpoint Properties dialog box.

To add a table checkpoint while recording:



- 1** Select **Insert > Checkpoint > Standard Checkpoint** or click the **Insert Checkpoint or Output Value** button. The QuickTest window is hidden, and the pointer changes to a pointing hand. For more information on using the pointing hand feature, see “Tips for Using the Pointing Hand” on page 532.
- 2** Click the table you want to check. The Object Selection - Checkpoint Properties dialog box opens.
- 3** Select a table item from the displayed object tree and click **OK**. If the Table Checkpoint Properties dialog box opens, skip to step 4.

Otherwise, for certain objects in certain environments, the Define Row Range dialog box opens.



Select the range of rows you want to include in your checkpoint. You can include:

- **All rows.** Includes all of the rows in the table. Note that capturing all of the data for large table or list view objects may take some time.
- **Visible Rows (from row X to row Y).** Includes only the rows visible on the screen. Note that this option may not be available for some environments or object types.
- **Another range -- from row _ to row _.** You can specify any row range in the table.

Click **OK**. The Define Row Range dialog box closes, and the Table Checkpoint Properties dialog box displays the rows you specified (above the grid area).

- 4** In the Table Checkpoint Properties dialog box, specify the settings for the checkpoint. For more information, see “Understanding the Table Checkpoint Properties Dialog Box” on page 535.

Note: For some environments, the Table Checkpoint Properties dialog box contains two tabs: Table Content and Properties. For other environments, the Table Checkpoint Properties dialog box displays only the options available in the Table Content tab, but does not contain any tabs.

- 5** Click **OK** to close the dialog box. A checkpoint statement is added for the selected object in the Keyword View and Expert View.

Tips for Using the Pointing Hand

- You can hold the left CTRL key to change the pointing hand to a standard pointer. You can then change the window focus or perform operations in QuickTest or in your application, such as right-clicking, using the scroll bars, or moving the pointer over an object to display a context menu.
- If the window containing the object you want to select is partially hidden by another window, hold the pointing hand over the partially hidden window for a few seconds until it comes to the foreground. Then point to and click the required object. You can configure the length of time required to bring a window into the foreground using the General pane of the Options dialog box.
- If the window containing the object you want to select is fully hidden by another window, or if a dialog box is hidden behind a window, press the left CTRL key and arrange the windows as needed.
- If the window containing the object you want to select is minimized, you can display it by holding the left CTRL key, right-clicking the application in the Windows task bar, and choosing **Restore** from the context menu.
- If the object you want to select can be displayed only by performing an event (such as right-clicking or moving the pointer over an object to display a context menu), hold the left CTRL key. The pointing hand temporarily turns into a standard pointer and you can perform the event. When the object you want to select is displayed, release the left CTRL key. The pointer becomes a pointing hand again.

To add a table checkpoint while editing:

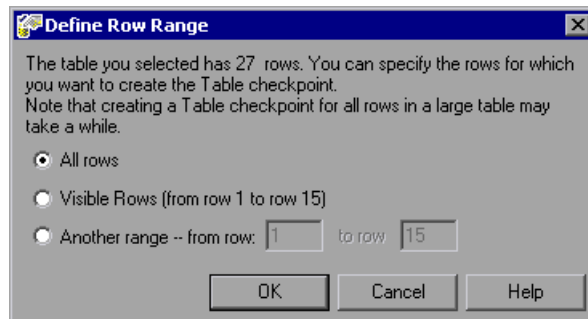
- 1 Depending on whether the object on which you want to perform a check is already in a step, do one of the following:
 - If you already recorded a step on the object you want to check, right-click the step and select **Insert Standard Checkpoint**. Alternatively, select the step and select **Insert > Checkpoint > Standard Checkpoint**.
 - If you have not recorded a step on the object you want to check, make sure the **Active Screen** button is selected and the Active Screen is visible. Click a step in your test where you want to add a checkpoint. The Active Screen displays the Web page or application screen corresponding to the highlighted step. Right-click the table in the Active Screen and select **Insert Standard Checkpoint**. The Object Selection - Checkpoint Properties dialog box opens. Select a table item from the displayed object tree and click **OK**.



Note: In some environments, you must have the table open in your application to insert a checkpoint on it.

- 2 If the Table Checkpoint Properties dialog box opens, skip to step 3.

Otherwise, for certain objects in certain environments, the Define Row Range dialog box opens.



Select the range of rows you want to include in your checkpoint. You can include:

- **All rows.** Includes all of the rows in the table. Note that capturing all of the data for large table or list view objects may take some time.
- **Visible Rows (from row X to row Y).** Includes only the rows visible on the screen. Note that this option may not be available for some environments or object types.
- **Another range -- from row X to row Y.** You can specify any row range between 1 and the number of rows listed in the table.

Click **OK**. The Define Row Range dialog box closes, and the Table Checkpoint Properties dialog box displays the rows you specified (above the grid area).

- 3** In the Table Checkpoint Properties dialog box, specify the settings for the checkpoint. For more information, see “Understanding the Table Checkpoint Properties Dialog Box” on page 535.

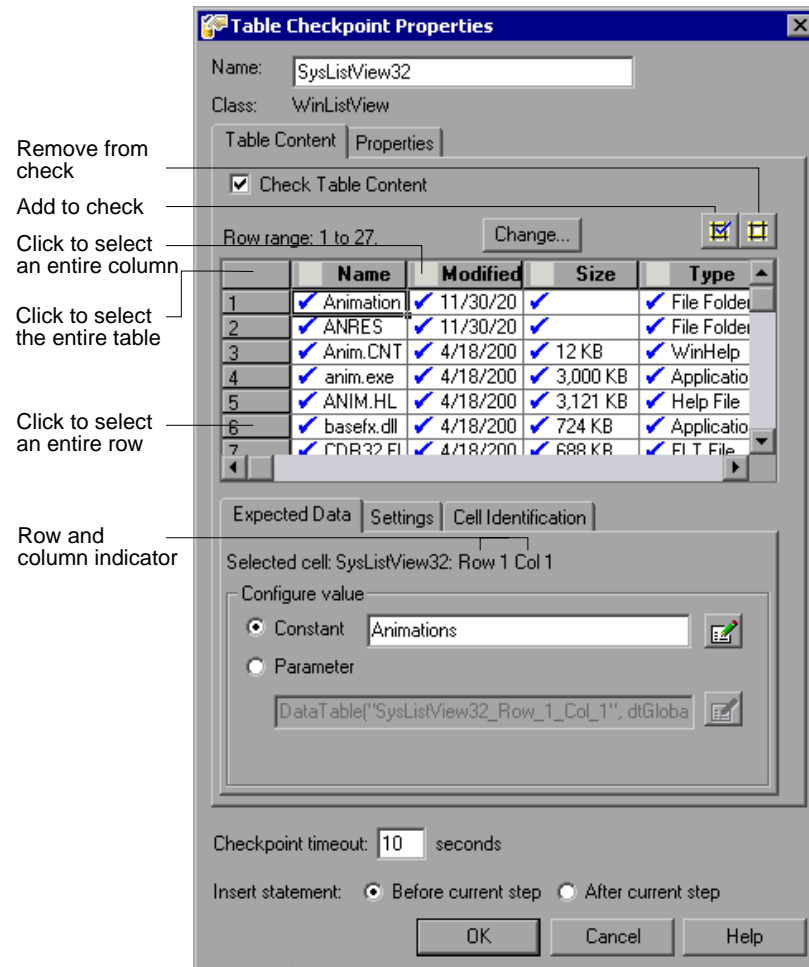
Note: For some environments, the Table Checkpoint Properties dialog box contains two tabs: Table Content and Properties. For other environments, the Table Checkpoint Properties dialog box displays only the options available in the Table Content tab, but does not contain any tabs.

- 4** Click **OK** to close the dialog box. A checkpoint statement is added for the selected object.

Understanding the Table Checkpoint Properties Dialog Box

The Table Checkpoint Properties dialog box enables you to specify which cell contents of your table to check and which verification method and type to use. You can also edit or parameterize the expected data for the cells included in the check.

For some environments, the Table Checkpoint Properties Dialog Box also enables you to check the properties of the object (using the Properties tab), in addition to checking the content (using the Table Content tab).



Note: Some of the options shown in this example are available only in certain environments and only for certain objects.

For information on the options in the Table Content tab (or the entire dialog box if no tab is displayed), see the sections below. For information on the options in the Properties tab, see “Checking Table Properties” on page 546.

Checking Table Content

The Table Checkpoint Properties dialog box enables you to check table content.

Note: If the Table Checkpoint Properties dialog box contains tabs, you use the Table Content tab to check table content.

You can:

- understand and set general table checkpoint options
- specify which cells to check
- specify the expected data (Expected Data tab)
- specify the value type criteria (Settings tab)
- specify how QuickTest should locate the cells to be checked (Cell Identification tab)


Understanding and Setting General Table Checkpoint Options

This section describes the general settings and options displayed in the Table Checkpoint Properties dialog box. Most of the options described in this section are available regardless of whether the Table Checkpoint Properties dialog box contains tabs.

Descriptive Information

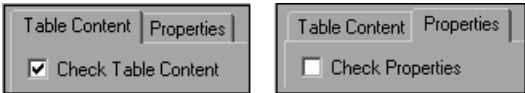
The top part of the Table Checkpoint Properties dialog box contains the following options:



Name	<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:</p> <ul style="list-style-type: none"> ➤ is unique ➤ does not begin or end with a space ➤ does not contain " (double quotation mark) ➤ does not contain the following character combinations: <p style="margin-left: 20px;">:= @@</p>
Class	<p>Specifies the type of object (read-only). This may be a table-type object or a list view-type object.</p>
Find in Repository button 	<p>Displays the checkpoint in its object repository.</p> <p>Note: This option is available only when editing an existing checkpoint. It is not available when creating a new checkpoint.</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:

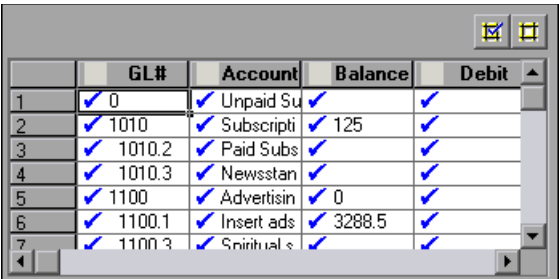
Checkpoint timeout: seconds

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. Select Before current step if you want to check the table content before the highlighted step is performed. Select After current step if you want to check the table content after the highlighted step is performed.</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

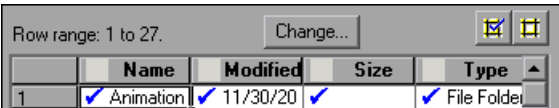
The grid area of the Table Checkpoint Properties dialog box represents the cells in the table. The column header names are captured from the table you selected for your checkpoint.



	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	✓ Snirhtual 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 548.

When you create a new table checkpoint, all cells contain a blue check mark, indicating they are all selected for verification. You can instruct QuickTest to check the entire table, specific rows, specific columns, or specific cells. QuickTest checks only cells containing a check mark.

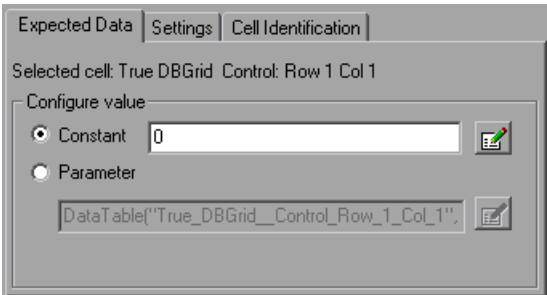
To:	Do this:
Add a single cell to or remove it from the check	Double-click the cell
Add an entire row to or remove it from the check	Double-click the row header
Add an entire column to or remove it from the check	Double-click the column header
Add all cells to or remove all cells from the check	Double-click the top-left corner of the grid
Add a range of cells to the check	Select the cells to add to the check and click the Add to Check button 
Remove a range of cells from the check	Select the cells to remove from the check and click the Remove from Check button 

Notes:

- Double-clicking on the grid toggles the settings for all selected cells. Therefore, if you double-click a row header, a column header, or the top left corner of the grid, any cells that were previously included in the check are removed from it, and any cells that were not previously included in the check are added to it.
 - When more than one cell is selected, the options in the Expected Data tab are disabled.
-

Specifying the Expected Data

The Expected Data tab displays options for setting the expected value of the selected cell in the table.



You can modify the value of a cell or you can parameterize it to use a value from an external source, such as the Data Table or an environment variable. During the run session, QuickTest compares the value specified in this tab with the actual value that it finds during the run session. If the expected value and the actual value do not match, the checkpoint fails.

To modify or parameterize several cells in the table, select a cell and then set your preferences for that cell in the Expected Data tab. Repeat the process for each cell you want to modify.

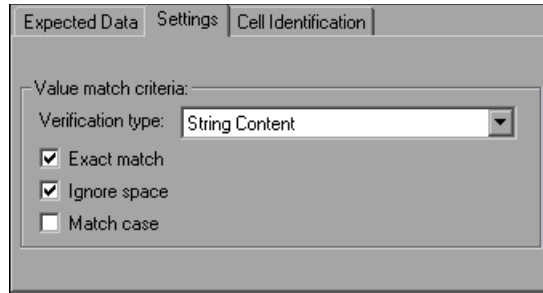
The Expected Data tab includes the following:

Selected cell	Indicates the table name and the row and column numbers of the selected cell.
Configure value	Enables you to set the expected value of the cell as a constant or parameter. For more information on modifying values, see “Setting Values in the Configure Value Area” on page 757.

Note: When more than one cell is selected, the options in the Expected Data tab are disabled.

Specifying the Value Type Criteria

The Settings tab includes options that determine how the actual cell values are compared with the expected cell values. The settings in this tab apply to all selected cells.



The default setting is to treat cell values as strings and to check for the exact text, while ignoring spaces.

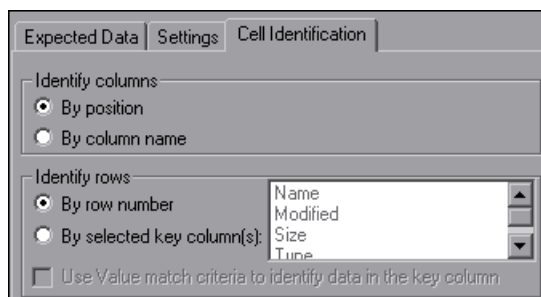
The Settings tab includes the following options:

Option	Description
Verification type	<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.

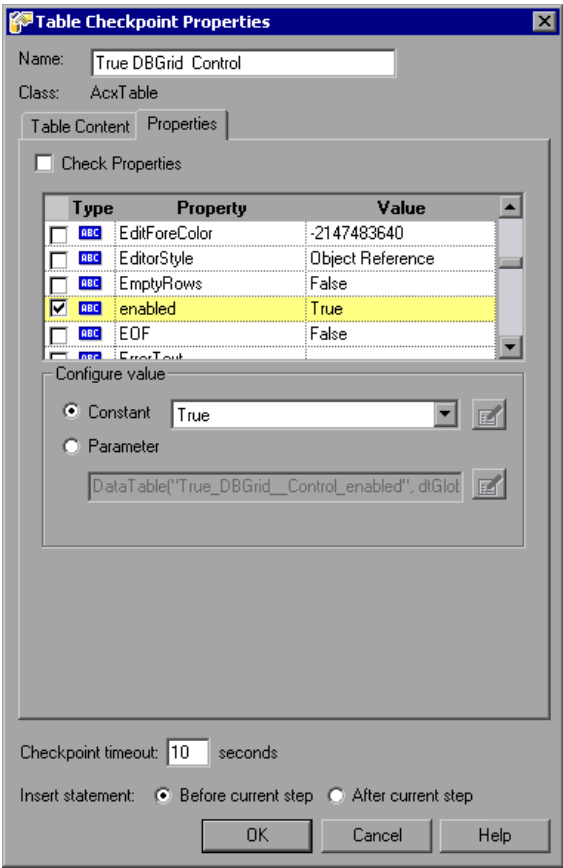


The Cell Identification tab includes the following options:

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

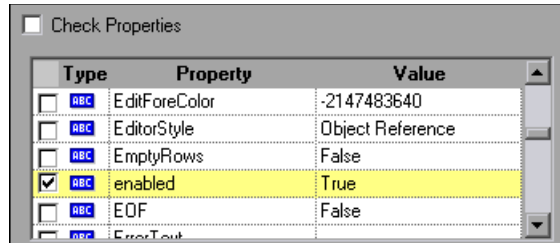
For certain environments, you can specify which table (or grid) properties you want to check. By default, when you create a table checkpoint on an object, QuickTest captures all the object’s properties, but does not select any properties to check.



Note: For information on general table checkpoint options, such as **Name** and **Checkpoint timeout**, see “Understanding and Setting General Table Checkpoint Options” on page 537.






Selecting Properties to Check

When you create a table checkpoint, the Properties pane displays the table object's default properties, including the properties, their values, and their types.



You instruct QuickTest to perform a properties check by selecting the **Check Properties** check box. (This check box is cleared by default.)

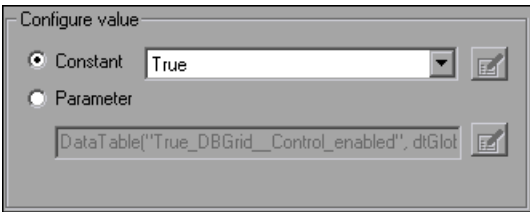
The Properties pane for the object contains the following:

Check box	<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 757.

Editing the Expected Value of a Table Property

The **Configure value** area enables you to define the expected value of the property as a **Constant** or a **Parameter**.



For information on modifying property values, see “Setting Values in the Configure Value Area” on page 757.

Modifying a Table Checkpoint

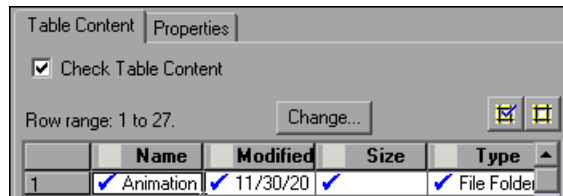
You can change the expected data, settings and cell identification options for an existing table checkpoint.

To modify the settings of the table checkpoint:

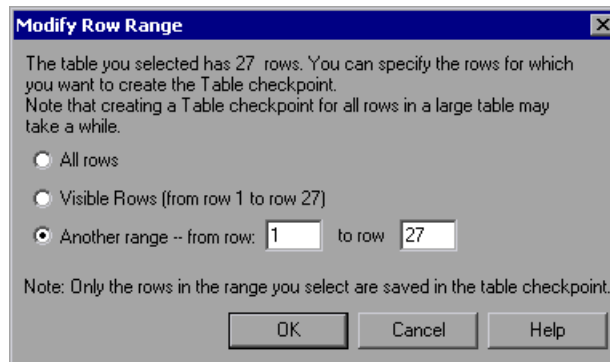
- 1 In the Keyword View or Expert View, right-click the table checkpoint that you want to modify and select **Checkpoint Properties**. Alternatively, select the step containing the checkpoint and select **Edit > Step Properties > Checkpoint Properties**. The Table Checkpoint Properties dialog box opens.
- 2 Modify the settings as described in “Understanding the Table Checkpoint Properties Dialog Box” on page 535.

To modify the number of rows in an existing table checkpoint:

- 1 Open the application containing the table or list view object you want to check and display the object in the application.
- 2 In the Keyword View or Expert View, right-click the table checkpoint that you want to modify and select **Checkpoint Properties**. Alternatively, select the step containing the checkpoint and select **Edit > Step Properties > Checkpoint Properties**. The Table Checkpoint Properties dialog box opens, displaying the currently selected row range.



- 3 In the Table Content tab, click the **Change** button at the top of the dialog box (above the grid area). The Modify Row Range dialog box opens.



- 4 Select the range of rows you want to include in your checkpoint. You can include all the rows, only the visible rows, or another range that you specify.

Note: The **Visible Rows** option may not be available for some environments or object types.

- 5 Click **OK**. The Modify Row Range dialog box closes, and the Table Checkpoint Properties dialog box displays the rows you specified in the Modify Row Range dialog box.
 - If your modified row range includes new rows, QuickTest captures the current values of the new rows from the open application.
 - If your modified row range includes some or all of the rows that were already included in the checkpoint, the expected values of those cells are not changed. This enables you to modify the row range without losing parameterization, regular expressions, or other changes you may have made to the expected cell values in your checkpoint.

Therefore, you cannot use the Modify Row Range dialog box to update the expected values of an existing table checkpoint. To update the expected values of your checkpoint, use the **Update Run Mode** option. For more information, see “Updating a Test Using the Update Run Mode Option” on page 1125.

- If your modified row range excludes some or all of the rows that were previously included in your checkpoint, those rows (and any modifications you made to the expected values) are deleted from the checkpoint.

21

Checking Text

QuickTest can check that a text string is displayed in the appropriate place in an application.

This chapter includes:

- About Checking Text on page 551
- Creating a Text Checkpoint on page 552
- Creating a Text Area Checkpoint on page 554
- The Text / Text Area Checkpoint Properties Dialog Box on page 557
- Modifying a Text or Text Area Checkpoint on page 570
- Creating a Standard Checkpoint for Checking Text on page 570

About Checking Text

You can check that a specified text string is displayed by adding one of the following checkpoints to your test.

- **Standard Checkpoint.** Enables you to check the **text** property of an object. You can use standard checkpoints to check text in Windows-based and other types of applications (including Web-based applications). For more information on standard checkpoints, see “Creating Standard Checkpoints” on page 506.
- **Text Area Checkpoint.** Enables you to check that a text string appears within a defined area in a Windows application, according to specified criteria. It is supported for a variety of QuickTest add-in environments, such as Standard Windows, Java, Visual Basic, and ActiveX. For more information, see the *HP QuickTest Professional Add-ins Guide*.

- **Text Checkpoint.** Enables you to check that the text is displayed in a screen, window, or Web page, according to specified criteria. Text checkpoints are supported for many QuickTest add-in environments (as listed in “Supported Checkpoints” on page 504). For more information, see the *HP QuickTest Professional Add-ins Guide*.

When checking text, QuickTest tries to retrieve the text directly from the object. If QuickTest cannot retrieve the text in this manner (for example, because the text is part of a picture), it tries to retrieve the text using an OCR (optical character recognition) mechanism. The OCR mechanism translates images of handwritten or typewritten text into machine-editable text.

Creating a Text Checkpoint

You can add a text checkpoint while recording or editing steps in a Windows- or Web-based application.

Note: Before you create a text checkpoint, make sure you configure the required capture settings in the General > Text Recognition pane (**Tools > Options > Text Recognition** node). For more information, see “The Options Dialog Box: General > Text Recognition Pane” on page 742 and “About Working with Text Recognition for Windows-Based Objects” on page 742.

To add a text checkpoint while recording:

- 1 Display the page, window, or screen containing the text you want to check.



- 2 Select **Insert > Checkpoint > Text Checkpoint**, or click the **Insert Checkpoint or Output Value** toolbar button and select **Text Checkpoint**.

The QuickTest window is hidden, and the pointer changes into a pointing hand. For more information about using the pointing hand feature, see “Tips for Using the Pointing Hand” on page 553.

- 3 Click the text string for which you want to create the checkpoint. The Text Checkpoint Properties dialog box opens.

- 4 Specify the checkpoint settings. For more information, see “The Text / Text Area Checkpoint Properties Dialog Box” on page 557.
- 5 Click **OK** to close the dialog box. A checkpoint statement is added for the selected object.

Tips for Using the Pointing Hand

- You can hold the left CTRL key to change the pointing hand to a standard pointer. You can then change the window focus or perform operations in QuickTest or in your application, such as right-clicking, using the scroll bars, or moving the pointer over an object to display a context menu.
- If the window containing the object you want to select is partially hidden by another window, hold the pointing hand over the partially hidden window for a few seconds until it comes to the foreground. Then point to and click the required object. You can configure the length of time required to bring a window into the foreground using the General pane of the Options dialog box.
- If the window containing the object you want to select is fully hidden by another window, or if a dialog box is hidden behind a window, press the left CTRL key and arrange the windows as needed.
- If the window containing the object you want to select is minimized, you can display it by holding the left CTRL key, right-clicking the application in the Windows task bar, and choosing **Restore** from the context menu.
- If the object you want to select can be displayed only by performing an event (such as right-clicking or moving the pointer over an object to display a context menu), hold the left CTRL key. The pointing hand temporarily turns into a standard pointer and you can perform the event. When the object you want to select is displayed, release the left CTRL key. The pointer becomes a pointing hand again.

To add a text checkpoint while editing a test:



- 1 Make sure the **Active Screen** toolbar button is selected.
- 2 Click the step where you want to add a checkpoint. The Active Screen displays the page or screen corresponding to the highlighted step.

- 3 Highlight a text string on the Active Screen.
- 4 Right-click the text string and select **Insert Text Checkpoint**. The Text Checkpoint Properties dialog box opens.
- 5 Specify the settings for the checkpoint. For more information, see “The Text / Text Area Checkpoint Properties Dialog Box” on page 557.
- 6 Click **OK** to close the dialog box. A checkpoint statement is added for the selected object.

Creating a Text Area Checkpoint

You can add a text area checkpoint only while recording a test on Windows-based applications, such as Standard Windows, Java, Visual Basic, and ActiveX. To determine whether text area checkpoints are supported for a specific QuickTest add-in environment, see the *HP QuickTest Professional Add-ins Guide*.

Note: Before you create a text area checkpoint, make sure you configure the required capture settings in the General > Text Recognition pane (**Tools > Options > Text Recognition** node). For more information, see “The Options Dialog Box: General > Text Recognition Pane” on page 742 and “About Working with Text Recognition for Windows-Based Objects” on page 742.

To add a text area checkpoint:

- 1 Select **Insert > Checkpoint > Text Area Checkpoint**, or click the arrow next to the **Insert Checkpoint** toolbar button and select **Text Area Checkpoint**.

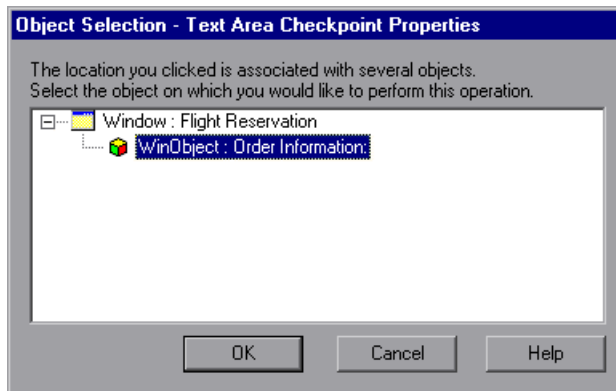
The QuickTest window is hidden, and the mouse pointer turns into a crosshairs pointer.

- 2 Define the area containing the text you want QuickTest to check by clicking and dragging the crosshairs pointer. (See “Considerations for Defining the Text Area” on page 556.)

Tip: Hold down the left mouse button and use the arrow keys to make precise adjustments to the defined area.

Release the mouse button after outlining the area required.

If the area you defined is associated with more than one object, the Object Selection–Text Area Checkpoint Properties dialog box opens.



- 3 Select the object for which you are creating the checkpoint. The Text Area Checkpoint Properties dialog box opens.
- 4 Specify the checkpoint settings. For more information, see “The Text / Text Area Checkpoint Properties Dialog Box” on page 557.
- 5 Click **OK** to close the dialog box. A checkpoint statement is added for the selected object in the Keyword View and Expert View.

Considerations for Defining the Text Area

When checking text displayed in a Windows-based application, it is often advisable to define a text area larger than the actual text you want QuickTest to check. You then use the Text Area Checkpoint Properties dialog box to configure the relative position of the Checked Text within the captured string. When QuickTest runs the test, it checks for the selected text within the defined area, according to the settings you configured.

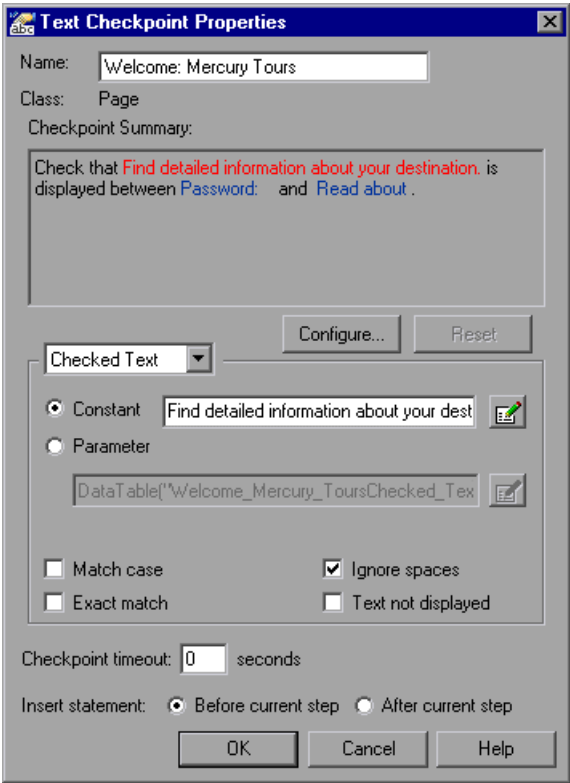
Consider the following when defining the area for a text area checkpoint:

- If you parameterize a text string, the captured area must be large enough to accommodate any string that might replace the one selected during a run session.
- The captured area must be large enough to include all parts of the required text (**Checked Text** / **Text Before** / **Text After**).
- Text may change its position during run sessions; therefore, make sure that the area you capture is large enough to allow for acceptable position shifts. If the defined area is too small, even a slight shift in the text's position will cause the run to fail, although the changed position may be acceptable to you. If, on the other hand, the position of the text on the screen is critical, or if you do not want it to exceed certain boundaries, set the defined area accordingly.

The Text / Text Area Checkpoint Properties Dialog Box

Description	<p>Enables you to specify the text to be checked, as well as specify which text is displayed before and after the checked text.</p> <p>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.</p> <p>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.</p>
How to Access	<ul style="list-style-type: none"> ➤ “Creating a Text Checkpoint” on page 552 ➤ “Creating a Text Area Checkpoint” on page 554 ➤ “Modifying a Text or Text Area Checkpoint” on page 570
Important Information	<p>Before you create a text or text area checkpoint, make sure you set the required options, as described in “The Options Dialog Box: General > Text Recognition Pane” on page 742.</p>
Learn More	<p>Conceptual overview: “About Checking Text” on page 551</p> <p>Primary tasks:</p> <ul style="list-style-type: none"> ➤ “Creating a Text Checkpoint” on page 552 ➤ “Creating a Text Area Checkpoint” on page 554 ➤ “Modifying a Text or Text Area Checkpoint” on page 570 <p>Additional related topics: “Use-Case Scenario: Checking Text in an Image” on page 750</p>

Below is an image of the Text/Text Area Checkpoint Properties dialog box:



This image is an example of the Text Checkpoint Properties dialog box that opens when adding a text checkpoint to an existing test during an editing session. The Text Checkpoint Properties dialog box options differ slightly during a recording session or when editing an existing checkpoint. The Text Area Checkpoint Properties dialog box is similar to the Text Checkpoint Properties dialog box.

Text/Text Area Checkpoint Properties Dialog Box Options


Information	Description
General checkpoint information area	<p>Enables you to view and specify the name of the checkpoint and to view the type of object.</p> <p>See: “Understanding and Setting General Text Checkpoint Information” on page 561.</p>
Checkpoint Summary area	<p>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.</p> <p>For text checkpoints in Web-based environments, it displays the text you selected when creating the checkpoint, plus some text before and after it. For text and text area checkpoints in Windows-based environments, it displays the text you selected when creating the checkpoint.</p> <p>Note: In Windows-based environments, if there is more than one line of text selected, the Checkpoint Summary area displays [complex value] instead of the selected text string. You can then click Configure to view and manipulate the actual selected text for the checkpoint.</p> <p>You can designate parts of the captured string as Checked Text and other parts as Text Before and Text After by clicking the Configure button. For more information, see “Configuring the Text Selection” on page 561.</p>

Information	Description
Options for checked text area	<p>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.</p> <p>See:</p> <ul style="list-style-type: none">➤ “Setting Options for Checked Text” on page 564➤ “Setting Options for Text Displayed Before the Checked Text” on page 566➤ “Setting Options for Text Displayed After the Checked Text” on page 567
Checkpoint timeout and Insert Statement area	<p>Specify when QuickTest should perform the checkpoint by specifying the timeout and location of the checkpoint. These options are available only when inserting a new checkpoint</p> <p>See “Setting Checkpoint Timeout and Statement Location Options” on page 569.</p>

Understanding and Setting General Text Checkpoint Information

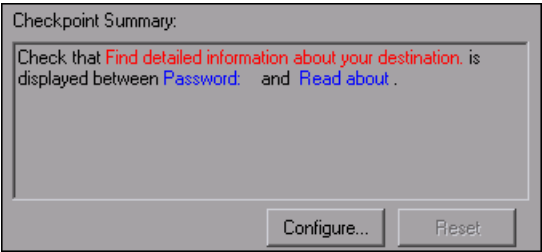
The top part of the Text/Text Area Checkpoint Properties dialog box contains the following options:



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:</p> <ul style="list-style-type: none">➤ is unique➤ does not begin or end with a space➤ does not contain " (double quotation mark)➤ does not contain the following character combinations: := @ @
Class	<p>Specifies the type of object (read-only).</p>
Find in Repository button 	<p>Displays the checkpoint in its object repository.</p> <p>Note: This option is available only when editing an existing checkpoint. It is not available when creating a new checkpoint.</p>

Configuring the Text Selection

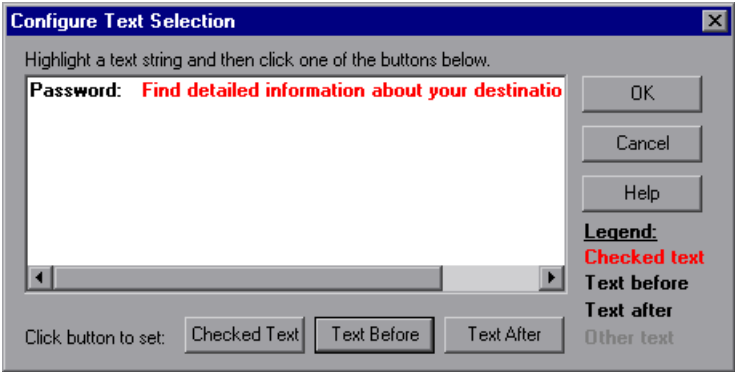
You can view and modify the text selection displayed in the Checkpoint Summary area.



The Checkpoint Summary area contains the following options:

Option	Description
Configure	Opens the Configure Text Selection dialog box, where you can specify the checked text, the text before (if any), and the text after (if any).
Reset	Resets the text selection to the previous configuration.

To designate **Checked Text**, **Text Before**, and **Text After**, you use the Configure Text Selection dialog box (opened by clicking the **Configure** button). The Configure Text Selection dialog box displays the text you captured when creating the text checkpoint, as well as text before and after the selected text. QuickTest displays the checked text in red and the text before and after it in black (as indicated in the **Legend** displayed in the dialog box).



To modify which text is checked and how that text is found, based on the text before and after it, highlight the text you want to set for one of these items and then click the appropriate button.

Option	Description
Checked Text	Sets the highlighted text as the checked text. QuickTest displays this text in red and the remainder in black.
Text Before	Sets the highlighted text as the text before the checked text.
Text After	Sets the highlighted text as the text after the checked text.

Note: If no text is selected in the Configure Text Selection dialog box, then nothing happens when you click these buttons.

To remove text from the current text selection configuration, highlight only the text you want included as the before or after text and click the appropriate button. Any text that is not selected as **Checked Text**, **Text Before**, or **Text After** is displayed in gray. The gray text is not displayed the next time the Configure Text Selection dialog box is opened.

For example, in the sample image above, if you wanted to check only the word *information*, and you wanted QuickTest to look for this text between *Find detailed* and *about your destination*, then:

- Highlight the word *information*, and click **Checked Text**. The word *information* remains red, and the other text turns black.
- Highlight the words *Find detailed* and click **Text Before**. The words *Find detailed* remain black, and all text preceding it turns gray. This gray text will be removed from the text configuration when you click **OK**.
- The words *about your destination* are already marked in black as the text after, so there is no need to modify this configuration.

Note: If you want to configure more text than is displayed, you must cancel the text checkpoint and select a larger text area or selection in your application.

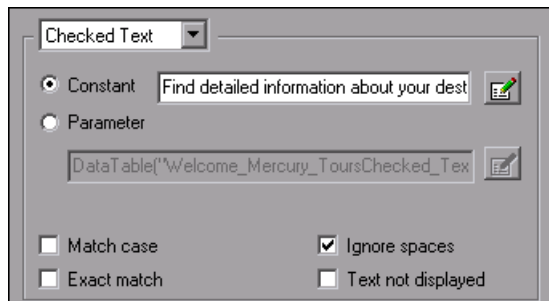
When you close the Configure Text Selection dialog box, the Checkpoint Summary area displays the new text selection configuration.

Setting Options for the Text to be Checked

The middle area of the Text/Text Area Checkpoint Properties dialog box enables you to set options for the checked text, text before, and text after, as described in the following sections.

Setting Options for Checked Text

You set options for the checked text by choosing **Checked Text** from the list box. In the Checked Text area, you can indicate whether you want the checked text to be a constant or a parameter, and you can set the criteria for a successful match.



You can choose from the following options for the checked text:

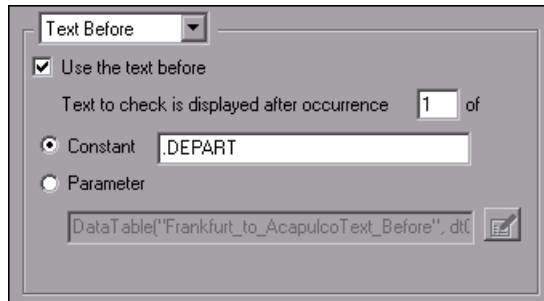
- **Constant.** (Default) Sets the expected value of the checked text as a constant. For information on modifying values, see “Setting Values in the Configure Value Area” on page 757.

Tip: The **Constant** box displays the checked text. You can change the checked text by typing in the **Constant** box or by using the Configure Text Selection dialog box.

- **Parameter.** Sets the expected value of the checked text as a parameter. For information on modifying values, see “Setting Values in the Configure Value Area” on page 757.
- **Match case.** Conducts a case-sensitive check.
- **Exact match.** Checks for the exact expected text. For example, if you create a checkpoint with the following description, Check that New York is displayed between Flight departing from and to San Francisco, and select **Exact match**, if the actual text is New York City, the checkpoint fails. If you do not select **Exact match**, the checkpoint passes because the expected text is contained within the actual text.
- **Ignore spaces.** Ignores spaces in the captured text when performing the check. The presence or absence of spaces does not affect the outcome of the check.
- **Text not displayed.** Checks that the text string is not displayed. For example, if you create a checkpoint with the following description, Check that New York is displayed between Flight departing from and to San Francisco, and select **Text not displayed**, QuickTest checks that the text New York is not displayed.

Setting Options for Text Displayed Before the Checked Text

You set options for the text displayed before the checked text by choosing **Text Before** from the list box. In the Text Before area, you can set the text before the checked text as a constant or a parameter.



You can choose from the following options when setting the text displayed before the checked text:

- **Use the text before.** Checks the text before the checked text. To ignore this text, clear this check box.
- **Text to check is displayed after occurrence.** Checks that the checked text is displayed after the specified text.

If the identical text string you specify is displayed more than once on the page, you can specify the occurrence of the string to which you are referring.

If you accept the default text that QuickTest recommends, the number in the dialog box will be correct. If you modify the text, confirm that the occurrence number is accurate.

If you choose a non-unique text string, change the occurrence number appropriately. For example, if you want to check that the words Mercury Tours are displayed after the fourth occurrence of the word the, enter 4 in the **Text to check is displayed after occurrence** box.

- **Constant.** (Default) Sets the expected value of the text before the checked text as a constant. For information on modifying values, see “Setting Values in the Configure Value Area” on page 757.

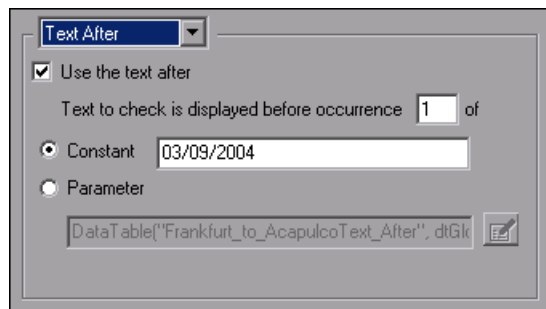
If you modify the text, whenever possible, use a string that is unique within the object so that the occurrence number is 1.

Tip: The **Constant** box displays the text before the checked text. You can change the text by typing in the **Constant** box or by using the Configure Text Selection dialog box.

- **Parameter.** Sets the expected value of the text before the checked text as a parameter. For information on modifying values, see “Setting Values in the Configure Value Area” on page 757.

Setting Options for Text Displayed After the Checked Text

You set options for or the text displayed after the checked text by choosing **Text After** from the list box. In the Text After area, you can set the text after the checked text as a constant or a parameter.



You can choose from the following options when setting the text displayed after the checked text:

- **Use the text after.** Checks the text after the checked text. To ignore this text, clear this check box.
- **Text to check is displayed before occurrence.** Checks that the checked text is displayed before the specified text. If the identical text string you specify is displayed more than once on the page, you can specify to which occurrence of the string you are referring.

If you accept the default text that QuickTest recommends, the number in the dialog box will be correct. If you modify the text, confirm that the occurrence number is also accurate.

If you choose a non-unique text string, change the occurrence number appropriately. For example, if you want to check that the words Mercury Tours are displayed before the fourth occurrence of the word the, enter 4 in the **Text to check is displayed before occurrence** box.

- **Constant.** (Default) Sets the expected value of the text after the checked text as a constant. For information on modifying values, see “Setting Values in the Configure Value Area” on page 757.

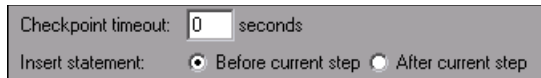
If you modify the text, whenever possible, use a string that is unique within the object so that the occurrence number is 1.

Tip: The **Constant** box displays the text after the checked text. You can change the text by typing in the **Constant** box or by using the Configure Text Selection dialog box.

- **Parameter.** Sets the expected value of the text after the checked text as a parameter. For information on modifying values, see “Setting Values in the Configure Value Area” on page 757.

Setting Checkpoint Timeout and Statement Location Options

You can specify the time interval during which QuickTest attempts to perform the checkpoint successfully by modifying the selections in the bottom part of the Text/Text Area Checkpoint Properties dialog box. You can also specify when to perform the checkpoint.



The screenshot shows a dialog box with two sections. The first section is labeled 'Checkpoint timeout:' and contains a text input field with the value '0' and the word 'seconds' to its right. The second section is labeled 'Insert statement:' and contains two radio button options: 'Before current step' (which is selected) and 'After current step'.

- **Checkpoint timeout.** Specifies the time interval (in seconds) during which QuickTest attempts to perform the checkpoint successfully. QuickTest continues to perform the checkpoint until it passes or until the timeout occurs. If the checkpoint does not pass before the timeout occurs, the checkpoint fails.

For example, suppose it takes some time for an object to achieve an expected state. Increasing the checkpoint timeout value in this case can help ensure that the object has sufficient time to achieve that state, enabling the checkpoint to pass (if the data matches) before the maximum timeout is reached.

If you specify a checkpoint timeout other than **0**, and the checkpoint fails, the Test Results window displays information on the checkpoint timeout.

- **Insert statement.** Specifies when to perform the checkpoint. Select **Before current step** if you want to check the value of the text before the highlighted step is performed. Select **After current step** if you want to check the value of the text after the highlighted step is performed.

Note: The **Insert statement** option is not available when adding a new text checkpoint or a text area checkpoint during recording, or when modifying an existing checkpoint. It is available only when adding a new text checkpoint to an existing test while editing.

Modifying a Text or Text Area Checkpoint

You can modify an existing text or text area checkpoint.

To modify a text or text area checkpoint:

- 1 In the Keyword View or Expert View, right-click the checkpoint that you want to modify and select **Checkpoint Properties**. Alternatively, select the step containing the checkpoint and select **Edit > Step Properties > Checkpoint Properties**. The Text/Text Area Checkpoint Properties dialog box opens.
- 2 Modify the settings. For more information, see “The Text / Text Area Checkpoint Properties Dialog Box” on page 557.

Creating a Standard Checkpoint for Checking Text

You can check the text property of an object in Windows-based and other types of applications (including Web-based applications) by using a standard checkpoint.

To add a standard checkpoint for checking text while recording:



- 1 Select **Insert > Checkpoint > Standard Checkpoint** or click the **Insert Checkpoint or Output Value** toolbar button and select **Standard Checkpoint**. The QuickTest window is hidden, and the pointer changes into a pointing hand. For more information about using the pointing hand feature, see “Tips for Using the Pointing Hand” on page 573.
- 2 Click the object whose text you want to check. The Object Selection - Checkpoint Properties dialog box opens.
- 3 Select the item you want to check from the displayed object tree.
- 4 Click **OK**. The Checkpoint Properties dialog box opens.
- 5 In the **Name** box, either accept the name that QuickTest assigns to the checkpoint or specify another name for it. By default, the checkpoint name is the name of the test object on which the checkpoint is being performed.

If you rename the checkpoint, make sure that the name:

- is unique
- does not begin or end with a space
- does not contain " (double quotation mark)
- does not contain the following character combinations:
 - :=
 - @ @

Note: The **Class** area displays the type of test object on which the checkpoint is being performed.

- 6 Select the **text** property.
- 7 If necessary, edit the **text** value you want QuickTest to check. Note that you can parameterize this value.
- 8 If you want to check only text, clear the other check boxes in the dialog box.
- 9 Click **OK** to close the dialog box. A checkpoint statement is added for the selected object.

To add a standard checkpoint for checking text while editing:

- 1** Right-click the step for the object whose text you want to check, and select **Insert Standard Checkpoint**. The Checkpoint Properties dialog box opens.
- 2** In the **Name** box, either accept the name that QuickTest assigns to the checkpoint or specify another name for it. By default, the checkpoint name is the name of the test object on which the checkpoint is being performed.

If you rename the checkpoint, make sure that the name:

- is unique
- does not begin or end with a space
- does not contain " (double quotation mark)
- does not contain the following character combinations:
 - :=
 - @ @

Note: The **Class** area displays the type of test object on which the checkpoint is being performed.

- 3** Select the **text** property.
- 4** If necessary, edit the text value you want QuickTest to check. Note that you can parameterize this value.
- 5** If you want to check only text, clear the other check boxes in the dialog box.
- 6** Click **OK** to close the dialog box. A checkpoint statement is added for the selected object.

For more information on creating standard checkpoints, see Chapter 18, "Checking Object Property Values Using Standard Checkpoints."

Tips for Using the Pointing Hand

- You can hold the left CTRL key to change the pointing hand to a standard pointer. You can then change the window focus or perform operations in QuickTest or in your application, such as right-clicking, using the scroll bars, or moving the pointer over an object to display a context menu.
- If the window containing the object you want to select is partially hidden by another window, hold the pointing hand over the partially hidden window for a few seconds until it comes to the foreground. Then point to and click the required object. You can configure the length of time required to bring a window into the foreground using the General pane of the Options dialog box.
- If the window containing the object you want to select is fully hidden by another window, or if a dialog box is hidden behind a window, press the left CTRL key and arrange the windows as needed.
- If the window containing the object you want to select is minimized, you can display it by holding the left CTRL key, right-clicking the application in the Windows task bar, and choosing **Restore** from the context menu.
- If the object you want to select can be displayed only by performing an event (such as right-clicking or moving the pointer over an object to display a context menu), hold the left CTRL key. The pointing hand temporarily turns into a standard pointer and you can perform the event. When the object you want to select is displayed, release the left CTRL key. The pointer becomes a pointing hand again.

22

Checking Databases

By adding database checkpoints, you can check the contents of databases accessed by your application. Database checkpoints are supported by all environments.

This chapter includes:

- About Checking Databases on page 575
- Creating a Check on a Database on page 576
- Understanding the Database Checkpoint Properties Dialog Box on page 581
- Modifying a Database Checkpoint on page 590

About Checking Databases

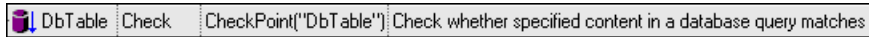
You can use database checkpoints in your test to check databases accessed by your application, and to detect defects. To do this, you define a query on your database. Then you create a database checkpoint that checks the results of the query.

You can define a database query in the following ways:

- Using Microsoft Query. You can install Microsoft Query from the custom installation of Microsoft Office.
- By manually defining an SQL statement.

Creating a Check on a Database

You create a database checkpoint based on the results of the query (**result set**) you defined on a database. You can create a check on a database to check the contents of the entire result set, or a part of it. QuickTest captures the current data from the database, saves this information as **expected data**, and inserts a database checkpoint into the test. This checkpoint is displayed in the Expert View as a DbTable.Check CheckPoint statement and as a step in the Keyword View, as follows:



When you run the test, the database checkpoint compares the current data in the database to the expected data defined in the Database Checkpoint Properties dialog box. If the expected data and the current results do not match, the database checkpoint fails. You can view the results of the checkpoint in the Test Results window. For more information, see Chapter 33, “Viewing Run Session Results.”

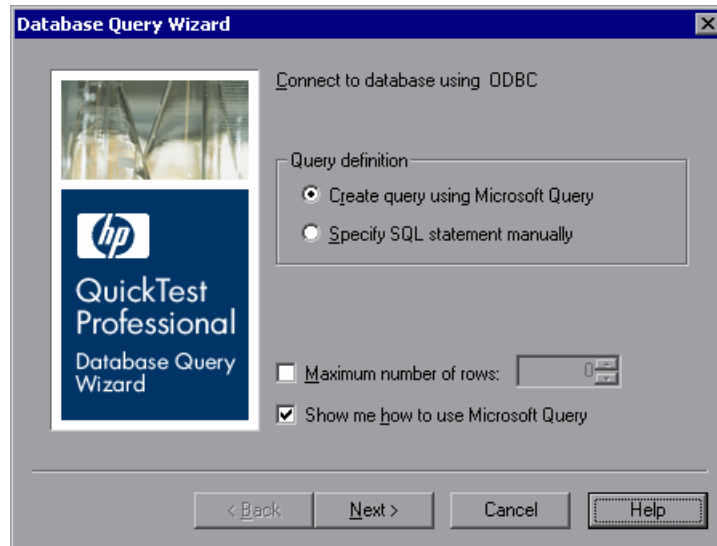
You can modify the expected data of a database checkpoint before you run your test. You can also make changes to the query in an existing database checkpoint. This can be useful if you move the database to a new location on the network.

Creating a Database Checkpoint

You can define the query for your checkpoint using Microsoft Query or by manually entering a database connection and SQL statement.

To create a database checkpoint:

- 1 Select **Insert > Checkpoint > Database Checkpoint**. The Database Query Wizard opens.



- 2 Select your database selection preferences. You can choose from the following options:
 - **Create query using Microsoft Query.** Opens Microsoft Query, enabling you to create a new query. After you finish defining your query, you return to QuickTest. This option is available only if you have Microsoft Query installed on your computer.
 - **Specify SQL statement manually.** Opens the **Specify SQL statement** screen in the wizard, which enables you to specify the connection string and an SQL statement.

- **Maximum number of rows.** Select this check box if you would like to limit the number of rows and enter the maximum number of database rows to check. You can specify a maximum of 32,000 rows.
 - **Show me how to use Microsoft Query.** Displays an instruction screen when you click **Next** before opening Microsoft Query. (Enabled only when **Create query using Microsoft Query** is selected).
- 3** Click **Next**. The screen that opens depends on the option you selected in the previous step.
- If you chose **Create query using Microsoft Query** in the previous step, Microsoft Query opens. Choose a data source and define a query. For more information on creating a query, see “Creating a Query in Microsoft Query” on page 579.

Note: If you chose **Show me how to use Microsoft Query**, the Instruction for Microsoft Query screen opens. When you click **OK**, Microsoft Query opens.

- If you chose **Specify SQL statement manually** in the previous step, the Specify SQL statement screen opens. Specify the connection string and the SQL statement, and click **Finish**. For more information on specifying SQL statements, see “Specifying SQL Statements” on page 580.

The Database Checkpoint Properties dialog box opens.

- 4** Select the checks to perform on the result set as described in “Understanding the Database Checkpoint Properties Dialog Box” on page 581. You can also modify the expected data in the result set.
- 5** Click **OK** to close the dialog box. A checkpoint statement is added for the selected object in the Keyword View and Expert View.

Creating a Query in Microsoft Query

You can use Microsoft Query to choose a data source and define a query within the data source.

To choose a data source and define a query in Microsoft Query:

- 1** When Microsoft Query opens during the insert database checkpoint process, choose a new or an existing data source.
- 2** Define a query.
- 3** When you are done, in the Finish screen of the Query Wizard, select **Exit and return to QuickTest Professional** and click **Finish** to exit Microsoft Query. Alternatively, click **View data or edit query in Microsoft Query** and click **Finish**. After viewing or editing the data, select **File > Exit and return to QuickTest Professional** to close Microsoft Query and return to QuickTest.
- 4** The Database Checkpoint Properties dialog box opens. Select the checks to perform on the result set as described in “Understanding the Database Checkpoint Properties Dialog Box” on page 581. You can also modify the expected data in the result set.
- 5** Click **OK** to close the dialog box. A checkpoint statement is added for the selected object in the Keyword View and Expert View.

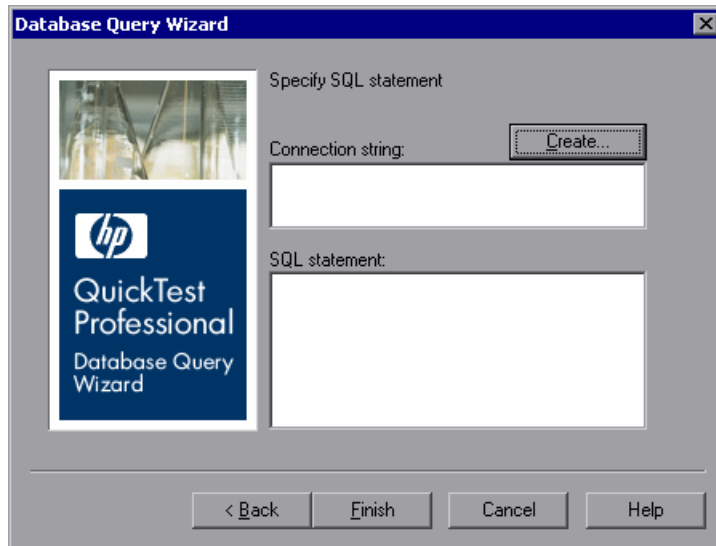
For more information on working with Microsoft Query, see your Microsoft Query documentation.

Specifying SQL Statements

You can manually specify the database connection string and the SQL statement.

To specify SQL statements:

- 1 Select **Specify SQL statement** in the Database Query Wizard. The following screen opens:



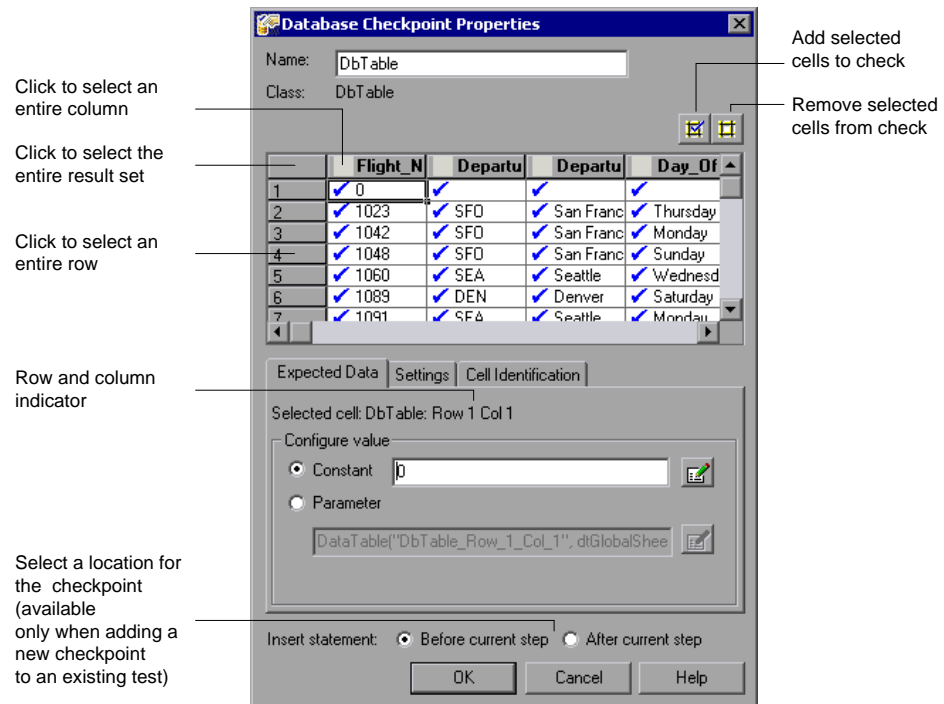
- 2 Specify the connection string and the SQL statement, and click **Finish**.
 - **Connection string.** Enter the connection string, or click **Create** to open the ODBC Select Data Source dialog box. You can select a **.dsn** file in the ODBC Select Data Source dialog box or create a new **.dsn** file to have the Database Query Wizard insert the connection string in the box for you.
 - **SQL statement.** Enter the SQL statement.

QuickTest takes several seconds to capture the database query and restore the QuickTest window.

- 3 Select the checks to perform on the result set as described in “Understanding the Database Checkpoint Properties Dialog Box” on page 581. You can also modify the expected data in the result set.
- 4 Click **OK** to close the dialog box. A checkpoint statement is added for the selected object in the Keyword View and Expert View.

Understanding the Database Checkpoint Properties Dialog Box

The Database Checkpoint Properties dialog box enables you to specify which cell contents of your database to check and which verification method and type to use. You can also edit or parameterize the expected data for the cells included in the check.



The Database Checkpoint Properties dialog box enables you to check database content.

- The information area at the top of the dialog box displays the name of the checkpoint and the class of the test object on which the checkpoint is being performed. You can rename the checkpoint, if required. When editing an existing checkpoint, you can also view it in the object repository. For more information, see “Identifying the Database Checkpoint” on page 583.
- The grid area displays the data that was captured for the checkpoint. This is the expected data. You use this area to specify which cells you want to check. For more information, see “Specifying Which Cells to Check” on page 583.
- **Expected Data tab.** Enables you to set each checked cell as a constant or parameterized value. For more information, see “Specifying the Expected Data” on page 585.
- **Settings tab.** Enables you to set the criteria for a successful match between the expected and actual values. For more information, see “Specifying the Value Type Criteria in the Settings Tab” on page 586.
- **Cell Identification tab.** Enables you to instruct QuickTest how to locate the cells to be checked. For more information, see “Specifying the Cell Identification Settings” on page 588.

Tip: The value matching settings and cell identification criteria apply to all selected cells in the checkpoint. If you want to use different value matching or cell identification criteria for different cells in the database, create separate checkpoints and specify the relevant cells for each.


- The **Insert statement** option enables you to specify the location of the checkpoint within the test. This option is available only when you add a new checkpoint while editing a test. For more information, see “Specifying The Statement Location” on page 589.

Identifying the Database Checkpoint

The top part of the Database Checkpoint Properties dialog box contains the following options:



The screenshot shows a dialog box with two fields. The 'Name' field is a text box containing 'DbTable' and has a small icon to its right. The 'Class' field is a text box containing '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:</p> <ul style="list-style-type: none"> ➤ is unique ➤ does not begin or end with a space ➤ does not contain " (double quotation mark) ➤ does not contain the following character combinations: <pre>:= @@</pre>
Class	Specifies the type of object (read-only).
Find in Repository button 	<p>Displays the checkpoint in its object repository.</p> <p>Note: This option is available only when editing an existing checkpoint. It is not available when creating a new checkpoint.</p>

Specifying Which Cells to Check

The grid area of the Database Checkpoint Properties dialog box represents the cells in the captured result set.

Tip: You can change the column widths and row heights of the grid by dragging the boundaries of the column and row headers.

When you create a new database checkpoint, all cells contain a blue check mark, indicating they are selected for verification. You can select to check the entire results set, specific rows, specific columns, or specific cells. QuickTest checks only cells containing a check mark.

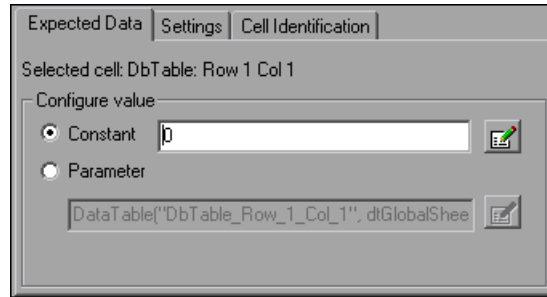
To:	Do this:
Add a single cell to or remove it from the check	Double-click the cell
Add an entire row to or remove it from the check	Double-click the row header
Add an entire column to or remove it from the check	Double-click the column header
Add the entire result set to or remove all cells from the check	Double-click the top-left corner of the grid
Add a range of cells to the check	Select the cells to add to the check and click the Add to Check button 
Remove a range of cells from the check	Select the cells to remove from the check and click the Remove from Check button 

Notes:

- Double-clicking on the grid toggles the settings for all selected cells. Therefore, if you double-click a row header, a column header, or the top left corner of the grid, any cells that were previously included in the check are removed from it, and any cells that were not previously included in the check are added to it.
- When more than one cell is selected, the options on the Expected Data tab are disabled.

Specifying the Expected Data

The Expected Data tab displays options for setting the expected value of the selected cell in the result set.



You can modify the value of a cell or you can parameterize it to use a value from an external source, such as the Data Table or an environment variable. During the run session, QuickTest compares the value specified in this tab with the actual value that it finds. If the expected value and the actual value do not match, the checkpoint fails. For example, you can instruct QuickTest to use a value from the Data Table as the expected value for a particular cell.

To modify or parameterize several cells in the result set, select a cell and then set your preferences for that cell in the Expected Data tab. Repeat the process for each cell you want to modify.

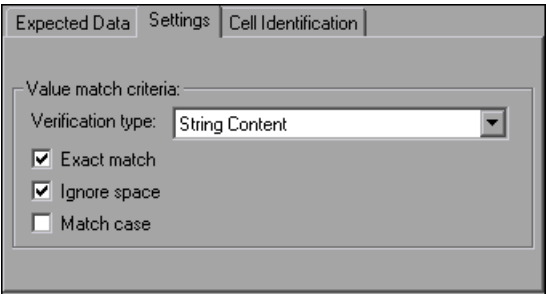
The Expected Data tab includes the following:

Selected cell	Indicates the table name and the row and column numbers of the selected cell.
Configure value area	Enables you to set the expected value of the cell as a constant or parameter. For more information on modifying values, see “Setting Values in the Configure Value Area” on page 757.

Note: When more than one cell is selected, the options on the Expected Data tab are disabled.

Specifying the Value Type Criteria in the Settings Tab

The Settings tab includes options that determine how the actual cell values are compared with the expected cell values. For example, you can instruct QuickTest to treat the value as a number so that 45 or 45.00 are treated as the same value, or you can instruct QuickTest to ignore spaces when comparing the values. The settings in this tab apply to all selected cells.



The default setting is to treat cell values as strings and to check for the exact text, while ignoring spaces.

The Settings tab includes the following options:

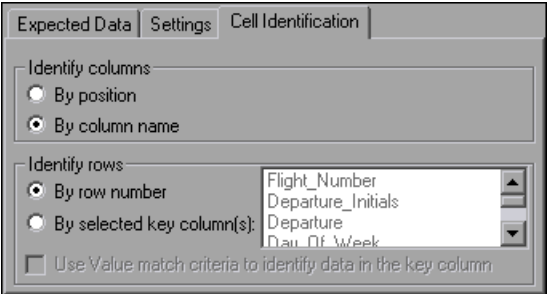
Option	Description
Verification type	<p>Specifies how cell contents are compared:</p> <ul style="list-style-type: none">➤ String Content. (Default) Evaluates the content of the cell as a string. For example, 2 and 2.00 are not recognized as the same string.➤ Numeric Content. Evaluates the content of the cell according to numeric values. For example, 2 and 2.00 are recognized as the same number.➤ Numeric Range. Compares the content of the cell against a numeric range, where the minimum and maximum values are any real number that you specify. This comparison differs from string and numeric content verification in that the actual result set data is compared against the range that you defined and not against a specific expected value.

Option	Description
Exact match	<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.



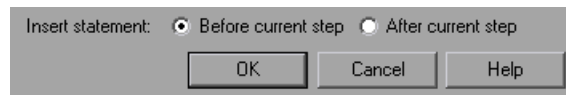
The Cell Identification tab includes the following options:

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>

Specifying The Statement Location

The **Insert statement** option specifies when to perform the checkpoint in the test.



- Select **Before current step** if you want to check the value of the object property before the highlighted step is performed.
- Select **After current step** if you want to check the value of the property after the highlighted step is performed.

Note: The **Insert statement** option is not available when adding a checkpoint during recording or when modifying an existing object checkpoint. It is available only when adding a new checkpoint to an existing test while editing it.

Modifying a Database Checkpoint

You can modify the SQL query definition and the expected data in an existing database checkpoint.

To modify the SQL query definition:

- 1** In the Keyword View, right-click the database object that you want to modify.
- 2** Select **Object Properties**.
- 3** Modify the SQL and connection string properties as necessary and click **OK**.

To modify the expected data in a database checkpoint:

- 1** In the Keyword View or Expert View, right-click the database checkpoint that you want to modify and select **Checkpoint Properties**. Alternatively, select the step containing the checkpoint and select **Edit > Step Properties > Checkpoint Properties**. The Database Checkpoint Properties dialog box opens.
- 2** Modify the settings as described “Understanding the Database Checkpoint Properties Dialog Box” on page 581.

23

Checking XML

By adding XML checkpoints to your test, you can check the contents of individual XML data files or documents that are part of your Web application.

This chapter includes:

- About Checking XML on page 592
- Creating XML Checkpoints on page 594
- Updating the XML Hierarchy for XML Test Object Operation Checkpoints (for WebService Test Objects Only) on page 614
- Modifying XML Checkpoints on page 622
- Reviewing XML Checkpoint Results on page 622
- Using XML Objects and Methods to Enhance Your Test on page 623

About Checking XML

XML (Extensible Markup Language) is a meta-markup language for text documents that is endorsed as a standard by the World Wide Web Consortium (W3C). XML makes the complex data structures portable between different computer environments/operating systems and programming languages, facilitating the sharing of data.

XML files contain text with simple tags that describe the data within an XML document. These tags describe the data content, but not the presentation of the data. Applications that display an XML document or file use either Cascading Style Sheets (CSS) or XSL Formatting Objects (XSL-FO) to present the data.

You can verify the data content of XML files with XML checkpoints. A few common uses of XML checkpoints are described below:

- An XML file can be a static data file that is accessed to retrieve commonly used data for which a quick response time is needed—for example, country names, zip codes, or area codes. Although this data can change over time, it is normally quite static. You can use an XML file checkpoint to validate that the data has not changed from one application release to another.
- An XML file can consist of elements with attributes and values (character data). There is a parent and child relationship between the elements, and elements can have attributes associated with them. If any part of this hierarchy (including data) changes, your application's ability to process the XML file may be affected. Using an XML checkpoint, you can check the content of an element to make sure that its tags, attributes, and values have not changed.
- Web service operations often return XML values. If the Web Services Add-in is installed on your computer, you can send a Web service operation command to the service and use an XML checkpoint to verify that the service returns the XML in the expected structure and with the expected values.

- XML files are often an intermediary that retrieves dynamically changing data from one system. The data is then accessed by another system using Document Type Definitions (DTD), enabling the accessing system to read and display the information in the file. You can use an XML checkpoint and parameterize the captured data values if you want to check an XML document or file whose data changes in a predictable way.
- XML documents and files often need a well-defined structure to be portable across platforms and development systems. One way to accomplish this is by developing an XML schema, which describes the structure of the XML elements and data types. You can use schema validation to check that each item of content in an XML file adheres to the schema description of the element in which the content is to be placed.

Note: XML checkpoints are compatible with namespace standards. A change in namespace between expected and actual values results in a failed checkpoint.

For more information on XML standards, see <http://www.w3.org/XML/>

For more information on namespace standards, see <http://www.w3.org/TR/1999/REC-xml-names-19990114/>

Creating XML Checkpoints

You can perform checkpoints on XML documents contained in Web pages or frames, on XML files, and on test objects that support XML. An XML checkpoint is a verification point that compares a current value for a specified XML element, attribute and/or value with its expected value. When you insert a checkpoint, QuickTest adds a checkpoint step in the Keyword View and adds a `Check CheckPoint` statement in the Expert View. When you run the test, QuickTest compares the expected results of the checkpoint to the current results. If the results do not match, the checkpoint fails.

After you run your test, you can view summary results of the XML checkpoint in the Test Results window. You can also view detailed results by opening the XML Checkpoint Results window. For more information, see Chapter 33, “Viewing Run Session Results.”

You can create three types of XML checkpoints:

- **XML Web Page/Frame Checkpoint.** Checks an XML document within a Web page or frame.
- **XML File Checkpoint.** Checks a specified XML file.
- **XML Test Object Checkpoint.** Checks the XML data for an object or operation.

Creating XML Checkpoints for Web Pages and Frames

You can create an XML checkpoint for any XML document contained in a Web page or frame using the **XML Checkpoint (From Application)** option. You can create an **XML Checkpoint (From Application)** only while recording.

To create an XML Checkpoint on XML contained in a Web page or frame:

- 1 Begin recording your test.



- 2 Select **Insert > Checkpoint > XML Checkpoint (From Application)**, or click the **Insert Checkpoint or Output Value** toolbar button and select **XML Checkpoint (From Application)**.

Note: The **XML Checkpoint (From Application)** option is available only when the Web Add-in is installed and loaded. For more information on loading add-ins, see the section on working with QuickTest add-ins in the *HP QuickTest Professional Add-ins Guide*.

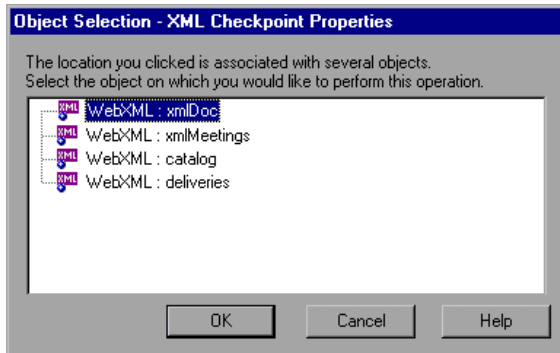
You can also insert a Web page or frame checkpoint using the **XML (From Resource)** option by selecting an existing WebXML test object as long as the actual XML object is currently available (open in the browser). For more information, see “Creating XML Test Object Checkpoints” on page 603.

The QuickTest window is hidden, and the pointer changes into a pointing hand. For more information about using the pointing hand feature, see “Tips for Using the Pointing Hand” on page 599.

- 3 Click a Web page or frame that contains XML documents.

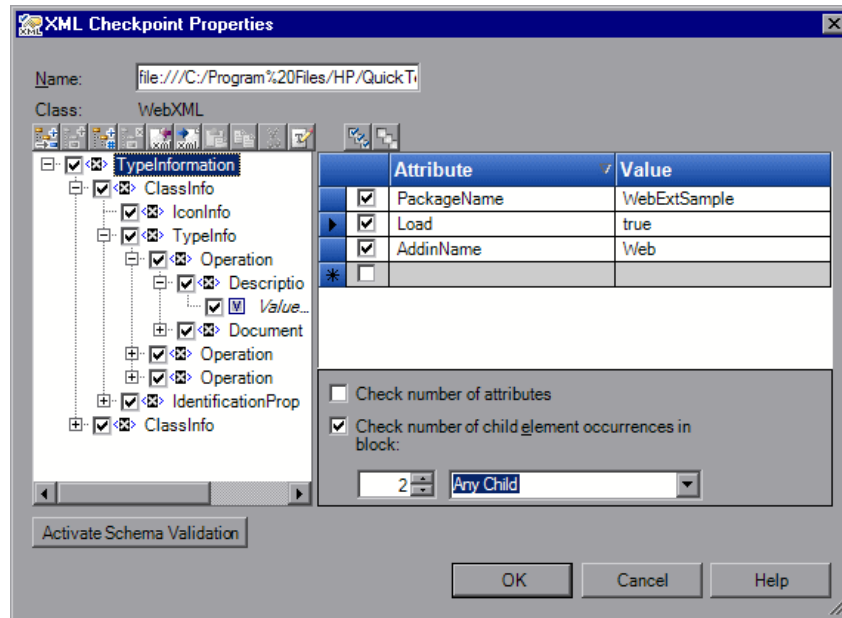
If only one XML file is associated with the Web page or frame, the XML Checkpoint Properties dialog box opens. In this case, proceed to step 5.

If more than one XML document is associated with the selected location, the Object Selection - XML Checkpoint Properties dialog box opens.



- 4 Select the XML document you want to check, and click **OK**.

The XML Checkpoint Properties dialog box opens and displays the root element of the selected XML document. You can expand the XML tree to view the element hierarchy and values (character data).



Note: If the XML source on which you base your checkpoint is in a valid XML format, but not to W3 standards, an error message informs you that the XML tree in the dialog box will be displayed in read-only format and that you must fix the XML source using the Edit XML as Text dialog box. For more information on this dialog box, see “Understanding the Edit XML as Text Dialog Box” on page 613.

5 In the **Name** box, either accept the name that QuickTest assigns to the checkpoint or specify another name for it. By default, the checkpoint name is the name of the test object on which the checkpoint is being performed.

If you rename the checkpoint, make sure that the name:

- is unique
- does not begin or end with a space
- does not contain " (double quotation mark)
- does not contain the following character combinations:
:=
@@

6 Select the items to check for the checkpoint. For more information, see “Understanding the XML Checkpoint Properties Dialog Box” on page 607.

7 When you finish setting your checkpoint preferences, click **OK** to add the XML checkpoint. A checkpoint similar to the following is added to the Keyword View.

Item	Operation	Value
▼ Action1		
▼ Simple XML Example		
▼ Simple XML Example		
▼ contents		
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")
```

Tips for Using the Pointing Hand

- You can hold the left CTRL key to change the pointing hand to a standard pointer. You can then change the window focus or perform operations in QuickTest or in your application, such as right-clicking, using the scroll bars, or moving the pointer over an object to display a context menu.
- If the window containing the object you want to select is partially hidden by another window, hold the pointing hand over the partially hidden window for a few seconds until it comes to the foreground. Then point to and click the required object. You can configure the length of time required to bring a window into the foreground using the General pane of the Options dialog box.
- If the window containing the object you want to select is fully hidden by another window, or if a dialog box is hidden behind a window, press the left CTRL key and arrange the windows as needed.
- If the window containing the object you want to select is minimized, you can display it by holding the left CTRL key, right-clicking the application in the Windows task bar, and choosing **Restore** from the context menu.
- If the object you want to select can be displayed only by performing an event (such as right-clicking or moving the pointer over an object to display a context menu), hold the left CTRL key. The pointing hand temporarily turns into a standard pointer and you can perform the event. When the object you want to select is displayed, release the left CTRL key. The pointer becomes a pointing hand again.

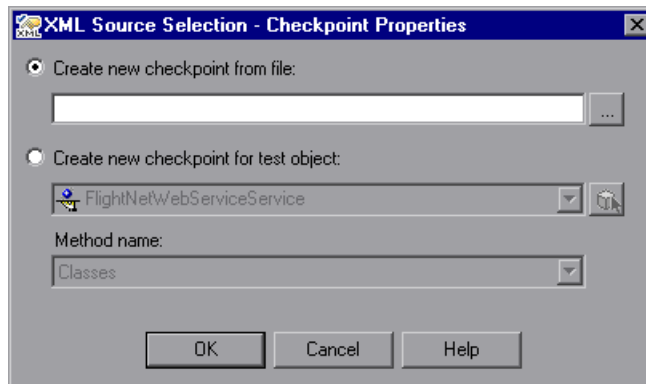
Creating XML File Checkpoints

You create XML file checkpoints to directly access and verify specific XML files in your system. You can create an XML file checkpoint while you are recording or editing your test.

To create an XML file checkpoint:



- 1 Select **Insert > Checkpoint > XML Checkpoint (From Resource)** or click the **Insert Checkpoint or Output Value** toolbar button, and select **XML Checkpoint (From Resource)**. The XML Source Selection - Checkpoint Properties dialog box opens.



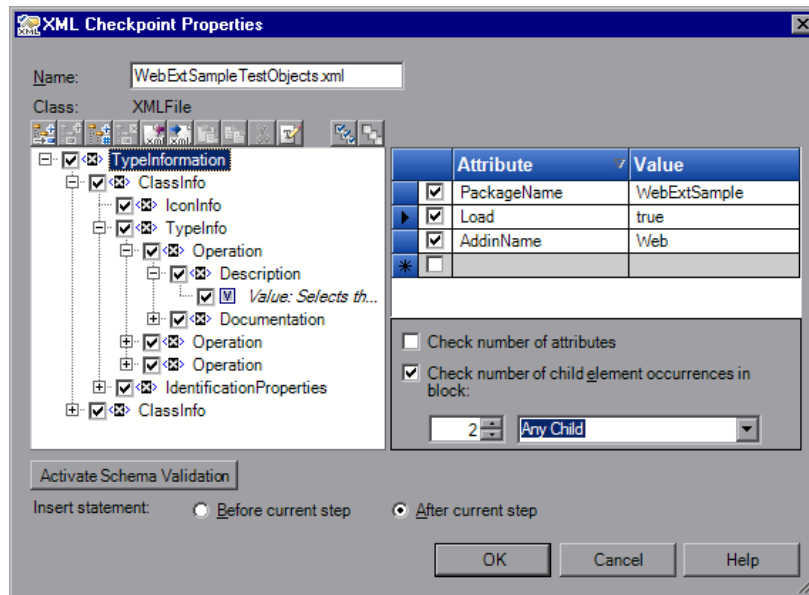
Tip: You can also insert an XML file checkpoint by selecting an existing XMLFile test object as long as the source file for the test object exists in the location stored with the test object. For more information, see “Creating XML Test Object Checkpoints” on page 603.

- 2 Select **Create new checkpoint from file**. Enter the file path or Internet address of the XML file.

Alternatively, click the browse button to open the Open XML File dialog box. In the sidebar, select the location of the XML file, and then navigate to the XML file for which you want to create a checkpoint. You can specify an XML file either from your file system or from Quality Center. Select the file and click **Open**. The file path and name are entered in the box.

Note: If you enter a relative path, QuickTest searches for the XML file in the folders listed in the Folders pane of the Options dialog box. After QuickTest locates the file, it saves it as an absolute path and uses the absolute path during the run session. For more information, see “Setting Folder Testing Options” on page 1237.

- 3 Click **OK** in the XML Source Selection - Checkpoint Properties dialog box. The XML Checkpoint Properties dialog box opens and displays the root element of the selected XML file. You can expand the XML tree to view the element hierarchy and values (character data).



Note: If the XML source on which you base your checkpoint is in a valid XML format, but not to W3 standards, an error message informs you that the XML tree in the dialog box will be displayed in read-only format and that you must fix the XML source manually using the Edit XML as Text dialog box. For more information on this dialog box, see “Understanding the Edit XML as Text Dialog Box” on page 613.

- 4 In the **Name** box, either accept the name that QuickTest assigns to the checkpoint or specify another name for it. By default, the checkpoint name is the name of the test object on which the checkpoint is being performed.

If you rename the checkpoint, make sure that the name:

- is unique
- does not begin or end with a space
- does not contain " (double quotation mark)
- does not contain the following character combinations:
:=
@@

- 5 Select the items to check for the checkpoint. For more information, see “Understanding the XML Checkpoint Properties Dialog Box” on page 607.
- 6 When you finish setting your checkpoint preferences, click **OK** to add the XML checkpoint. A checkpoint similar to the following is added to the Keyword View.

Item	Operation	Value	Documentation
▼ Action1			
FlightNetWebServiceService	AvailableCities		<No documentation summary is av.
avalcities.xml	Check	CheckPoint("avalcities.xml")	Check whether the content of the

QuickTest inserts this step as follows in the Expert View:

```
XMLFile("avalcities.xml").Check CheckPoint("avalcities.xml")
```

Creating XML Test Object Checkpoints

You can create an XML test object checkpoint to check the elements, attributes and/or values of XML associated with the selected test object. For example, you can check the XML that is returned from an operation performed on a Web service. You can create an XML test object checkpoint while you are recording or editing your test.

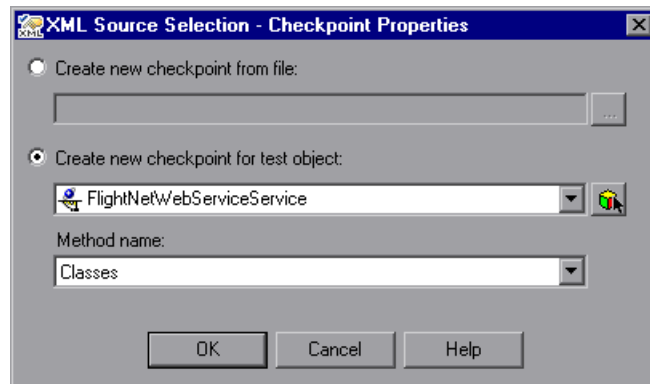
Note: You cannot insert an XML checkpoint from the Active Screen.

To create an XML test object checkpoint:



- 1 Select **Insert > Checkpoint > XML Checkpoint (From Resource)**, or click the **Insert Checkpoint or Output Value** toolbar button and select **XML Checkpoint (From Resource)**.

The XML Source Selection - Checkpoint Properties dialog box opens.



- 2 Select **Create new checkpoint for test object** and select the test object you want to check.



To select an object that is not displayed in the list, click **Object from Repository**. Then select an XML test object from the object repository on which to create a new checkpoint. The selected object must support XML. For information on selecting objects, see “Selecting an Object from the Repository or Application” on page 788.

You can select an existing WebXML or XMLFile test object type as long as the actual XML object is currently available (in an open browser or in the file system, as relevant) or, if you are working with the QuickTest Web Services Add-in you can select a WebService test object.

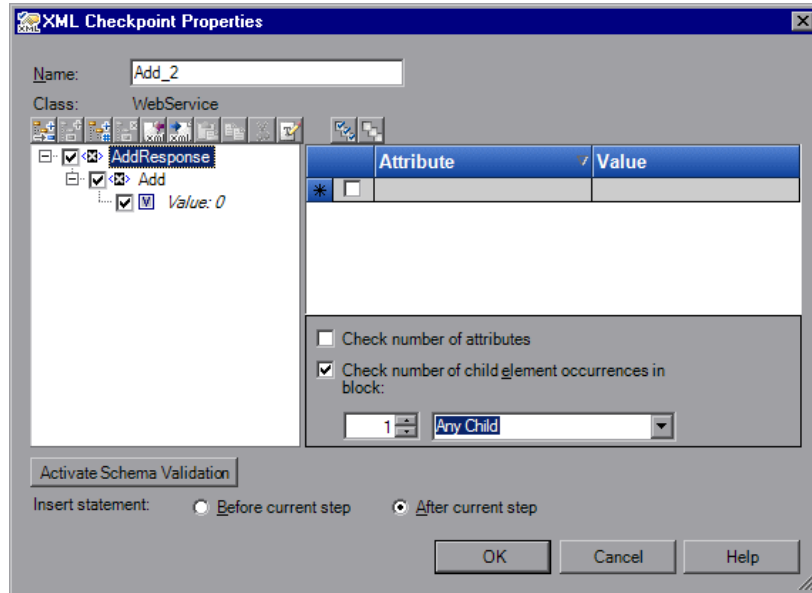
Note: Selecting a WebXML or XMLFile test object is identical to using the **XML Checkpoint (From Application)** or **Create new checkpoint from file** options, but may be faster than browsing to these objects and can be inserted while recording or editing. However, to use this option, the XML source must be available when you select the test object (the Web page must be open or the file must exist in the same location as when the test object was defined).

- 3** If you select a WebService test object, then the **Method name** box is enabled. Select the Web service operation whose return values you want to check.

Notes:

- The **Method name** box is available only if the Web Services Add-in is installed and loaded. The **Method name** box is enabled only if you select a WebService test object.
 - XML Checkpoints on Web service operations compare the expected values of the checkpoint to the actual values returned from the last native Web service operation performed on the test object. If a different Web service operation step is performed prior to the checkpoint, then the checkpoint fails.
-

- 4 Click **OK**. The XML Checkpoint Properties dialog box opens and displays the root element of the selected XML test object. You can expand the XML tree to view the element hierarchy and values (character data).



Note: When you create an XML checkpoint for a test object operation return value (for a WebService test object), only a generic XML tree is created and shown in the XML Checkpoint Properties dialog box. The data that is expected when each operation is called during the test is not included. You must update the XML hierarchy by populating the XML tree with the actual elements, attributes, and values you want to check. For more information, see “Updating the XML Hierarchy for XML Test Object Operation Checkpoints (for WebService Test Objects Only)” on page 614.

- 5 In the **Name** box, either accept the name that QuickTest assigns to the checkpoint or specify another name for it. By default, the checkpoint name is the name of the test object on which the checkpoint is being performed.

If you rename the checkpoint, make sure that the name:

- is unique
- does not begin or end with a space
- does not contain " (double quotation mark)
- does not contain the following character combinations:
:=
@@

- 6 Select the items to check for the checkpoint. For more information, see “Understanding the XML Checkpoint Properties Dialog Box” on page 607.
- 7 When you finish setting your checkpoint preferences, click **OK** to add the XML checkpoint. A checkpoint similar to the following is added to the Keyword View.

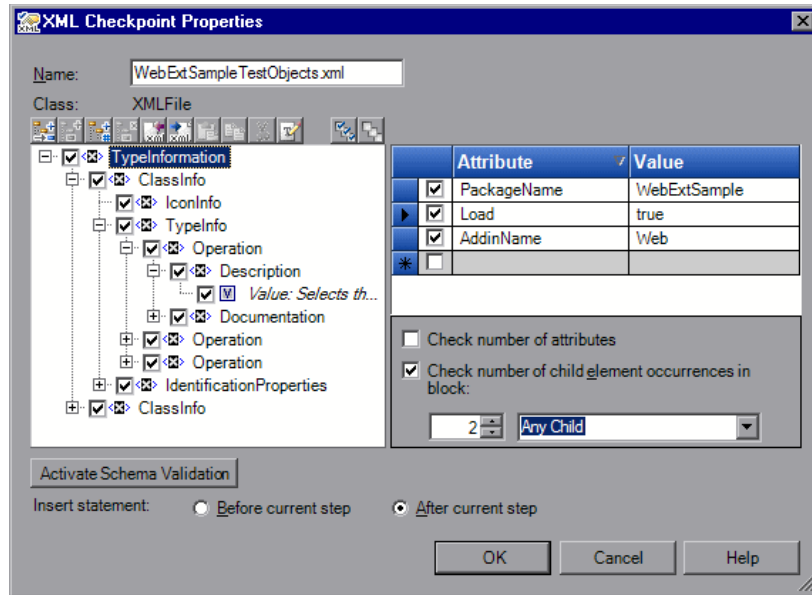
▼	Action1		
	FlightNetWebService	Airlines	
	FlightNetWebService	Check	CheckPoint("Airlines")

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 elements, attributes, and values in the XML tree




In the XML tree, select the check boxes of the elements, attributes, and/or values that you want to check. For each element you want to check, select the checks you want to perform. For each attribute or value you want to check, select the checks you want to perform, or the parameterization options you want to set.

Identifying the Object











The top part of the dialog box displays test object information on the test object for which you are creating a checkpoint:





Option	Description
Name	<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:</p> <ul style="list-style-type: none">➤ is unique➤ does not begin or end with a space➤ does not contain " (double quotation mark)➤ does not contain the following character combinations: := @@
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>
Find in Repository button 	<p>Displays the checkpoint in its object repository.</p> <p>Note: This option is available only when editing an existing checkpoint. It is not available when creating a new checkpoint.</p>



Modifying the XML Tree

The following commands are available according to the node you select in the tree:

Command	Icon	Description
Add Child		Adds a child node below the selected node in the tree.
Insert Sibling		Adds a sibling node at the same level as the selected node in the tree.
Add Value		Enables you to assign a constant or parameterized value to the selected element.
Delete		Deletes the selected node. Note that you cannot delete the root node of the checkpoint.
Import XML		Enables you to browse to an existing XML file and import it. The new file replaces the selected node and its current sub-tree.
Export XML		Enables you to save the content of the checkpoint tree to an XML file. Enabled only when the root node of the tree is selected.
Paste		Pastes a cut or copied node as a child node below the selected node in the XML tree. Note: You cannot paste an XML element node as its own descendant.
Copy		Makes a copy of the selected node, which you can then paste in another location in the XML tree.
Cut		Prepares the selected node to be cut and copies it to the clipboard. When you paste the node in the new location, it is removed from the original location in the XML tree.
Edit XML as Text		Opens the Edit XML as Text dialog box, enabling you to modify the XML text of the selected node and its subnodes in a text editor. For more information, see “Understanding the Edit XML as Text Dialog Box” on page 613.

Command	Icon	Description
Select All		Selects all element and value nodes in the XML tree as well as all element attributes.
Clear All		Clears all element and value nodes in the XML tree as well as all element attributes.
Duplicate		Adds a new node, identical to the selected one, as a sibling node at the same level as the selected node in the XML tree. Note: This command is available only from the context menu (right-click menu).

XML Tree

The XML tree displays the hierarchical relationship between each element and value in the XML tree, enabling you to select the specific elements, attributes and values you want to check. Each element is displayed with a  icon. Each value is displayed with a  icon.

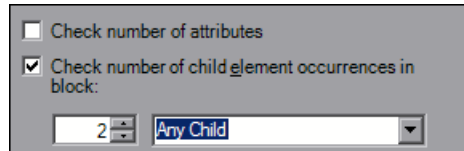
Select the check box next to an element or value node to include that item in the checkpoint. Select an element node in the XML tree to display, edit, or parameterize its expected attributes and values on the right of the XML Checkpoint Properties dialog box. Select a value node in the XML tree to display, edit, or parameterize its expected value on the right of the XML Checkpoint Properties dialog box.

Tip: The XML tree pane and the **Attribute** and **Value** columns in the right pane are resizable.

Checkpoint Options

The checkpoint options area on the bottom right of the XML Checkpoint Properties dialog box enables you to select the types of checks you want to perform on selected elements.

When you select an element in the XML Tree, the checkpoint options area includes the name of the selected element and the available element checks.



☐ Check number of attributes

☒ Check number of child element occurrences in block:

2 Any Child

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 schemas embedded within your XML document. For more information, see “Understanding the Schema Validation Dialog Box” on page 618.

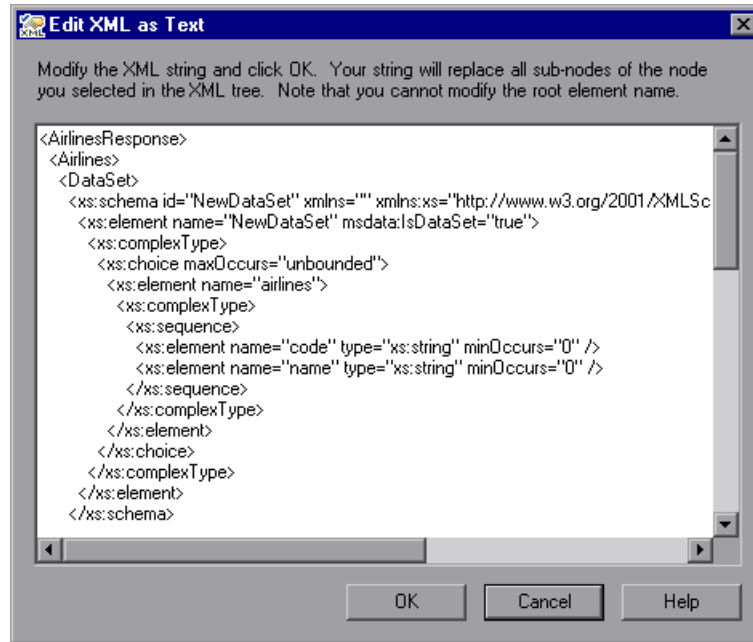
Insert Statement Options

If you are inserting a checkpoint while editing your test, the bottom part of the XML Checkpoint Properties dialog box displays **Insert statement** options, enabling you to choose whether you want to insert the XML checkpoint before or after the step that you selected. Select **Before current step** if you want to check the value of the text before the highlighted step is performed. Select **After current step** (default) if you want to check the value of the text after the highlighted step is performed.

Note: The **Insert statement** options are not available if you are adding an XML checkpoint while recording or if you are modifying an existing XML checkpoint. They are available only if you are adding a new XML checkpoint to an existing test.

Understanding the Edit XML as Text Dialog Box

The Edit XML as Text dialog box enables you to edit XML content from the XML tree in a text editor.



This dialog box is used mainly for constructing an entire XML segment from a string or for fixing syntax problems that prevent the dialog box from displaying the XML tree correctly. It is also useful when you want to use copy-paste functionality to edit the tree.

When you click **OK** in the Edit XML as Text dialog box, the sub-tree of the node you previously selected in the XML tree (or the entire tree if no node was selected or if the root node was selected) is completely replaced by the XML content from the Edit XML as Text dialog box.

Note: You cannot modify the name of the root element displayed in the Edit XML as Text dialog box.

Updating the XML Hierarchy for XML Test Object Operation Checkpoints (for WebService Test Objects Only)

This section is relevant only when working with XML Checkpoints on WebService test object operations (with the QuickTest Professional Web Services Add-in).

When you create an XML checkpoint for a test object operation (for a WebService test object), the expected operation return value data cannot be generated. Therefore, only a generic XML tree is created. To check the operation return values, you must first populate the XML tree with the actual elements, attributes, and values that the operation is expected to return.


You can use one of the three methods below to populate the XML tree:

- Updating the XML Tree Manually
- Importing an XML Tree from a File
- Updating the XML Tree Using Update Run Mode

Updating the XML Tree Manually

You can update the XML tree by adding elements, attributes, and values manually in the XML Checkpoint Properties dialog box.

To update the XML tree manually:

- 1 In the Keyword View, select the checkpoint whose XML tree you want to update. Click in the **Value** cell.
- 2  Click the **Checkpoint Properties** button or right-click and select **Checkpoint Properties**. The XML Checkpoint Properties dialog box opens.
- 3 Select a node in the XML tree and then click a toolbar button or select an option from the context (right-click) menu to:
 - Add an element at the same level as the selected node.
 - Add an element below the selected node.
 - Add a value to the selected node.
 - Copy the selected node.





- Cut the selected node (the selected node is removed only after you paste it in another location).



- Paste a cut or copied node as a child node below the selected node.



- Edit the XML text of the selected node.



- Delete the selected node.



- Duplicate the selected node, adding an identical node as a sibling node at the same level. (This command is available only from the right-click context menu.)

For more information on the available tools in the XML Checkpoint Properties dialog box, see “Understanding the XML Checkpoint Properties Dialog Box” on page 607.

Importing an XML Tree from a File

You can import an XML tree from an existing file for a specific element in the XML tree hierarchy or for the whole tree.

To import an existing XML tree from a file:

- 1 In the Keyword View, select the checkpoint whose XML tree you want to update.
- 2  Click in the **Value** cell and then click the **Checkpoint Properties** button. The XML Checkpoint Properties dialog box opens.
- 3 If you want to import an XML hierarchy for the whole XML tree, select the root node. If you want to import an XML hierarchy for a specific element, select the element in the XML tree hierarchy.
- 4  Click the **Import XML** button. A message warns you that the imported hierarchy overwrites the selected node and its sub-tree. Click **Yes** to close the message.
- 5 In the Import XML from File dialog box, browse to the required XML file and click **Open**. The XML hierarchy is imported from the file.
- 6 If required, configure a constant or parameterized value for each of the element and value nodes in the XML tree. For more information on parameterizing values, see Chapter 24, “Parameterizing Values.”

Updating the XML Tree Using Update Run Mode

QuickTest cannot generate the expected return values of an operation when you insert an XML checkpoint on a Web service operation, but it can generate this information after it runs the operation. Therefore, you can run your Web service test in Update Run mode to automatically populate or update the elements, attributes and values in your XML tree.

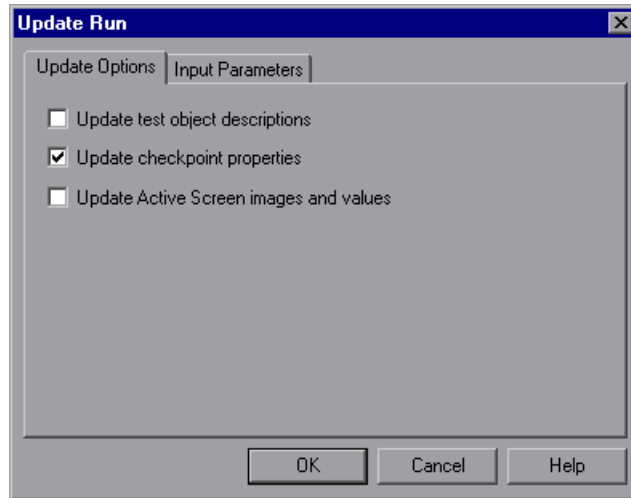
To generate a new XML tree based on the current return values of the Web service operation, ensure that none of the node, attribute, or value check boxes are selected in the XML checkpoint.

To maintain the current hierarchy in the XML tree and update only the expected values, select one or more node, attribute, or value check boxes in the dialog box.

Note: XML Checkpoints on Web service operations check the actual values returned from the last native Web service operation performed on the test object. If a different Web service operation step is performed prior to the checkpoint, then the update run cannot update the XML tree for that operation.

To update an XML tree using Update Run mode:

- 1 Open a test containing XML Test Object checkpoints for Web service operations.
- 2 Click the down arrow next to the **Run** button in the toolbar and select **Update Run Mode** or select **Automation > Update Run Mode**. The Update Run dialog box opens.



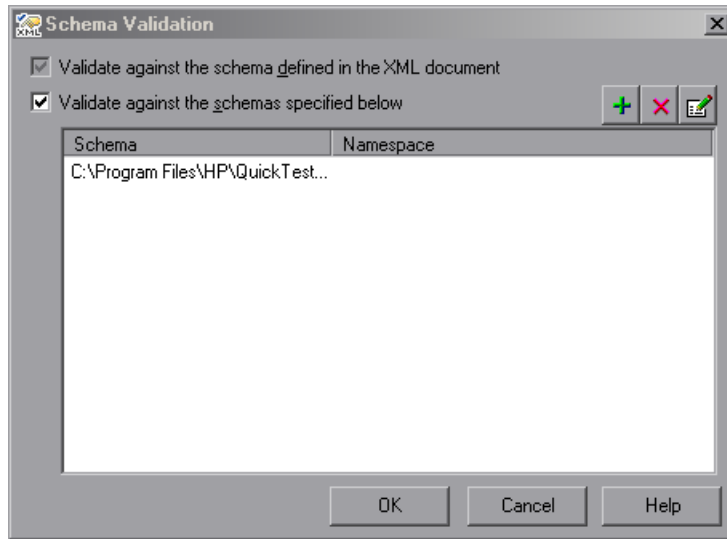
- 3 Select **Update checkpoint properties** and click **OK**. QuickTest runs the test and updates the XML hierarchy for each XML checkpoint.



- 4 If you want to confirm that QuickTest successfully updated your checkpoint, expand the tree in the Test Results window and select the XML checkpoint. Then check that Update done is displayed in the right-hand pane. (If the Test Results window did not open automatically at the end of the run, click the **Results** button or select **Automation > Results**.)

Understanding the Schema Validation Dialog Box

The Schema Validation dialog box enables you to specify an XML schema against which you want to validate the hierarchy of the XML in your application or file.



The Schema Validation dialog box contains the following options:




- **Validate against the schema defined in the XML document.** Instructs QuickTest to use the schema or schemas defined within your XML document to validate the hierarchy of the XML in your Web page/frame, XML file, or XML test object.
- **Validate against the schemas specified below.** Instructs QuickTest to use one or more external XML schema files to validate the hierarchy of your XML. If you select this option, any schemas defined within your XML document are also checked. (The **Validate against the schema defined in the XML document** is automatically selected and is disabled.)

When you select the **Validate against the schemas specified below** option, the **Add Schema** button is enabled. Clicking this button opens the Add Schema dialog box, in which you can specify the following:

- **Schema path or URL.** Enter the path or URL of your XML schema file. Alternatively, click the browse button to navigate to the XML schema you want to use to validate the XML in your Web page/frame, XML file, or XML test object. You can specify schema files either from your file system or from Quality Center. For each external file you add, you must specify its path or URL and namespace.
- **Schema namespace.** (If applicable.) If your schema file has a namespace, specify it. QuickTest checks that the namespace matches the schema file as part of the validation process. If the schema file has a namespace and you do not specify it, or if the namespace you specify is different to the one specified in the schema file, the validation fails.

Click **OK** in the Add Schema dialog box to add the selected schema to the list in the Schema Validation dialog box. Click the **Add Schema** button again if you want to add another schema.

If you select **Use external schemas**, the following toolbar buttons are enabled when appropriate:

Button	Description
	Enables you to add an external schema file to the list. For more information, see “Understanding the Add Schema Dialog Box” on page 621.
	Enables you to modify the details of the selected external schema file in the list. For more information, see “Understanding the Edit Schema Dialog Box” on page 621.
	Enables you to remove the selected external schema file from the list.

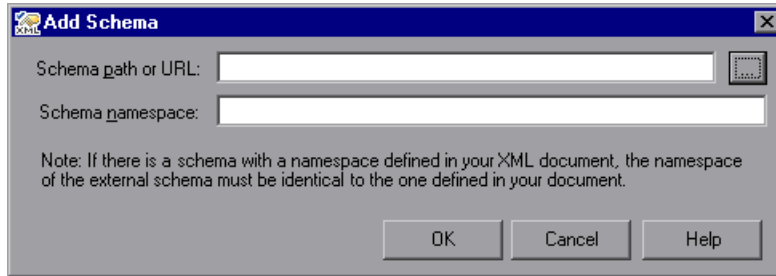
Guidelines for Schema Validation

Following are specific guidelines to consider when specifying a schema file to validate your XML.

- If you are validating an XML file using a schema defined in the XML file, the schema can be defined with an absolute or relative path. When you specify a relative path, QuickTest searches for the schema in the folders listed in the Folders pane of the Options dialog box. For more information, see “Setting Folder Testing Options” on page 1237.
- If you are validating an XML document located on the Web with a schema file located on your file system, you cannot use UNC format (for example, \\ComputerName\Path\To\Schema) to specify the schema file location. Instead, map the schema file location to a network drive.
- If there is a schema with a namespace defined in your XML document, the namespace of the external schema must be identical to the one defined in your document. Using an external XML schema file to validate an XML document may cause an unexpected result if the XML document has an XML schema declaration, and the namespace in the external schema file and the schema defined in the document are not identical.
- When you perform a schema validation, QuickTest validates all of the elements in the XML document, even if certain XML elements are not associated with a schema file. Any XML elements that are not associated with a schema file cause the schema validation to fail.

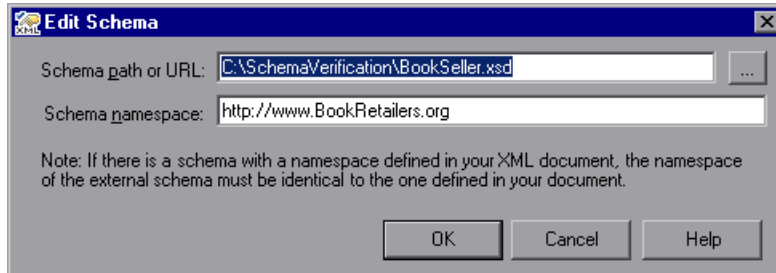
Understanding the Add Schema Dialog Box

The Add Schema dialog box enables you to specify the path or URL of an external schema file and its namespace. If there is a schema with a namespace defined in your XML document, the namespace of the external schema must be identical to the one defined in your document.



Understanding the Edit Schema Dialog Box

The Edit Schema dialog box displays the path and namespace of the schema file you selected in the list. You can modify the path or URL of the selected schema file, and its namespace.



Modifying XML Checkpoints

You can change the expected data and settings of an existing XML checkpoint.

To modify an XML checkpoint:

- 1 In the Keyword View or the Expert View, right-click the XML checkpoint that you want to modify and select **Checkpoint Properties**. Alternatively, select the step containing the XML checkpoint and select **Edit > Step Properties > Checkpoint Properties**. The XML Checkpoint Properties dialog box opens.
- 2 Modify the settings as described in the previous sections.

Reviewing XML Checkpoint Results

By adding XML checkpoints to your tests, you can verify that the data and structure in your XML documents, XML files, or XML test objects have not changed unexpectedly. When you run your test, QuickTest compares the expected results of the checkpoint to the actual results of the run session. If the results do not match, the checkpoint fails.

You can view summary results of the XML checkpoint in the Test Results window. You can view detailed results by opening the XML Checkpoint Results window. For more information on XML checkpoint results, see “Analyzing XML Checkpoint Results” on page 1037.

Note: XML Checkpoints on Web service operations compare the expected values of the checkpoint to the actual values returned from the last native Web service operation performed on the test object. If a different Web service operation step is performed prior to the checkpoint, then the checkpoint fails.

Using XML Objects and Methods to Enhance Your Test

QuickTest provides several scripting methods that you can use with XML data. You can use these scripting methods to retrieve data and return new XML objects from existing XML data. You do this by using the XMLUtil, or WebXML objects to return XML data and then using the supported XMLData objects and methods to manipulate the returned data.

Tip: All XMLData objects and methods are compatible with namespace and XPath standards.

For more information on XML standards, see <http://www.w3.org/XML/>

For more information on namespace standards, see <http://www.w3.org/TR/1999/REC-xml-names-19990114/>

For more information on XPath standards, see <http://www.w3.org/TR/1999/REC-xpath-19991116>

For more information on programming in the Expert View, see Chapter 29, “Working in the Expert View and Function Library Windows.” For more information on XML objects and methods, see the Supplemental section of the *HP QuickTest Professional Object Model Reference*.

24

Parameterizing Values

QuickTest enables you to expand the scope of a basic test by replacing fixed values with parameters. This process, known as **parameterization**, greatly increases the power and flexibility of your test.

This chapter includes:

- About Parameterizing Values on page 626
- Parameterizing Values in Steps and Checkpoints on page 628
- Using Test and Action Input Parameters on page 635
- Using Data Table Parameters on page 639
- Using Environment Variable Parameters on page 645
- Using Random Number Parameters on page 655
- Example of a Parameterized Test on page 657
- Using the Data Driver to Parameterize Your Test on page 662

About Parameterizing Values

You can use the parameter feature in QuickTest to enhance your test by parameterizing the values that it uses. A **parameter** is a variable that is assigned a value from an external data source or generator.

You can parameterize the values in steps or the values of action parameters using one of the following parameter types:

- **Test/action parameters.** Test parameters enable you to use values passed from your test. Action parameters enable you to pass values from other actions in your test.

To use a value within a specific action, you must pass the value down through the action hierarchy of your test to the required action. You can then use that parameter value to parameterize a step in your test. For example, suppose that Action3 is a nested action of Action1 (a top-level action), and you want to parameterize a step in Action3 using a value that is passed into your test from the external application that runs (calls) the test. You can pass the value from the test level to Action1, then to Action3, and then parameterize the required step using this action input parameter value (that was passed through from the external application).

Alternatively, you can pass an output action parameter value from an action step to a later sibling action at the same hierarchical level. For example, suppose that Action2, Action3, and Action4 are sibling actions at the same hierarchical level, and that these are all nested actions of Action1. You can parameterize a call to Action4 based on an output value retrieved from Action2 or Action3. You can then use these parameters in your action step.

For more information, see “Guidelines for Working with Action Parameters” on page 479.

- **Data Table parameters.** Enable you to create a **data-driven** test (or action) that runs several times using the data you supply. In each repetition, or **iteration**, QuickTest uses a different value from the Data Table.

For example, suppose your application includes a feature that enables users to search for contact information from a membership database. When the user enters a member's name, the member's contact information is displayed, together with a button labelled **View <MemName>'s Picture**, where **<MemName>** is the name of the member. You can parameterize the name property of the button using a list of values so that during each iteration of the run session, QuickTest can identify the different picture buttons.

- **Environment variable parameters.** Enable you to use variable values from other sources during the run session. These may be values you supply, or values that QuickTest generates for you based on conditions and options you choose.

For example, you can have QuickTest read all the values for filling in a Web form from an external file, or you can use one of QuickTest's built-in environment variables to insert current information about the computer running the test.

- **Random number parameters.** Enable you to insert random numbers as values in your test. For example, to check how your application handles small and large ticket orders, you can have QuickTest generate a random number and insert it in a **number of tickets** edit box.

Tips:

- If you want to parameterize the same value in several steps in your test, you may want to consider using the Data Driver rather than adding parameters manually. For more information see, "Using the Data Driver to Parameterize Your Test" on page 662.
 - You can also parameterize identification property values of test objects in the object repository using repository parameters. For more information, see "Working with Repository Parameters" on page 228.
-

Parameterizing Values in Steps and Checkpoints

You can parameterize values in steps and checkpoints while working with your test.

You can parameterize the values of object properties for a selected step. You can also parameterize the values of the operation arguments defined for the step.

For example, your application may include a form with an edit box into which the user types the user name. You may want to test whether your application reads this information and displays it correctly in a dialog box. You can insert a text checkpoint that uses the built-in environment variable for the logged-in user name, to check whether the displayed information is correct.

Note: When you parameterize the value of an object property for a local object, you are modifying the test object description in the local object repository. Therefore, all occurrences of the specified object within the action are parameterized. For more information on the local object repository, see Chapter 5, “Managing Test Objects in Object Repositories.”

Parameterizing the value of a checkpoint property enables you to check how an application performs the same operation based on different data.

For example, if you are testing the Mercury Tours sample Web site, you may create a checkpoint to check that once you book a ticket, it is booked correctly. Suppose that you want to check that flights are booked correctly for a variety of different destinations. Rather than create a separate test with a separate checkpoint for each destination, you can add a Data Table parameter for the destination information. This enables you to create a list of different destinations. QuickTest will then check the flight information for a different destination for each iteration of the test.

For more information on using checkpoints, see Chapter 17, “Understanding Checkpoints.”

When you define a value as a parameter, you specify the parameter type and its settings.

For more information on using specific parameter types, see:

- “Using Test and Action Input Parameters” on page 635
- “Using Data Table Parameters” on page 639
- “Using Environment Variable Parameters” on page 645
- “Using Random Number Parameters” on page 655

For more information on parameterizing values:





- “Parameterizing Values for Operations” on page 629
- “Parameterizing Property Values for Objects and Checkpoints” on page 631


Tip: When you use the Step Generator to add new steps, you can parameterize the values for the operation you select. For more information, see “Inserting Steps Using the Step Generator” on page 777.

Parameterizing Values for Operations


If the method, property, or function used in the step has arguments, you can parameterize the argument values as required. For example, if the operation uses the **Click** method, you can parameterize the values for the **x** argument, the **y** argument, or both.

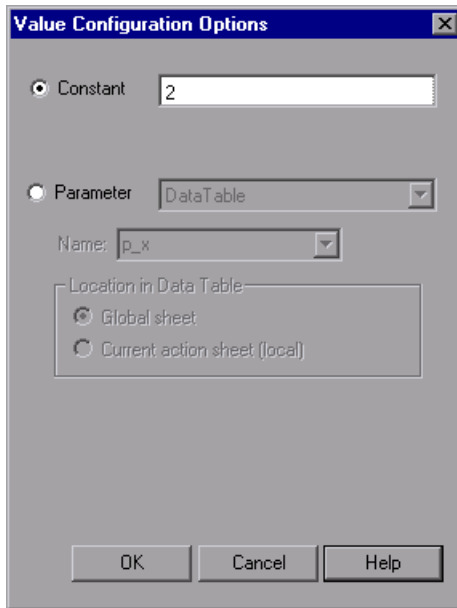
When you select a parameterized value in the Keyword View, the icon for the parameter type is displayed. For example, in the following segment, the value of the **Set** method has been defined as a random number parameter. QuickTest enters a random number value into the **creditnumber** edit box each time the test runs.

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.



The dialog box titled "Value Configuration Options" has a close button (X) in the top right corner. It contains two radio buttons: "Constant" (selected) and "Parameter". The "Constant" option has a text box next to it containing the value "2". The "Parameter" option has a dropdown menu next to it showing "DataTable". Below these, there is a "Name:" label followed by a dropdown menu showing "p_x". Underneath, there is a section titled "Location in Data Table" with two radio buttons: "Global sheet" (selected) and "Current action sheet (local)". At the bottom of the dialog box are three buttons: "OK", "Cancel", and "Help".

Note: The parameter options shown in this dialog box change according to the parameter type selected in the **Parameter** box.

- 3 Select **Parameter**. If the value is already parameterized, the **Parameter** section displays the current parameter definition for the value. If the value is not yet parameterized, the **Parameter** section displays the default parameter definition for the value. For more information, see “Understanding Default Parameter Values” on page 634.
- 4 Accept or change the parameter definition:
 - Click **OK** to accept the displayed parameter statement and close the dialog box.
 - Modify the value settings for the selected parameter type and click **OK**.
 - Change the parameter type. The options in the **Parameter** section change according to the parameter type you select.

For more information on configuring values for specific parameter types, see:



- “Defining the Settings for a Test or Action Parameter” on page 637
- “Defining the Settings for a Data Table Parameter” on page 642
- “Defining the Settings for an Environment Variable Parameter” on page 652
- “Defining Settings for a Random Number Parameter” on page 656

Parameterizing Property Values for Objects and Checkpoints


You can parameterize the values for one or more properties of an object stored in the local object repository in the Object Properties dialog box or Object Repository window. You can parameterize the values for one or more properties of a checkpoint in the Checkpoint Properties dialog box.

Note: For information on parameterizing a property value for an object in a shared object repository, see Chapter 7, “Managing Object Repositories.”

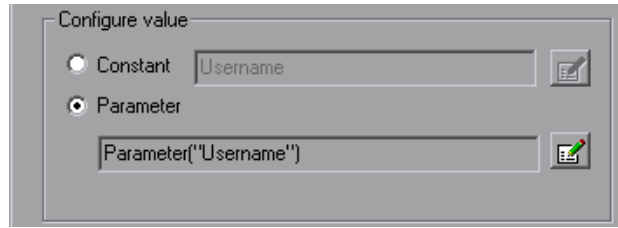
To parameterize local object values:

- 1** Open the dialog box for the object properties in one of the following ways:
 - Select a step and select **Edit > Step Properties > Object Properties**, or right-click a step and select **Object Properties**. The Object Properties dialog box opens.
 -  ➤ Open the **Object Repository** dialog box and select the object,
- 2** Click in the **Value** cell for the property that you want to parameterize, and click the parameterization icon . The Value Configuration Options dialog box opens.
- 3** Select **Parameter**. If the value is already parameterized, the **Parameter** box displays the current parameter definition for the value. If the value is not yet parameterized, the **Parameter** box displays the default parameter definition for the value. For more information, see “Configuring a Selected Value” on page 760.
- 4** Click **OK** to accept the displayed parameter statement or change the displayed parameter definition, and then click **OK**.
- 5** To accept the displayed parameter statement and parameterize another of the displayed values, select another property and follow the previous steps.

To parameterize checkpoint property values:

- 1** Open the dialog box for the checkpoint properties in one of the following ways:
 - Select **Edit > Step Properties > Checkpoint Properties**, or right-click the checkpoint and select **Checkpoint Properties**.
 -  ➤ Open the **Object Repository** dialog box and select the checkpoint.

- 2** In the **Configure value** area of the dialog box, select **Parameter**.



If the value is already parameterized, the **Parameter** box displays the current parameter definition for the value. If the value is not yet parameterized, the **Parameter** box displays the default parameter definition for the value. For more information, see “Understanding Default Parameter Values” on page 634.

- 3** Accept or change the displayed parameter definition:

- To accept the displayed parameter statement and close the dialog box, click **OK**.



- To change the parameter type or modify the value settings for the selected property, click the **Parameter Options** button. The Parameter Options dialog box opens for the displayed parameter type.

- 4** To accept the displayed parameter statement and parameterize another of the displayed values, select another property and follow the previous steps.

For more information on defining value settings for specific parameter types, see:

- “Setting Test and Action Parameter Options” on page 636
- “Setting Data Table Parameter Options” on page 641
- “Choosing Global or Action Data Table Parameters” on page 643
- “Using Random Number Parameters” on page 655

Understanding Default Parameter Values

When you select a value that has not yet been parameterized, QuickTest generates a default parameter definition for the value. The following table describes how the default parameter settings are determined:

When parameterizing	Condition	Default parameter type	Default parameter name
A value for a step or a checkpoint in an action	At least one input action parameter is defined in the current action	Action parameter	The first input parameter displayed in the Parameters tab of the Action Properties dialog box
An input action parameter value for a nested action	At least one input action parameter is defined for the action calling the nested action	Action parameter	The first input parameter displayed in the Parameters tab of the Action Properties dialog box of the calling action
An input action parameter value for a top-level action call	At least one input parameter is defined for the test	Test parameter	The first input parameter displayed in the Parameters pane of the Test Settings dialog box

If the relevant condition described above is not true, the default parameter type is Data Table. If you accept the default parameter details, QuickTest creates a new Data Table parameter with a name based on the selected value. Data Table parameters are created in the Global sheet.

For more information on Data Table sheets, see Chapter 42, “Working with Data Tables.”

Using Test and Action Input Parameters

You can parameterize a step using a test or action input parameter. This enables the step to use values that have been passed from the application that ran (called) your test. For example, you can use an input test parameter as the value for a method argument.

You can parameterize a value using a test or action parameter only if the parameter has been defined for the test or action. For more information on defining parameters, see “Defining Parameters for Your Test” on page 1280, “Setting Action Parameters” on page 472, and “Setting Action Call Parameter Values” on page 483.

You can parameterize steps by selecting input parameters in the Parameter Options or Value Configuration Options dialog box. The parameter options that are available in these dialog boxes depend on where you are currently located in your test, and whether test or action parameters are defined. For more information, see “Using Action Parameters” on page 476 and “Defining Parameters for Your Test” on page 1280.

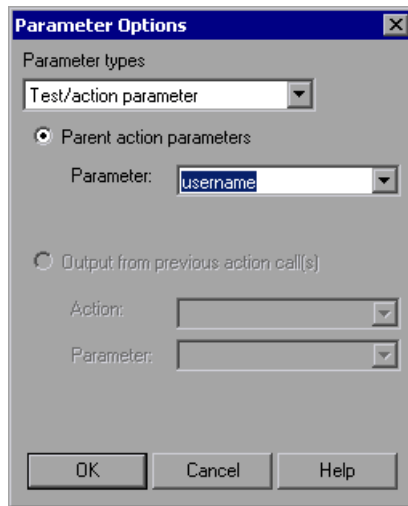
Alternatively, you can enter the parameter name in the Expert View using the Parameter utility object, in the format: `Parameter("ParameterName")` for the current action, or `Parameter("ActionName", "ParameterName")` to use the output parameter from a previous action as an input parameter in the current action. For more information, see “Using Action Parameters in Steps in the Expert View” on page 638.

Tip: You can also create test or action parameter output values that retrieve values during the run session and store them for use at another point in the run session. You can then use these output values to parameterize a step in your test. For more information, see “Outputting a Value to an Action Parameter” on page 684.

Setting Test and Action Parameter Options

When you choose to parameterize a value, the dialog box that opens enables you to select a parameter type and the parameter options to use. The image below shows the dialog box that opens when you select to parameterize a checkpoint expected value. The dialog boxes for parameterizing other value types such as argument values, object property values, and output storage locations provide similar options.

When **Test/action parameter** is selected as the parameter type, you can select the required parameter from a list of existing parameters.



Tip: When you open the dialog box to parameterize a value, the default parameter type may be set to **Test/action parameter**. For more information on default parameter type settings, see “Understanding Default Parameter Values” on page 634.

Defining the Settings for a Test or Action Parameter

The following options are available for configuring test or action parameters:

- **Test parameters** or **Parent action parameters**. Parameter defined in the test or parent action. (If no output parameters are defined in the test or parent action, this area is disabled.) **Test parameters** are available only for top-level actions. They are defined in the Parameters pane of the Test Settings dialog box. **Parent action parameters** are available for subsequent steps and for nested actions. They are defined in the action containing the steps or in the action that calls the nested action.
 - **Parameter**. Specifies the name of the input parameter. The read-only list of available parameters contains the names and full descriptions of the currently defined input parameters for the action. You can resize the display, as needed, and, if the list of parameters is long, you can scroll through the list.
- **Output from previous action call(s)**. Any previous action in the same hierarchical level for which output parameters are defined. (If no output parameters are defined in previous actions, this area is disabled.)
 - **Action**. Specifies the previous action from which you can choose an output parameter. You can choose any action in the list.
 - **Parameter**. Specifies the name of the output parameter. The read-only list of available parameters contains the names and full descriptions of the currently defined output parameters from the previous action(s). You can resize the display, as needed, and, if the list of parameters is long, you can scroll through the list.

You can also use test or action parameter variables using parameterization objects and methods in the Expert View. For more information, see the *HP QuickTest Professional Object Model Reference*.

Using Action Parameters in Steps in the Expert View

Instead of selecting input (or output) parameters from the appropriate dialog boxes while parameterizing steps or inserting output value steps, you can enter input and output parameters as values in the Expert View using the Parameter utility object in the format: `Parameter("ParameterName")`.

Suppose you have test steps that enter information in a form to display a list of purchase orders in a table, and then return the total value of the orders displayed in the table.

You can define input parameters, called **SoldToCode** and **MaterialCode**, for the codes entered in the **Sold to** and **Materials** edit boxes of the form so that the Orders table that is opened is controlled by the input parameter values passed when the test is called.

You can define an output parameter, called **TotalValue**, to store the returned value. The output value (**TotalValue**) could then be returned to the application that called the test.

The example described above might look something like this (parameters are in bold font):

```
Browser("Mercury").Page("List Of Sales").WebEdit("Sold to").
    Set Parameter("SoldToCode")
Browser("Mercury").Page("List Of Sales").WebEdit("Materials").
    Set Parameter("MaterialCode")
Browser("Mercury").Page("List Of Sales").WebButton("Enter").Click
NumTableRows = Browser("Mercury").Page("List Of Sales").
    WebTable("Orders").RowCount
Parameter("TotalValue") = Browser("Mercury").Page("List Of Sales").
    WebTable("Orders").GetCellData(NumTableRows,"Total")
```

Using Data Table Parameters

You can supply the list of possible values for a parameter by creating a Data Table parameter. Data Table parameters enable you to create a data-driven test, or action that runs several times using the data you supply. In each repetition, or **iteration**, QuickTest uses a different value from the Data Table (taken from the subsequent row in the Data Table).

For example, consider the Mercury Tours sample Web site, which enables you to book flight requests. To book a flight, you supply the flight itinerary and click the **Continue** button. The site returns the available flights for the requested itinerary.

You could conduct the test by accessing the Web site and submitting numerous queries. This is a slow, laborious, and inefficient solution. By using Data Table parameters, you can run the test for multiple queries in succession.

When you parameterize your test, you first create steps that access the Web site and check for the available flights for one requested itinerary.

You then substitute the existing itinerary with a Data Table parameter and add your own sets of data to the relevant sheet of the Data Table, one for each itinerary.

	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.

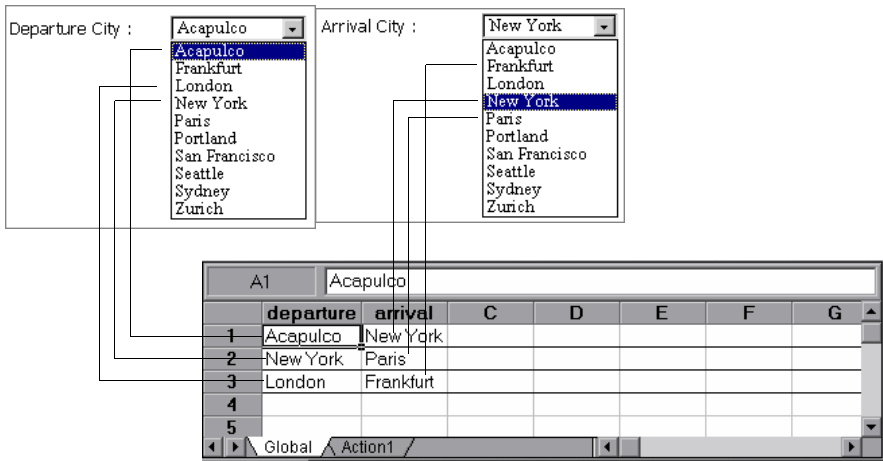
Each column in the table represents the list of values for a single Data Table parameter. The column header is the parameter name.

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 42, “Working with Data Tables.”

Tip: You can also create Data Table output values, which retrieve values during the run session and insert them into a column in the Data Table. You can then use these columns as Data Table parameters in your test. For more information, see Chapter 25, “Outputting Values.”

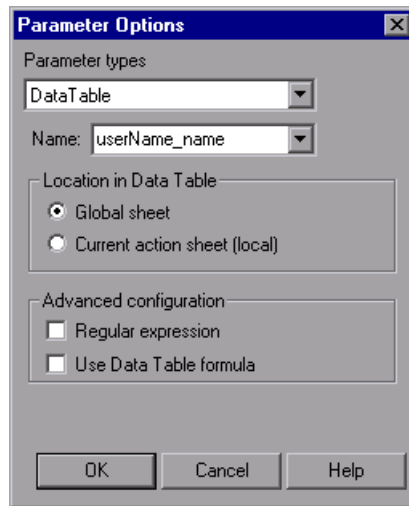
In the previous example, QuickTest submits a separate query for each itinerary when you run the test.



Setting Data Table Parameter Options

When you choose to parameterize a value, the dialog box that opens enables you to select a parameter type and the parameter options to use. The image below shows the dialog box that opens when you select to parameterize a checkpoint expected value. The dialog boxes for parameterizing other value types such as argument values, object property values, and output storage locations provide similar options.

When **Data Table** is selected as the parameter type, you can configure your parameter to use values from the Data Table.



Tip: When you open the dialog box to parameterize a value, **Data Table** may be set as the default parameter type. For more information on default parameter type settings, see “Understanding Default Parameter Values” on page 634.

Defining the Settings for a Data Table Parameter

The following options are available for configuring Data Table parameters:

Name. Specifies the name of the parameter in the Data Table. You can create a new parameter by using the default parameter name or entering a new, descriptive name. Alternatively, you can select an existing Data Table parameter from the list.

Note: The parameter name must be unique in the sheet. It can contain letters, numbers, periods, and underscores. The first character of the parameter name must be a letter or an underscore. If you specify an invalid name, QuickTest displays a warning message when you click **OK**. You can choose to edit the name manually or to instruct QuickTest to fix the name automatically (by adding an underscore at the beginning of the name).

Location in Data Table. Specifies whether to store the parameter in the global or current action sheet in the Data Table.

For more information on global and action Data Table parameters, see “Choosing Global or Action Data Table Parameters” on page 643. For more information on actions, see Chapter 15, “Working with Actions” and Chapter 16, “Working with Advanced Action Features.”

Advanced configuration (if applicable):

- **Regular expression.** Sets the value of the parameter as a regular expression. For more information, see “Understanding and Using Regular Expressions” on page 762. Note that this option is available only when parameterizing checkpoint and object property values.
- **Use Data Table formula.** (If applicable.) Inserts two columns in the Data Table. The first column contains a formula that checks the validity of output in the second column. QuickTest uses the data in the output column to compute the formula, and inserts a value of TRUE or FALSE in the table cell of the formula column. Note that this option is available only for checkpoints. For more information on using Data Table formulas, see “Using Formulas in the Data Table” on page 1216.

Note: You can also define Data Table variables using parameterization objects and methods in the Expert View. For more information, see the *HP QuickTest Professional Object Model Reference*.

Choosing Global or Action Data Table Parameters

When you parameterize a step in a test using the Data Table, you must decide whether you want to make it a **global Data Table parameter** (per test) or a **local Data Table parameter** (per action).

This decision should be based on whether you want the data to be used only for a single action (use local Data Table parameters), or available to other actions (use global Data Table parameters) and when you want subsequent iterations (different data) to be used for a particular parameter (each time the test repeats or each time the action repeats within the test).

- Global Data Table parameters take data from the Global sheet in the Data Table. The Global sheet contains the data that replaces global parameters in each iteration of the test. By default, the test runs one iteration for each row in the Global sheet of the Data Table. Using the Run pane of the Test Settings dialog box, you can also set the test to run only one iteration, or to run iterations on specified rows within the Global sheet of the Data Table. You can use the parameters defined in the Global data sheet in any action.

Tip: By outputting values to the global Data Table sheet from one action and using them as input parameters in another action, you can pass values from one action to another. For more information, see Chapter 25, “Outputting Values.”

For more information on setting global iteration preferences, see “Defining Run Settings for Your Test” on page 1270.

- Local Data Table parameters take data from the action's sheet in the Data Table. The data in the action's sheet replaces the action's Data Table parameters in each iteration of the action. By default, actions run only one iteration.

Using the Run tab of the Action Call Properties dialog box, you can also set a particular call of the action to run iterations for all rows in the action's sheet or to run iterations on specified rows within the action's sheet. When you set your action properties to run iterations on all rows, QuickTest inserts the next value from the action's data sheet into the corresponding action parameter during each **action iteration**, while the values of the global parameters stay constant.

For more information on setting action iteration preferences, see "Inserting a Call to an Existing Action" on page 468.

Note: After running a parameterized test, you can view the actual values taken from the Data Table in the Test Results Run-Time Data Table. For more information, see "Viewing the Run-Time Data Table" on page 1056.

If you have multiple rows in the Global data sheet, the entire test runs multiple times. If you have multiple rows in a local data sheet, the corresponding action runs multiple times before running the next action in the test. If you have multiple rows in both Global and local data sheets, each single test iteration runs all iterations of each action before running the next iteration of the test.

Using Environment Variable Parameters

QuickTest can insert a value from the Environment variable list, which is a list of variables and corresponding values that can be accessed from your test. Throughout the test run, the value of an environment variable remains the same, regardless of the number of iterations, unless you change the value of the variable programmatically in your script.

Tip: Environment parameters are especially useful for localization testing, when you want to test an application where the user interface strings change, depending on the selected language. Environment parameters can be used for testing the same application on different browsers. You can also vary the input values for each language by selecting a different Data Table file each time you run the test. For more information, see Chapter 42, “Working with Data Tables.”

There are several types of environment variables:

- **User-Defined Internal.** Variables that you define within the test. These variables are saved with the test and are accessible only within the test in which they were defined.

You can create or modify internal, user-defined environment variables for your test in the Environment pane of the Test Settings dialog box or in the Parameter Options dialog box.

For more information on creating or modifying environment variables in the Test Settings dialog box, see “Defining Environment Settings for Your Test” on page 1283.

For information on creating or modifying environment variables in the Parameter Options dialog box, see “Setting Environment Variable Parameter Options” on page 652.

Tip: You can also create environment output values, which retrieve values during the test run and output them to internal environment variable parameters for use in your test. For more information, see Chapter 25, “Outputting Values.”

- **User-Defined External.** Variables that you predefine in the active external environment variables file. You can create as many files as you want and select an appropriate file for each test, or change files for each test run. Note that external environment variable values are designated as read-only within the test. For more information, see “Using User-Defined External Environment Variables” on page 647.
- **Built-in.** Variables that represent information about the test and the computer on which the test is run, such as **Test path** and **Operating system**. These variables are accessible from all tests, and are designated as read-only. For more information, see “Using Built-in Environment Variables” on page 650.

Note: QuickTest also has a set of predefined environment variables that you can use to set the values of the Record and Run Settings dialog options. You should not use the names of these variables for any other purpose. For more information, see the section on using environment variables to specify the Record and Run details for your test in the *HP QuickTest Professional User Guide*.

Using User-Defined External Environment Variables

You can create a list of variable-value pairs in an external file in **.xml** format. You can then select the file as the active external environment variable file for a test and use the variables from the file as parameters.

You can set up your environment variable files manually, or you can define the variables in the Environment pane of the Test Settings dialog box and use the **Export** button to create the file with the correct structure. For more information on exporting environment variables, see Chapter 45, “Setting Options for Individual Tests.”

Notes:

- You can also store environment variable files in Quality Center. For more information, see “Using Environment Variable Files with Quality Center” on page 649.
 - You can create several external variable files with the same variable names and different values and then run the test several times, using a different file each time. This is especially useful for localization testing.
-

If you create your files manually, you must use the correct format, as defined below. You can use the QuickTest environment variable file schema in:

<QuickTest Professional installation folder>\help\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 Select **File > Settings** to open the Test Settings dialog box. For more information on the Test Settings dialog box, see Chapter 45, “Setting Options for Individual Tests.”
- 2 Click the **Environment** node.
- 3 Select **User-defined** from the **Variables type** list.
- 4 Select the **Load variables and values from external file (reloaded each run session)** check box.
- 5 Use the browse button or enter the full path of the external environment variables file you want to use with your test. The variables defined in the selected file are displayed in blue in the list of user-defined environment variables.

You can now select the variables in the active file as external user-defined environment parameters in your test. For more information, see “Setting Environment Variable Parameter Options” on page 652.

Using Environment Variable Files with Quality Center

When working with Quality Center and environment variable files, you must save the environment variable file in the Test Resources module in your Quality Center project before you specify the file in the Environment pane of the Test Settings dialog box.

You can add a new or an existing environment variable file to your Quality Center project. Note that adding an existing file from the file system to a Quality Center project creates a copy of the file in Quality Center. Thus, once you save the file to the project, changes made to the Quality Center environment variable file will not affect the file system file and vice versa.

To use an environment variable file with Quality Center:

- 1** To add a new environment variable file, create a new **.xml** file in your file system, as described in “Using User-Defined External Environment Variables” on page 647.
- 2** In Quality Center, create a new environment variable resource and then upload the **.xml** file you created in the previous step to the project’s Test Resources module. For more information, see the *HP Quality Center User Guide*.
- 3** In QuickTest, connect to the Quality Center project. For more information, see “Connecting to and Disconnecting from Quality Center” on page 1418.
- 4** In the Test Settings dialog box, click the **Environment** node.
- 5** Select **User-defined** from the **Variables type** list.
- 6** Select **Load variables and values from external file (reload each run session)**.
- 7** In the **File** box, click the browse button to find the user-defined variable file in the Quality Center project.
- 8** Save your test. QuickTest saves the file to the Quality Center project.

For more information on working with Quality Center, see Chapter 51, “Integrating with Quality Center” and the *HP Quality Center User Guide*.

Using Built-in Environment Variables

QuickTest provides a set of built-in variables that enable you to use current information about the test and the QuickTest computer running your test. These can include the test name, the test path, the operating system type and version, and the local host name.

For example, you may want to perform different checks in your test based on the operating system being used by the computer that is running the test. To do this, you could include the OSVersion built-in environment variable in an If statement.

You can also select built-in environment variables when parameterizing values. For more information, see “Setting Environment Variable Parameter Options” on page 652.

The following built-in environment variables are available:

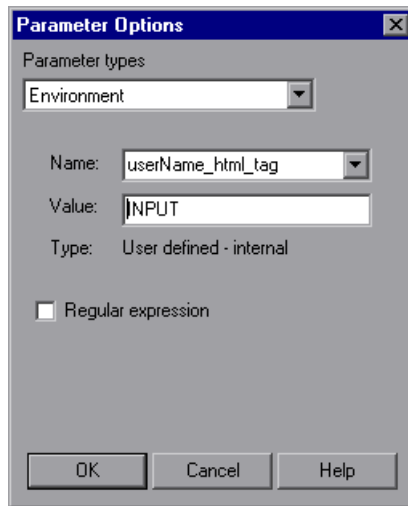
Name	Description
ActionIteration	The action iteration currently running.
ControllerHostName	The name of the controller’s computer. This variable is relevant only when running as a GUI Vuser from the LoadRunner controller.
GroupName	The name of the group in the running scenario. This variable is relevant only when running as a GUI Vuser from the LoadRunner controller.
LocalHostName	The local host name.
OS	The operating system.
OSVersion	The operating system version.
ProductDir	The folder path where the product is installed.
ProductName	The product name.
ProductVer	The product version.

Name	Description
ResultDir	The path of the folder in which the current test results are located. Note: You cannot use the ResultDir environment variable when running a test from Business Availability Center, LoadRunner, or the Silent Test Runner in QuickTest.
ScenarioId	The identification number of the scenario. This variable is relevant only when running as a GUI Vuser from the LoadRunner controller.
SystemTempDir	The system temporary directory.
TestDir	The path of the folder in which the test is located.
TestIteration	The test iteration currently running.
TestName	The name of the test.
UpdatingActiveScreen	Indicates whether the Active Screen images and values are being updated during the update run process. For more information, see “Updating a Test Using the Update Run Mode Option” on page 1125.
UpdatingCheckpoints	Indicates whether checkpoints are being updated during the update run process. For more information, see “Updating a Test Using the Update Run Mode Option” on page 1125.
UpdatingTODescriptions	Indicates whether the set of properties used to identify test objects are being updated during the update run process. For more information, see “Updating a Test Using the Update Run Mode Option” on page 1125.
UserName	The Windows login user name.
VuserId	The Vuser identification under load. This variable is relevant only when running as a GUI VUser from the LoadRunner controller.

Setting Environment Variable Parameter Options

When you choose to parameterize a value, the dialog box that opens enables you to select a parameter type and the parameter options to use. The image below shows the dialog box that opens when you select to parameterize a checkpoint expected value. The dialog boxes for parameterizing other value types such as argument values, object property values, and output storage locations provide similar options.

When you select **Environment** as the parameter type, you can configure your parameter to use values from the Environment variable list.



Defining the Settings for an Environment Variable Parameter

The following options are available for configuring environment variable parameters:

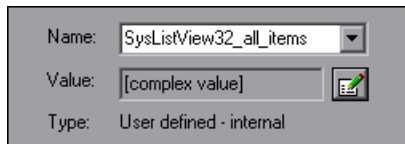
- **Name.** Specifies the name of the parameter. For an internal user-defined environment variable parameter, you can create a new parameter by using the default parameter name or entering a new, descriptive name. Alternatively, you can select an existing internal user-defined environment variable parameter from the list.

Notes:

- If you edit the name displayed in the **Name** box for an existing parameter, you create a new internal user-defined environment variable parameter. The original environment variable parameter is not modified.
- If you are parameterizing an argument that receives a predefined constant or number, only the environment variable parameters whose value is of type **integer** are shown in the **Name** list.

- **Value.** Specifies the value of the parameter. You can enter the value for a new user-defined internal parameter, or modify the value for an existing user-defined internal parameter. External and built-in environment variable parameter values cannot be modified in this dialog box.

If the entire value of a selected environment variable parameter cannot be displayed in the **Value** box, it is shown as **[complex value]**. For example, the value of a list's **all items** property is a multi-line value, where each line contains the value of an item in the list.



You can view or edit a complex value by clicking the **View/Edit Complex Value** button. For more information, see “Viewing and Editing Complex Parameter Values” on page 654.

- **Type.** Specifies the type of environment variable parameter (read-only):
 - **internal user-defined**
 - **external user-defined**
 - **built-in**

Tip: The value of an environment variable remains the same throughout the test run, regardless of the number of iterations, unless you change the value of the variable programmatically in your script.

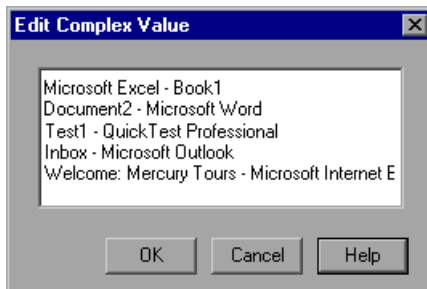
- **Regular expression.** Sets the value of the parameter as a regular expression. This option is available only when parameterizing a checkpoint or object property text string value, and the selected environment variable parameter type is **internal user-defined**. For more information on regular expressions, see “Understanding and Using Regular Expressions” on page 762.

Note: You can also define environment variables using parameterization objects and methods in the Expert View. For more information, see the *HP QuickTest Professional Object Model Reference*.

Viewing and Editing Complex Parameter Values



When you click the **View/Edit Complex Value** button for a parameter with a value that cannot be displayed entirely in the **Value** box, the Edit Complex Value dialog box displays the full contents of the value.



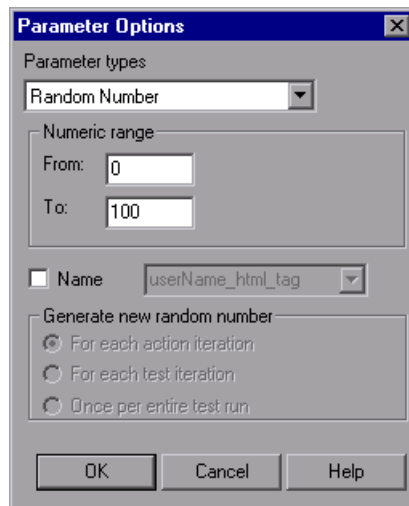
You can edit the value for an internal user-defined environment variable parameter.

For an external or built-in environment variable parameter, you can view the value but you cannot modify it in this dialog box.

Using Random Number Parameters

When you choose to parameterize a value, the dialog box that opens enables you to select a parameter type and the parameter options to use. The image below shows the dialog box that opens when you select to parameterize a checkpoint expected value. The dialog boxes for parameterizing other value types such as argument values, object property values, and output storage locations provide similar options.

When you select **Random Number** as the parameter type, the Parameter Options dialog box enables you to configure your parameter to use random numbers.



Defining Settings for a Random Number Parameter

The following options are available for configuring random number parameters:

- **Numeric range.** Specifies the range from which the random number is generated. By default, the random number range is between 0 and 100. You can modify the range by entering different values in the **From** and **To** boxes. The range must be between 0 and 2147483647 (inclusive).
- **Name.** Assigns a name to your parameter. Assigning a name to a random parameter enables you to use the same parameter several times in your test. You can select an existing named parameter or create a new named parameter by entering a new, descriptive name.
- **Generate new random number.** Defines the generation timing for a named random parameter. This box is enabled when you select the **Name** check box. You can select one of the following options:
 - **For each action iteration.** Generates a new number at the end of each action iteration.
 - **For each test iteration.** Generates a new number at the end of each global iteration.
 - **Once per entire test run.** Generates a new number the first time the parameter is used. The same number is used for the parameter throughout the test run.

Notes:

- Random number parameters are not appropriate for non-numeric values, such as text or hypertext links.
 - If you select an existing parameter, then changing the settings in the dialog box affects all instances of that parameter in the test.
 - You can also define random number variables using parameterization objects and methods in the Expert View. For more information, see the *HP QuickTest Professional Object Model Reference*.
-

Example of a Parameterized Test

The following example shows how to parameterize a step method and a checkpoint using Data Table parameters.

When you test your application, you may want to check how it performs the same operations with multiple sets of data. For example, if you are testing the Mercury Tours sample Web site, you may want to check that the correct departure and the arrival cities are selected before you book a particular flight.

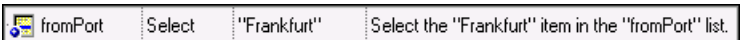
Suppose that you want to check that the flights are booked correctly for a variety of different locations. Rather than create a separate test with a separate checkpoint for each location, you can parameterize the location information. For each iteration of the test, QuickTest then checks the flight information for the different locations.


The following is a sample test of a flight booking procedure. The departure city is Frankfurt and the arrival city is Acapulco.

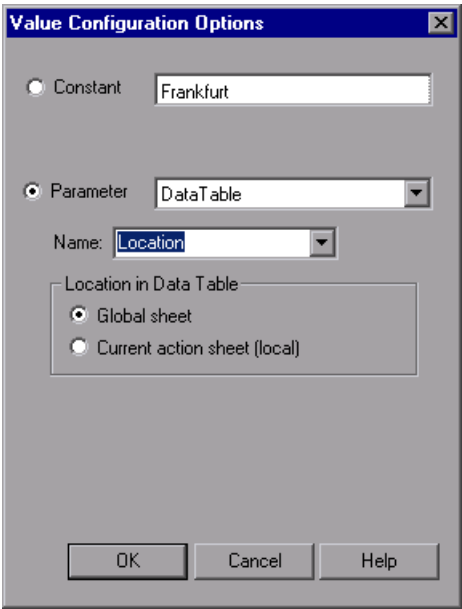
Item	Operation	Value	Documentation
▼ Action1			
Welcome: Mercury Tours			
Welcome: Mercury Tours			
userName	Set	"mercury"	Enter "mercury" in the "userName" edit box.
password	SetSecure	"435b8eb45e4fd...	Enter the encrypted string "435b8eb45e4fd8dd653c4d65fc1aa90e5c...
Sign-In	Click	26,8	Click the "Sign-In" 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.

Step 1: Parameterize a Step

Parameterize the method argument of the **fromPort** step:



In the Keyword View, click in the **Value** cell of the step and then click the parameterization icon . In the Value Configuration Options dialog box, select the **Parameter** radio button. In the **Name** box, rename p_item to **Location**.



Click **OK**. The **Location** column is added to the Data Table.

For more information on parameterizing a step, see “Parameterizing Values in Steps and Checkpoints” on page 628.

Step 2: Parameterize a Checkpoint

In the following example, you add a parameterized text checkpoint to check that the correct locations were selected before you book a flight.

Select the Select a Flight step. In the Active Screen, highlight the text Frankfurt to Acapulco, right-click and insert a text checkpoint:

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

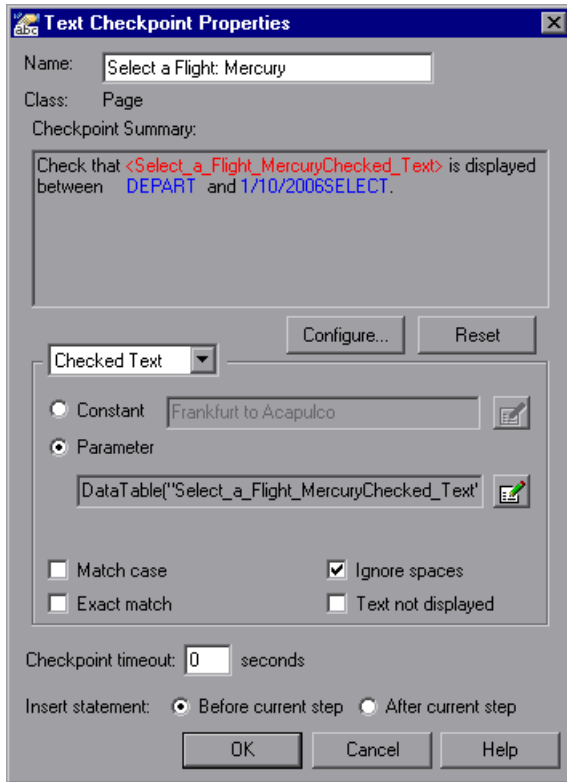
Frankfurt to Acapulco 03/09/2004

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 628.

Step 3: 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 42, “Working with Data Tables.”

Modified Test

The following example shows the test after parameterizing the step and creating a parameterized text checkpoint.

Welcome: Mercury T...			
userName	Set	"mercury"	Enter "mercury" in the "userName" edit box.
password	SetS...	"404ee5e998b25bdc6f116b031452..."	Enter the encrypted string "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> item in the "fromPort" list.
toPort	Select	"Acapulco"	Select the "Acapulco" item in the "toPort" list.
toMonth	Select	"Jun"	Select the "Jun" item in the "toMonth" list.
findFlights	Click	2,2	Click the "findFlights" image.
Select a Flight: Mercury	Check	Checkpoint("Frankfurt to Acapulco")	Check whether text in the "Select a Flight: Mercury" window matches the text "Frankfurt to Acapulco".
reserveFlights	Click	2,2	Click the "reserveFlights" image.
Book a Flight: Mercury			
passFirst0	Set	"John"	Enter "John" in the "passFirst0" edit box.
passLast0	Set	"Brown"	Enter "Brown" in the "passLast0" edit box.
creditnumber	Set	"333666777"	Enter "333666777" in the "creditnumber" edit box.

The parameterized value for the fromPort step is clearly shown as a Data Table parameter. To see the parameterization setting for the checkpoint, click in the **Value** column for the Select a Flight step.

Using the Data Driver to Parameterize Your Test

The Data Driver enables you to quickly parameterize several (or all) property values for test objects, checkpoints, and/or method arguments containing the same constant value within a given action.

You can choose to replace all occurrences of a selected constant value with a parameter, in the same way that you can use a **Find and Replace All** operation instead of a step-by-step **Find and Replace** process. QuickTest can also show you each occurrence of the constant so that you can decide whether or not to parameterize the value.

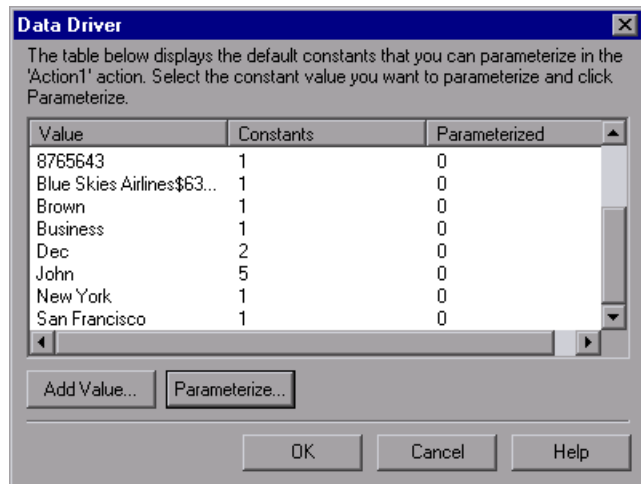
Notes:

- When finding multiple occurrences of a selected value, QuickTest conducts a search that is case sensitive and searches only for exact matches. (It does not find values that include the selected value as part of a longer string.)
 - You cannot use the Data Driver to parameterize the values of arguments for user-defined methods or VBScript functions.
-

To parameterize a value using the Data Driver:

- 1 Display the action you want to parameterize.
- 2 Select **Tools > Data Driver**.

QuickTest scans the test for constants before the Data Driver opens (this may take a few moments).



Note: If the action being scanned contains a large number of lines and constant values, QuickTest warns you that loading the constants may take some time. You can choose whether to wait for the constants to load, or to open the Data Driver wizard quickly without constants.

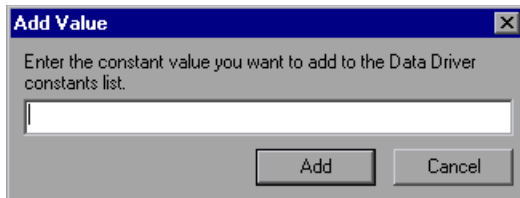
The Data Driver displays the Constants list for the action. For each constant value, it displays the number of times the constant value appears in the action.

By default, the list displays the constants for one or more of the arguments of the following methods: **Activate**, **Collapse**, **Deselect**, **Expand**, **ExtendSelect**, **Press**, **Select**, **SelectColumn**, **SelectRange**, **SelectRow**, **Set**, **SetCellData**, **SetSecure**, **SetText**, **Type**, and **WaitProperty**.

For more information on how to work with testing methods, see Chapter 29, “Working in the Expert View and Function Library Windows.” For syntax and method information, see the *HP QuickTest Professional Object Model Reference*.

Note: If you chose not to wait for the constants to load, the Data Driver opens with an empty Constants table. You can add the constant values that you want to parameterize to the Data Driver, as described below.

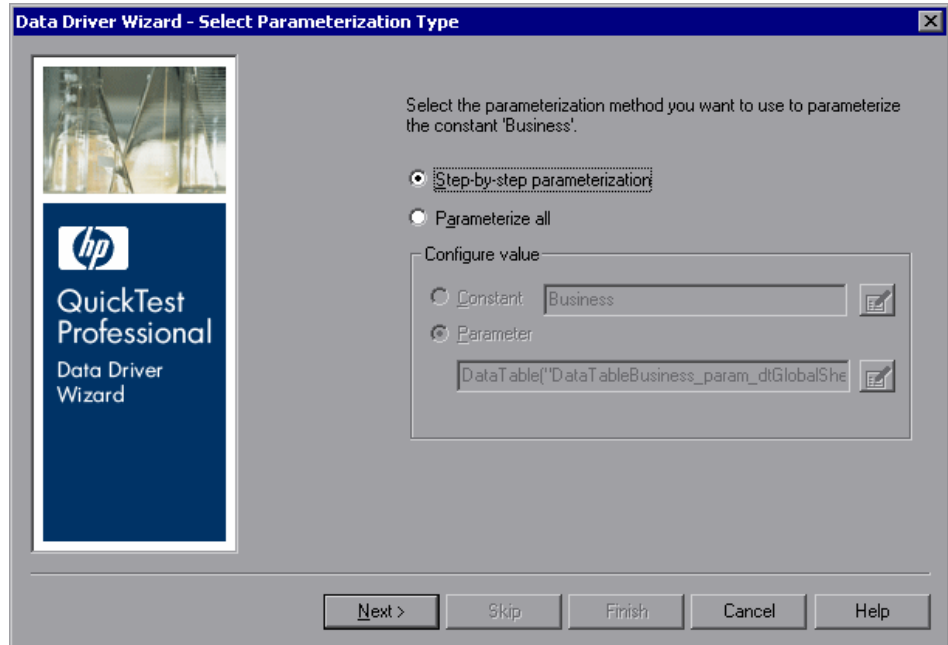
- 3 If you want to parameterize a value that is not currently displayed in the list (such as an object property value), click **Add Value**. The Add Value dialog box opens.



Enter a constant value in the dialog box and click **Add**. The constant is added to the list.

Note: You can add only constant values that currently exist in the test action.

- 4 Select the value you want to parameterize from the Constants list and click **Parameterize**. The Data Driver Wizard opens.

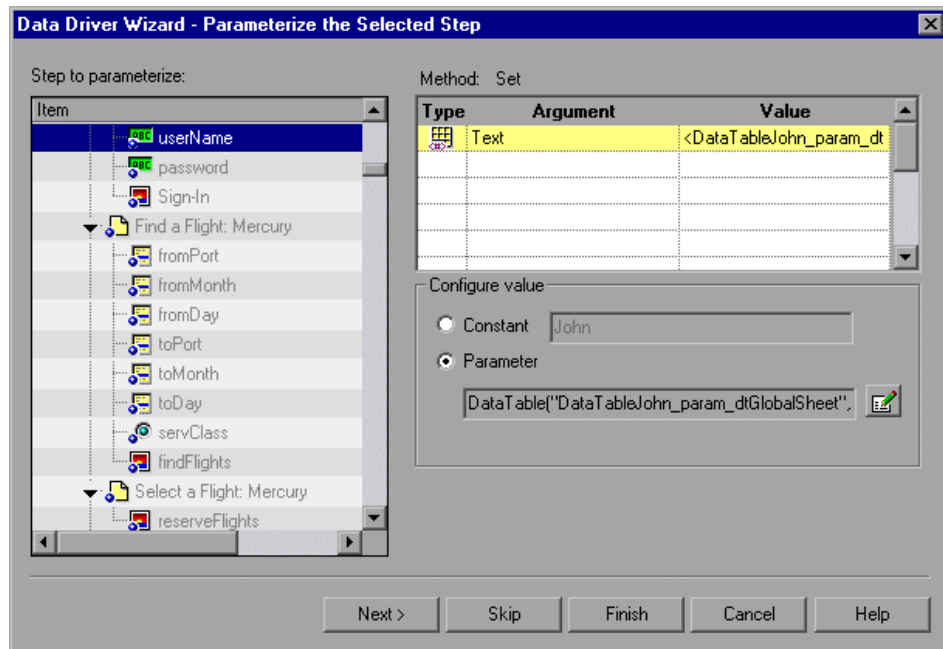


- 5 Select the type of parameterization you want to perform:
 - **Step-by-step parameterization.** Enables you to view the current values of each step containing the selected value. For each step, you can choose whether or not to parameterize the value and if so, which parameterization options you want to use.
 - **Parameterize all.** Enables you to parameterize all occurrences of the selected value throughout the action. You set your parameterization preferences one time and the same options are applied to all occurrences of the value.

- 6 If you selected **Step-by-step parameterization**, click **Next**. The Parameterize the Selected Step screen opens.

If you selected **Parameterize all**, the **Parameter** option is enabled in the **Configure value** area. Select your parameterization preferences the same way that you would for an individual step. For more information, see “Parameterizing Values in Steps and Checkpoints” on page 628. Proceed to step 9.

- 7 In the **Step to parameterize** area, the first step with an object property or checkpoint value containing the selected value is displayed in the test tree on the left. The parameterization options for the step are displayed on the right.



The default parameterization settings are displayed for the value. For more information on default parameterization settings, see “Understanding Default Parameter Values” on page 634.



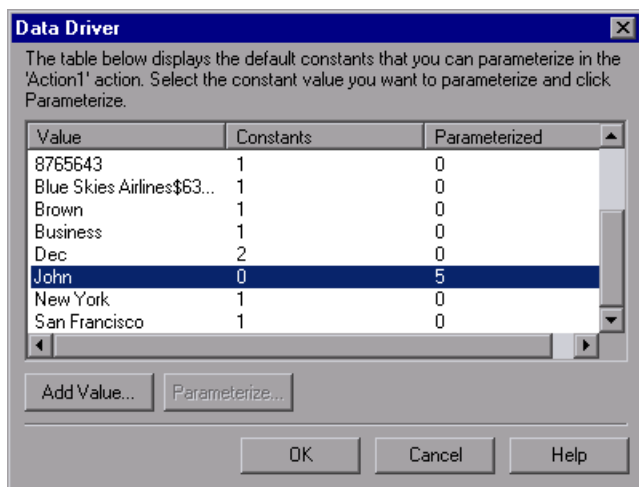
Accept the default parameterization settings or click the **Parameter Options** button to set the parameterization options you want to apply to this step. For more information, see “Parameterizing Values in Steps and Checkpoints” on page 628.

- Click **Next** to parameterize the selected step and view the next step containing the selected value.
- Click **Skip** if you do not want to parameterize the selected step.
- Click **Finish** to apply the parameterization settings of the current step to all remaining steps containing the selected value.

- 8 If you clicked **Next** in the previous step, and steps remain that contain the selected value, the Parameterize the Selected Step screen opens displaying the next relevant step. Repeat step 7 for each relevant step.

If there are no remaining steps containing the selected value, the Finished screen opens.

- 9 Click **Finish**. The Data Driver Wizard closes and the Data Driver main screen shows how many occurrences you selected to parameterize and how many remain as constants.



- 10** If you want to parameterize another constant value, select the value and repeat steps 4 to 9.
- 11** When you are finished parameterizing constants, click **OK**. The parameterization options you selected are applied to your action.

25

Outputting Values

QuickTest enables you to retrieve values in your test and store them in output value objects. You can subsequently retrieve these values and use them as input at a different stage in the run session.

This chapter includes:

- About Outputting Values on page 669
- Creating Output Values on page 670
- Outputting Property Values on page 676
- Specifying the Output Type and Settings on page 683
- Outputting Text Values on page 688
- Outputting Table Values on page 698
- Outputting Database Values on page 713
- Outputting XML Values on page 718
- Updating the XML Hierarchy for XML Test Object Operation Output Value Steps (For WebService Test Objects Only) on page 732
- Adding Existing Output Values to a Test on page 736

About Outputting Values

An **output value** step is a step in which one or more values are captured at a specific point in your test and stored for the duration of the run session. The values can later be used as input at a different point in the run session.

You can output the property values of any object. You can also output values from text strings, table cells, databases, and XML documents.

When you create output value steps, you can determine where the values are stored during the run session and how they can be used. During the run session, QuickTest retrieves each value at the specified point and stores it in the specified location. When the value is needed later in the run session, QuickTest retrieves it from this location and uses it as required.

Output values are stored only for the duration of the run session. When the run session is repeated, the output values are reset.

Note: After the run session, you can view the output values retrieved during the session as part of the session results. For more information, see “Viewing Parameterized Values and Output Value Results” on page 1053.

Creating Output Values

When you add an output value step to your test, you first select the category of values to output, for example, property values, text values, or XML element values. For more information, see *Output Value Categories*.

You can then determine which values to output. For more information, see “Viewing and Editing Output Values” on page 675.

You also determine the storage location for each value. For more information, see “Storing Output Values” on page 673.

Output Value Categories

You can create the following categories of output values:

- Standard output values
- Text and text area output values
- Table output values
- Database output values
- XML output values

Standard Output Values

You can use standard output values to output the property values of most objects. For example, in a Web-based application, the number of links on a Web page may vary based on the selections a user makes on a form on the previous page. You could create an output value in your test to store the number of links on the page.

Note: You can also use standard output values to output the contents of table cells. For more information, see “Table Output Values” on page 672.

Tip: You can use standard output values to output text strings by specifying the **text** property of the object as an output value.

For more information on standard output values, see “Outputting Property Values” on page 676.

Text and Text Area Output Values

You can use text output values to output text strings displayed in an application. When creating a text output value, you can output a part of the object’s text. You can also specify the text before and after the output text.

You can use text area output values to output text strings displayed within a defined area of a screen in a Windows-based application.

For example, suppose that you want to store the text of any error message that appears after a specific step in the Web application you are testing. Inside the If statement, you check whether a window exists with a known title bar value, for example Error. If it exists, you output the text in this window (assuming that the window size is the same for all possible error messages).

For more information on text output values, see “Outputting Text Values” on page 688. For more information on text area output values, see “Creating Text Area Output Values” on page 690.

Table Output Values

Table output values are a subset of standard output values, described above. You can use table output values to output the contents of table cells. For some types of tables, you can specify a row range from which to choose the table cells. During the run session, QuickTest retrieves the current data from the specified table cells according to the settings that you specified and outputs the values to the Data Table.

For more information, see “Outputting Table Values” on page 698.

Database Output Values

You can use database output values to output the value of the contents of database cells, based on the results of a query (result set) that you define on a database. You can create output values from the entire contents of the result set, or from a part of it. During the run session, QuickTest retrieves the current data from the database and outputs the values according to the settings that you specified.

For more information, see “Outputting Database Values” on page 713.

XML Output Values

You can use XML output values to output the values of XML elements and attributes in XML documents.

After the run session has finished, you can view summary results of the XML output values in the Test Results window. You can also view detailed results by opening the XML Output Value Results window. For more information, see Chapter 33, “Viewing Run Session Results.”

For example, suppose that an XML document in a Web page contains a price list for new cars. You can output the price of a particular car by selecting the appropriate XML element value to output.

For more information on XML output values, see “Outputting XML Values” on page 718.

Output Value Categories and Environments

QuickTest add-ins help you to create and run tests and components on applications in a variety of development environments. For information about using output values for each add-in environment installed with QuickTest Professional, see “Supported Output Values” on page 1548.

Storing Output Values

When you define an output value, you can specify where and how each value is stored during the run session.

You can output a value to:

- a test or action parameter
- the run-time Data Table
- an environment variable

Note: Output values are stored only for the duration of the test, and are not saved with the test. If you select to output a value to an existing parameter, Data Table column, or environment variable, the existing value is overwritten when the output value step runs. When the run session ends, the original value is restored.

Storing Values in Test and Action Parameters

You can output a value to an action parameter, so that values from one part of a run session can be used later in the run session, or be passed back to the application that ran (called) the test.

For example, suppose you are testing a shopping application that calculates your purchases and automatically debits your account with the amount that you purchased. You want to test that the application correctly debits the purchase amount from the account each time that the action is run with a different list of items to purchase. You could output the total amount spent to an action parameter value, and then use that value later in your run session in the action that debits the account.

For more information on action parameters in general, see “Using Action Parameters” on page 476.

Storing Values in the Run-time Data Table

The option to output a value to the run-time Data Table is especially useful with a **data-driven** test (or action) that runs several times. In each repetition, or **iteration**, QuickTest retrieves the current value and stores it in the appropriate row in the run-time Data Table.

For example, suppose you are testing a flight reservation application and you design a test to create a new reservation and then view the reservation details. Every time you run the test, the application generates a unique order number for the new reservation. To view the reservation, the application requires the user to input the same order number. You do not know the order number before you run the test.

To solve this problem, you output a value to the Data Table for the unique order number generated when creating a new reservation. Then, in the View Reservation screen, you use the column containing the stored value to insert the output value into the order number input field.

When you run the test, QuickTest retrieves the unique order number generated by the site for the new reservation and enters this output value in the run-time Data Table. When the test reaches the order number input field required to view the reservation, QuickTest inserts the unique order number stored in the run-time Data Table into the order number field.

Storing Values in Environment Variables

When you output a value to an internal user-defined environment variable, you can use the environment variable input parameter at a later stage in the run session.

Note: You can output values only to internal user-defined environment variables and not to external or built-in environment variables, which are read-only.

For example, suppose you are testing an application that prompts the user to input an account number on a Welcome page and then displays the user's name. You can use a text output value to capture the value of the displayed name and store it in an environment variable.

You can then retrieve the value in the environment variable to enter the user's name in other places in the application. For example, in an Order Checkbook Web page, which for security reasons requires users to enter the name to appear on the checks, you could use the value to insert the user's name into the **Name** edit box.

Viewing and Editing Output Values

When you insert an output value step in your test, the Keyword View shows the step with **Output** displayed in the **Operation** column and **CheckPoint** displayed in the **Value** column, followed by the name assigned to the output value.

The output value statement is displayed in the Expert View with the following syntax:

Object.Output CheckPoint(*Name*)



You can view or edit the output value or its details in the relevant Output Value Properties dialog box, by right-clicking the step and choosing **Output Value Properties**. Alternatively, you can click the step in the **Value** column in the Keyword View and then click the **Output Properties** button.

For more information on the options available in the different Output Value Properties dialog boxes, see:

- “Defining Standard Output Values” on page 679
- “Defining Text and Text Area Output Values” on page 692
- “Outputting Table Content” on page 703
- “Outputting Table Properties” on page 709
- “Defining Database Output Values” on page 715
- “Understanding the XML Output Properties Dialog Box” on page 727

Outputting Property Values

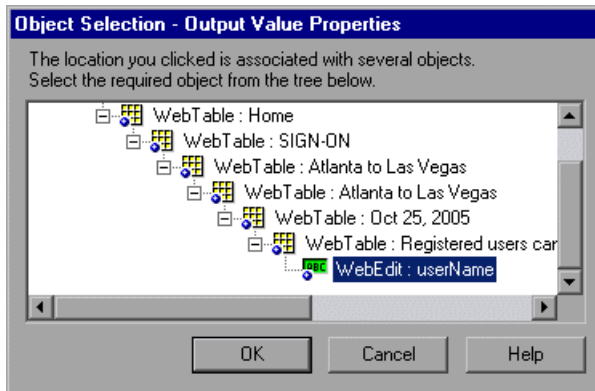
You can use standard output values to output the property values of most objects. You can also use standard output values to output the contents of table cells.

You can create standard output values while recording or editing your test.

To create standard output values while recording:



- 1** Select **Insert > Output Value > Standard Output Value**. Alternatively, you can click the arrow beside the **Insert Checkpoint or Output Value** button in the toolbar and select **Standard Output Value**. The pointer changes into a pointing hand. For more information on using the pointing hand feature, see “Tips for Using the Pointing Hand” on page 677.
- 2** In your application, click the object for which you want to specify an output value. If the location you clicked is associated with more than one object, the Object Selection – Output Value Properties dialog box opens.



- 3** In the Object Selection dialog box, select the object for which you want to specify an output value, and click **OK**. The Output Value Properties dialog box opens for the selected object. If you select a **Table** item, the Table Output Value Properties dialog box opens.

- 4 Specify the property values to output and their settings. For more information, see “Defining Standard Output Values” on page 679. If you selected a **Table** item, see “Outputting Table Content” on page 703 and “Outputting Table Properties” on page 709.
- 5 When you finish defining the output value details, click **OK**. QuickTest inserts an output value step in your test.

Tips for Using the Pointing Hand

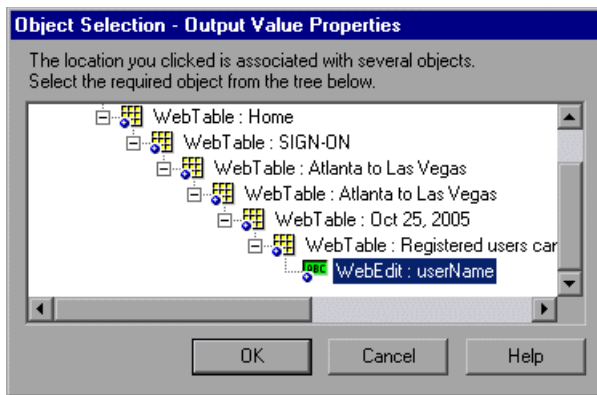
- You can hold the left CTRL key to change the pointing hand to a standard pointer. You can then change the window focus or perform operations in QuickTest or in your application, such as right-clicking, using the scroll bars, or moving the pointer over an object to display a context menu.
- If the window containing the object you want to select is partially hidden by another window, hold the pointing hand over the partially hidden window for a few seconds until it comes to the foreground. Then point to and click the required object. You can configure the length of time required to bring a window into the foreground using the General pane of the Options dialog box.
- If the window containing the object you want to select is fully hidden by another window, or if a dialog box is hidden behind a window, press the left CTRL key and arrange the windows as needed.
- If the window containing the object you want to select is minimized, you can display it by holding the left CTRL key, right-clicking the application in the Windows task bar, and choosing **Restore** from the context menu.
- If the object you want to select can be displayed only by performing an event (such as right-clicking or moving the pointer over an object to display a context menu), hold the left CTRL key. The pointing hand temporarily turns into a standard pointer and you can perform the event. When the object you want to select is displayed, release the left CTRL key. The pointer becomes a pointing hand again.

To create standard output values while editing your test:

- 1** Make sure the **Active Screen** button is selected.
- 2** Click a step whose Active Screen contains the object for which you want to specify an output value. The Active Screen displays the captured bitmap or HTML source corresponding to the highlighted step.

For Windows-based applications, make sure that the Active Screen contains property data for the object for which you want to specify an output value. For more information, see “Setting Active Screen Options” on page 1240.

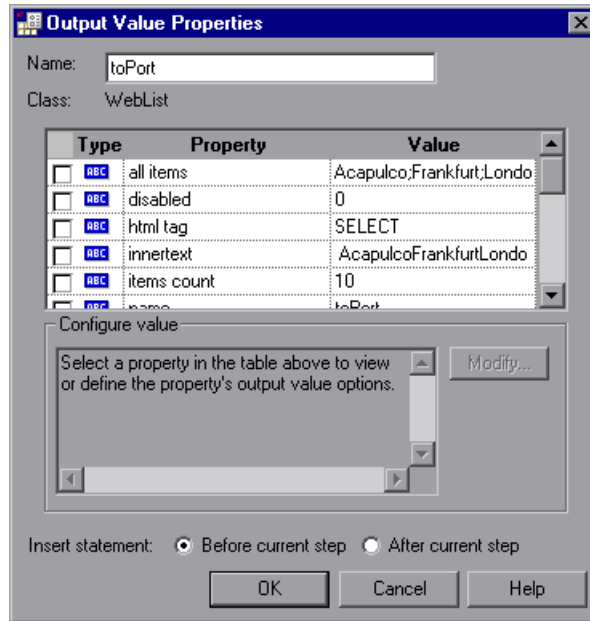
- 3** In the Active Screen, right-click the object for which you want to specify an output value and select **Insert Output Value**. Alternatively, you can right-click the step in your test and select **Insert Output Value**.
- 4** If the location you clicked is associated with more than one object, the Object Selection – Output Value Properties dialog box opens.



- 5** Select the object for which you want to specify an output value, and click **OK**. The Output Value Properties dialog box opens for the selected object. If you select a **Table** item, the Table Output Value Properties dialog box opens.
- 6** Specify the property values to output and their settings. For more information, see “Defining Standard Output Values” on page 679. If you selected a **Table** item, see “Outputting Table Content” on page 703 and “Outputting Table Properties” on page 709.
- 7** When you finish defining the output value details, click **OK**. QuickTest inserts an output value step in your test.

Defining Standard Output Values

The Output Value Properties dialog box enables you to choose which property values to output and to define the settings for each value that you select.




Note: If you insert an output value on a Web page, the Page Output Value Properties dialog box opens. This dialog box is identical to the Output Value Properties dialog box, except that it contains two additional option areas, **HTML verification** and **All objects in page**. These options are relevant only for checkpoints and are disabled when defining output values.

You can select a number of properties to output for the same object and define the output settings for each property value before closing the dialog box. When the output value step is reached during the run session, QuickTest retrieves all of the specified property values.





Identifying the Output Value

The top part of the dialog box displays information on the output value:

Item	Description
Name	<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:</p> <ul style="list-style-type: none">➤ is unique➤ does not begin or end with a space➤ does not contain " (double quotation mark)➤ does not contain the following character combinations: := @@
Class	<p>The type of test object. In this example, the WebList class indicates it is a list object in a Web application.</p>
Find in Repository button  (Located to the right of the Name box)	<p>Displays the output value in its repository.</p> <p>Note: This option is not available when creating a new output value. It is available only when editing an existing output value.</p>

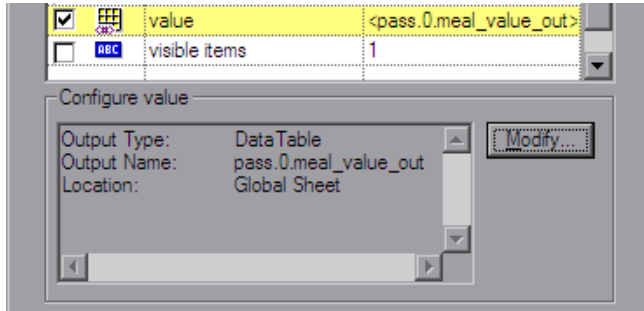
Selecting the Property Values to Output

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” on page 682.

Specifying the Output Settings for a Property Value

When you select a check box for a property, the property details are highlighted and the current output definition for the selected property value is displayed in the **Configure value** area.



When a property value is first selected for output, the default output definition for the value is displayed in the **Configure value** area. For more information on default output definitions, see “Understanding Default Output Definitions” on page 683.

When you select a property value to output, you can:

- Change the output type and/or settings for the selected value by clicking the **Modify** button. The Output Options dialog box opens and displays the current output type and settings for the value. For more information, see “Specifying the Output Type and Settings” on page 683.
- Accept the displayed output definition by selecting another property value or by clicking **OK**.

Specifying the Location for the Output Value Step

If the **Insert statement** area is displayed at the bottom of the dialog box, you can specify where the new output value step should be inserted in your test. For more information, see “Selecting the Location for the Output Value Step” on page 687.

Specifying the Output Type and Settings

The output type and settings that you define for each value determine where it is stored and how it can be used during the run session. When the output value step is reached, QuickTest retrieves each value selected for output and stores it in the specified location for use later in the run session.

When you create a new output value step, QuickTest assigns a default definition to each value selected for output. For more information, see “Understanding Default Output Definitions” on page 683.

You can change the current output definition for the selected value by selecting a different output type and/or changing the output settings in the Output Value Properties dialog box.

Understanding Default Output Definitions

When you initially select a value for output, QuickTest generates a default output definition for the value.

When you output a value for a step in a test action:

- If at least one output parameter is defined in the action, the default output type is **Test/action parameter** and the default output name is the first output parameter displayed in the Action Properties dialog box.
- If no output parameters are defined in the action, the default output type is Data Table and QuickTest creates a new Data Table output name based on the selected value.

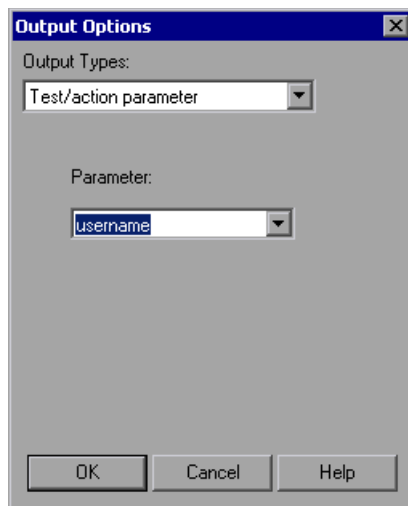
The output value is created in the Global sheet of the Data Table. For more information on creating output parameters for actions, see “Removing Actions from a Test” on page 460.

For more information on Data Table sheets, see Chapter 42, “Working with Data Tables.”

Outputting a Value to an Action Parameter

You can output a value to an action parameter, so that the values can be used later in the run session, or the values can be passed back to the external application that ran (called) the test. You can only output a value to an action parameter if the parameter has been defined as an output parameter for the calling action. You open the Output Options dialog box by clicking the **Modify** button in any Output Value Properties dialog box.

When **Test/action parameter** is selected as the output type, the Output Options dialog box enables you to select the parameter in which to store the selected value for the duration of the run session.

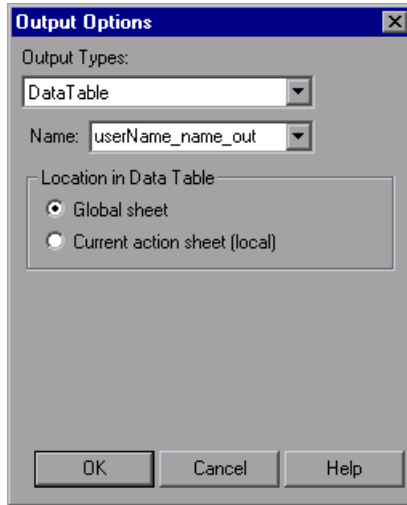


Tip: When you open the Output Options dialog box, QuickTest may display **Test/action parameter** as the default output type. This occurs if at least one output parameter is defined in the action.

The **Parameter** box specifies the name of the parameter in which to store the output value. The read-only list of available parameters contains the names and full descriptions of the currently defined output parameters for the action. You can resize the display, as needed, and, if the list of parameters is long, you can scroll through the list.

Outputting a Value to the Data Table

When **Data Table** is selected as the output type, the Output Options dialog box enables you to specify where to store the selected value within the run-time Data Table. You open the Output Options dialog box by clicking the **Modify** button in any Output Value Properties dialog box.



Tip: When you open the Output Options dialog box, QuickTest may display **Data Table** as the default output type. For more information, see “Understanding Default Output Definitions” on page 683.

The following options are available when outputting a value to the Data Table:

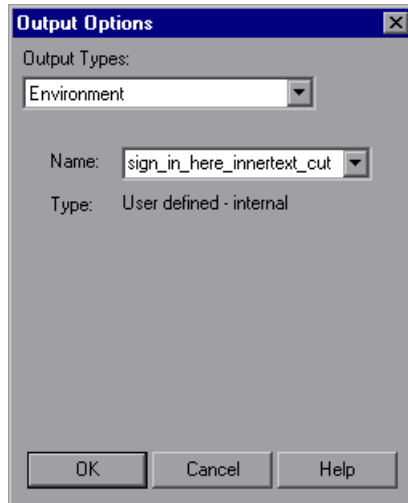
- **Name.** Specifies the name of the column in the Data Table in which to store the value. QuickTest suggests a default name for the output. You can select an existing output name from the list, or create a new output name by using the default output name or entering a valid descriptive name.

You can define a new name containing letters, numbers, periods, and underscores. The first character of the output name must be a letter or an underscore. The output name must be unique in the Data Table sheet.

- **Location in Data Table.** When outputting values for a test, specifies whether to add the Data Table column name in the global or current action sheet in the Data Table. For more information on the use of data in the global and current action sheets, see “Using Global and Action Data Sheets” on page 429. For more information on actions, see Chapter 15, “Working with Actions” and Chapter 16, “Working with Advanced Action Features.”

Outputting a Value to an Environment Variable

When you select **Environment** as the output type, the Output Options dialog box enables you to specify the internal user-defined environment variable in which to store the selected value for the duration of the run session. You open the Output Options dialog box by clicking the **Modify** button in any Output Value Properties dialog box.



The following options are available when outputting a value to an Environment variable:

- **Name.** Specifies the name of the internal user-defined environment variable in which to store the value. The list contains all currently defined internal user-defined environment variables with the corresponding type. You can select an existing variable from the list, or you can create a new internal environment variable by modifying the displayed name or by entering a new, descriptive name.

Note: If you edit the name displayed in the **Name** box for an existing variable, you create a new internal user-defined environment variable. The original environment variable is not modified.

Alternatively, you can output the value to an existing environment variable. If you select an existing variable from the list, QuickTest prompts you to choose whether to overwrite its current value with the new value when the output value step runs.

If you choose not to overwrite the current value of the selected variable, a new environment variable is created with the original variable name and an identifying suffix.

- **Type.** Displays the environment variable type. Since it is not possible to output values to external or built-in environment variables, the type is always **User-defined - internal**.

For more information on environment variables, see “Using Environment Variable Parameters” on page 645.

Selecting the Location for the Output Value Step

When you create output values while editing a test, the **Insert statement** area is displayed at the bottom of the dialog box.

By default, QuickTest inserts the new output value step before the current step (the step you selected when you chose the **Output Value** option). You can instruct QuickTest to insert the new output value step after the current step, by selecting the **After current step** option.

Note: This option is not available while recording. QuickTest automatically inserts the new output value step after the previously recorded step. It is also not available when modifying an existing output value step.

Outputting Text Values

You can create a text output value from a text string displayed in an application. You can define the output value as part of the displayed text, and specify the text before and/or after the output text.

You can also create a text output value from defined text areas. For more information, see “Creating Text Area Output Values” on page 690.

Note: Before you create a text / text area output value, make sure you configure the required capture settings in the General > Text Recognition pane (**Tools > Options > Text Recognition** node). For more information, see “The Options Dialog Box: General > Text Recognition Pane” on page 742 and “About Working with Text Recognition for Windows-Based Objects” on page 742.

Creating Text Output Values

You can create a text output value while recording or editing your test.

Note: Before you create a text output value, make sure you configure the required capture settings in the General > Text Recognition pane (**Tools > Options > Text Recognition** node). For more information, see “The Options Dialog Box: General > Text Recognition Pane” on page 742 and “About Working with Text Recognition for Windows-Based Objects” on page 742.

To create a text output value while recording:

- 1 Highlight or display the text string you want to use for an output value.
- 2 Select **Insert > Output Value > Text Output Value**. The pointer changes into a pointing hand. For more information on using the pointing hand feature, see “Tips for Using the Pointing Hand” on page 689.
- 3 In your application, click the text string for which you want to specify a text output value. The Text Output Value Properties dialog box opens.

- 4 Specify the settings for the output value. For more information, see “Defining Text and Text Area Output Values” on page 692.
- 5 When you finish defining the text output value details, click **OK**. QuickTest inserts an output value step in your test.

To create a text output value when editing your test:

- 1 Make sure the **Active Screen** is displayed.
- 2 Click a step in your test where you want to create an output value. The Active Screen displays the screen corresponding to the highlighted step.
- 3 In the Active Screen, highlight or display the text string you want to specify as an output value.
- 4 Right-click and select **Insert Text Output**. The Text Output Value Properties dialog box opens.
- 5 Specify the settings for the output value. For more information, see “Defining Text and Text Area Output Values” on page 692.
- 6 When you finish defining the output value details, click **OK**. QuickTest inserts an output value step in your test.

Tips for Using the Pointing Hand

- You can hold the left CTRL key to change the pointing hand to a standard pointer. You can then change the window focus or perform operations in QuickTest or in your application, such as right-clicking, using the scroll bars, or moving the pointer over an object to display a context menu.
- If the window containing the object you want to select is partially hidden by another window, hold the pointing hand over the partially hidden window for a few seconds until it comes to the foreground. Then point to and click the required object. You can configure the length of time required to bring a window into the foreground using the General pane of the Options dialog box.
- If the window containing the object you want to select is fully hidden by another window, or if a dialog box is hidden behind a window, press the left CTRL key and arrange the windows as needed.

- If the window containing the object you want to select is minimized, you can display it by holding the left CTRL key, right-clicking the application in the Windows task bar, and choosing **Restore** from the context menu.
- If the object you want to select can be displayed only by performing an event (such as right-clicking or moving the pointer over an object to display a context menu), hold the left CTRL key. The pointing hand temporarily turns into a standard pointer and you can perform the event. When the object you want to select is displayed, release the left CTRL key. The pointer becomes a pointing hand again.

Creating Text Area Output Values

You can create a text area output value from a text string displayed in a defined area of a screen in a Windows-based application. You can define the output value as part of the displayed text, and you can specify the text before and/or after the output text. You can create a text area output value only while recording on Windows-based applications.

When you use text-area selection to capture text displayed in a Windows application, it is often advisable to define a text area larger than the actual text you want QuickTest to use as an output value. When QuickTest runs your test, it outputs the selected text, within the defined area, according to the settings you configured.

Because text may change its position during test runs, you must make sure that the area defined is large enough so that the output text is always within its boundaries. For more information, see “About Working with Text Recognition for Windows-Based Objects” on page 742.

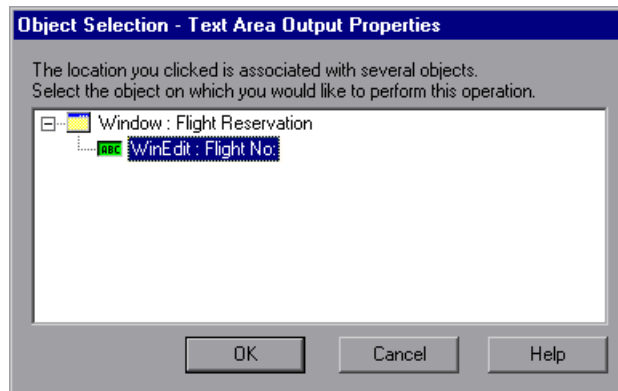
Note: Before you create a text area output value, make sure you configure the required capture settings in the General > Text Recognition pane (**Tools > Options > Text Recognition** node). For more information, see “The Options Dialog Box: General > Text Recognition Pane” on page 742.

To create a text area output value:

- 1 While recording, select **Insert > Output Value > Text Area Output Value**. The QuickTest window is hidden, and the mouse pointer turns into a crosshairs pointer.
- 2 Define the area containing the text you want QuickTest to use as an output value by clicking and dragging the crosshairs pointer. Release the mouse button after outlining the required area.

Tip: Hold down the left mouse button and use the arrow keys to make precise adjustments to the defined area.

If the area you defined is associated with more than one object, the Object Selection – Text Area Output Properties dialog box opens.

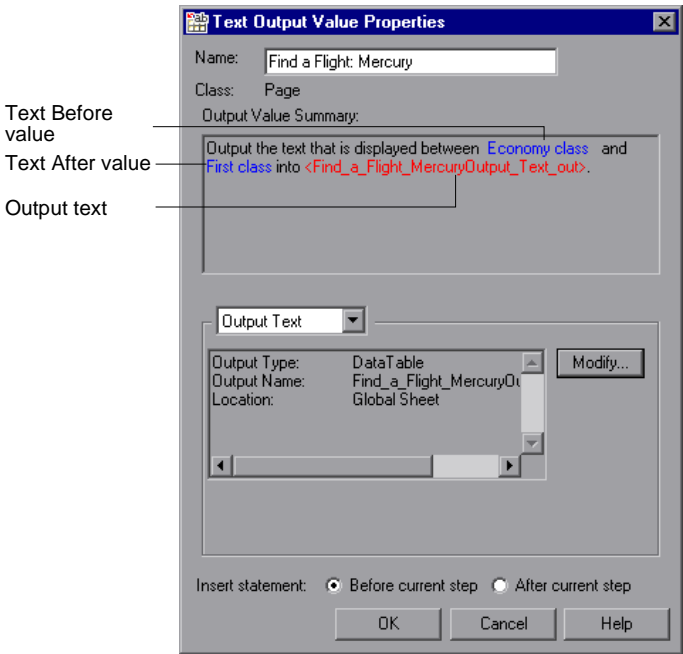


- 3 Select the object for which you are creating the output value. The Text Area Output Value Properties dialog box opens.
- 4 Specify the settings for the output value. For more information, see "Defining Text and Text Area Output Values" on page 692.
- 5 When you have finished defining the output value details, click **OK**. QuickTest inserts an output value step in your test.

Defining Text and Text Area Output Values

You can specify a text string as an output value. You can also specify the text that is displayed before and after the output value text string. This is helpful when the text string you want to specify as an output value is displayed several times in the defined screen area or when the text could change in a predictable way during a run session.

The Text Output Value Properties and Text Area Output Value Properties dialog boxes enables you to define the output value settings for the selected text string, and to define the options for the text displayed before and after the output value.



The top of the Text Output Value Properties dialog box displays the name of the output value and the class of the test object on which the output value check is being performed. You can modify the output value name, if required. For more information, see “Identifying the Output Value” on page 680.

The **Output Value Summary** pane at the top of the dialog box describes the text string for the output value. The text string is the string displayed between the **Text Before** value and the **Text After** value. This pane also shows the output name assigned to the text string. QuickTest automatically displays the text output in red, and the text before and after the text output in blue. For example, in the dialog box displayed above, the output value is the text displayed between **Economy class** (the **Text Before** value) and **First class** (the **Text After** value).


For a text area output value, the output value string contains all the text in the selected area. Although the Text Output Value Properties and Text Area Output Value Properties dialog boxes are identical, when you create a text area output value, the **Text Before** and **Text After** values are not captured.

When you create a text or text area output value, you can specify the captured text as an output value. You can also specify options for **Text Before** and **Text After** values. For example, you can define these values as parameters. If the specified text is displayed more than once in the selected object or area, you can specify the exact occurrence that relates to the output value. If you are editing your test, you can also specify the location for the output value step.

Identifying the Output Value

The top part of the Table Output Value Properties dialog box contains the following options:



Name	<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:</p> <ul style="list-style-type: none">➤ is unique➤ does not begin or end with a space➤ does not contain " (double quotation mark)➤ does not contain the following character combinations: := @@
Class	<p>Specifies the type of object (read-only). This may be a table-type object or a list view-type object.</p>
Find in Repository button 	<p>Displays the output value in its repository.</p> <p>Note: This option is not available when creating a new output value. It is available only when editing an existing output value.</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 683.

You can accept the displayed output definition, or you can click **Modify** to specify the output settings for the selected text. For more information, see “Specifying the Output Type and Settings” on page 683.

Specifying Options for the Text Before/Text After Values

When you select **Text Before** or **Text After** from the list box, you can define the options for the text displayed before or after the output value string.

Text Before

☒ Use the text before

Text to capture is displayed after occurrence 1 of

☒ Constant Economy class

☐ Parameter

DataTable("Find_a_Flight_MercuryText_Before", dtGlc)

Text After

☒ Use the text after

Text to capture is displayed before occurrence 1 of

☒ Constant First

☐ Parameter

DataTable("Find_a_Flight_MercuryText_After", dtGlc)

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
Text to capture is displayed before occurrence / Text to capture is displayed after occurrence	<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>When you are creating a text area output value, the Text Before and Text After values are not captured. You can enter the text by typing or copying it into the Constant box.</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 758.</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 687.

Outputting Table Values

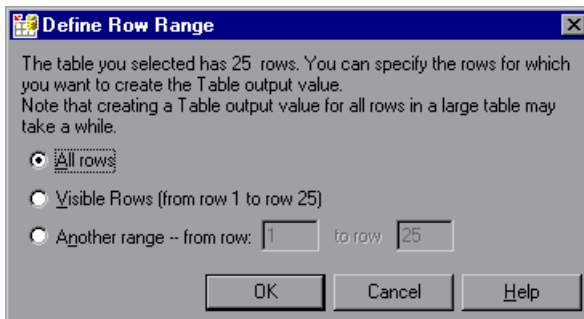
You can output the values of table cells and table properties while recording or editing your test. You specify the values to output using the Table Output Value Properties dialog box.

To output table values while recording:



- 1** Select **Insert > Output Value > Standard Output Value** or select **Standard Output Value** from the **Insert Checkpoint or Output Value** button. The QuickTest window is hidden, and the pointer changes into a pointing hand. For more information on using the pointing hand feature, see “Tips for Using the Pointing Hand While Recording” on page 702.
- 2** Click the table from which you want to output cell values. The Object Selection - Output Value Properties dialog box opens.
- 3** Select a table item from the displayed object tree and click **OK**. If the Table Output Value Properties dialog box opens, skip to step 4.

Otherwise, for certain objects in certain environments, such as WinList View objects, the Define Row Range dialog box opens.



Select the range of rows you want to include in your output value. You can include:

- **All rows.** Includes all of the rows in the table. Note that capturing all of the data for large table or list view objects may take some time.
- **Visible Rows (from row X to row Y).** Includes only the rows visible on the screen. Note that this option may not be available for some environments or object types.
- **Another range -- from row _ to row _.** You can specify any row range in the table.

Click **OK**. The Define Row Range dialog box closes, and the Table Output Value Properties dialog box displays the rows you specified (above the grid area).

- 4** In the Table Output Value Properties dialog box, specify the settings for the output value. For information on specifying the table content to output, see “Outputting Table Content” on page 703. For information on specifying the object properties to output, see “Outputting Table Properties” on page 709.

Note: For some environments, the Table Output Value Properties dialog box contains two tabs: Table Content and Properties. For other environments, the Table Output Value Properties dialog box displays only the options available in the Table Content tab, but does not contain any tabs.

- 5** Click **OK** to close the dialog box. An output value statement is added for the selected object in the Keyword View and Expert View.

To add a table output value while editing:

- 1** Depending on whether the object from which you want to output a value is already in a step, do one of the following:

- If you have already recorded a step on the object from which you want to output values, right-click the step and select **Insert Output Value**. Alternatively, select the step and select **Insert > Output Value > Standard Output Value**.

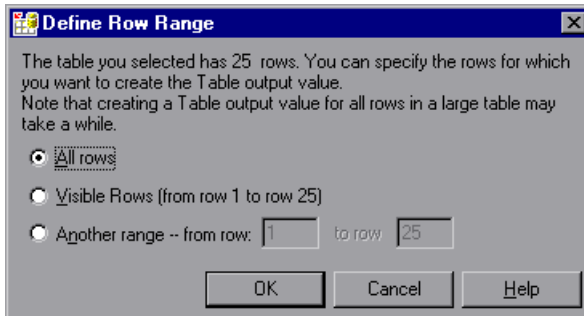


- If you have not recorded a step on the object from which you want to output values, make sure the **Active Screen** button is selected and the Active Screen is visible. Click a step in your test where you want to add an output value. The Active Screen displays the application screen corresponding to the highlighted step. Right-click the table in the Active Screen and select **Insert Output Value**. The Object Selection - Output Value Properties dialog box opens. Select a table item from the displayed object tree and click **OK**.

Note: In some environments, you must have the table open in your application to output a value from it.

- 2** If the Table Output Value Properties dialog box opens, skip to step 3.

Otherwise, for certain objects in certain environments, the Define Row Range dialog box opens.



Select the range of rows you want to include in your output value. You can include:

- **All rows.** Includes all of the rows in the table. Note that capturing all of the data for large table or list view objects may take some time.
- **Visible Rows (from row X to row Y).** Includes only the rows visible on the screen. Note that this option may not be available for some environments or object types.
- **Another range -- from row X to row Y.** You can specify any row range between 1 and the number of rows listed in the table.

Click **OK**. The Define Row Range dialog box closes, and the Table Output Value Properties dialog box displays the rows you specified (above the grid area).

- 3** In the Table Output Value Properties dialog box, specify the settings for the output value. For information on specifying the table content to output, see “Outputting Table Content” on page 703. For information on specifying the object properties to output, see “Outputting Table Properties” on page 709.

Note: For some environments, the Table Output Value Properties dialog box contains two tabs: Table Content and Properties. For other environments, the Table Output Value Properties dialog box displays only the options available in the Table Content tab, but does not contain any tabs.

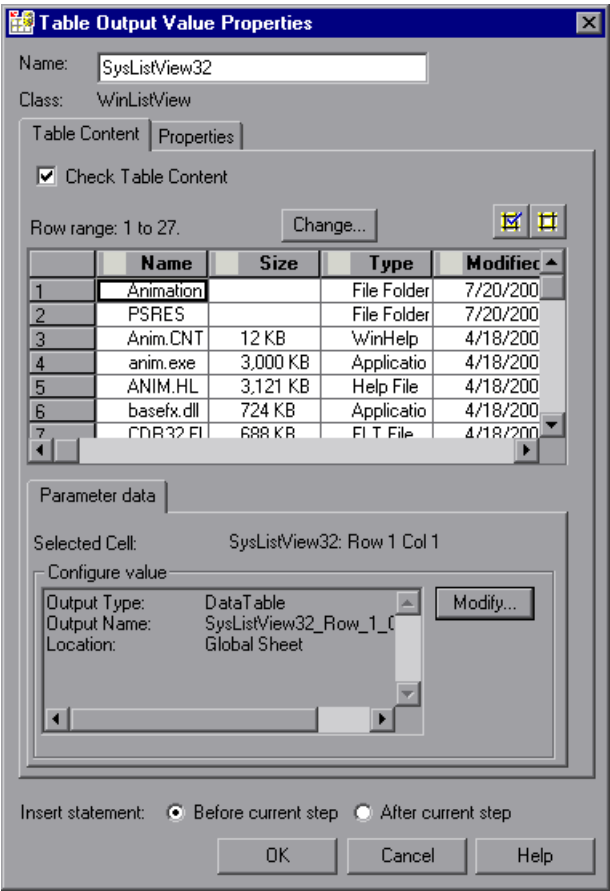
- 4** Click **OK** to close the dialog box. An output value statement is added for the selected object.

Tips for Using the Pointing Hand While Recording

- You can hold the left CTRL key to change the pointing hand to a standard pointer. You can then change the window focus or perform operations in QuickTest or in your application, such as right-clicking, using the scroll bars, or moving the pointer over an object to display a context menu.
- If the window containing the object you want to select is partially hidden by another window, hold the pointing hand over the partially hidden window for a few seconds until it comes to the foreground. Then point to and click the required object. You can configure the length of time required to bring a window into the foreground using the General pane of the Options dialog box.
- If the window containing the object you want to select is fully hidden by another window, or if a dialog box is hidden behind a window, press the left CTRL key and arrange the windows as needed.
- If the window containing the object you want to select is minimized, you can display it by holding the left CTRL key, right-clicking the application in the Windows task bar, and choosing **Restore** from the context menu.
- If the object you want to select can be displayed only by performing an event (such as right-clicking or moving the pointer over an object to display a context menu), hold the left CTRL key. The pointing hand temporarily turns into a standard pointer and you can perform the event. When the object you want to select is displayed, release the left CTRL key. The pointer becomes a pointing hand again.

Outputting Table Content

You can specify the table cells whose content you want to output. Depending on the environment, you do this either in the Table Content tab of the Table Output Value Properties dialog box—or directly in the Table Output Value Properties dialog box, if the dialog box does not contain any tabs.



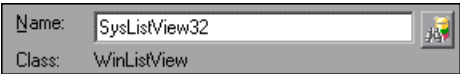
Notes:


- Some of the options shown in this example are available only in certain environments and only for certain objects.
 - For some environments, you can also specify object properties (using the Properties tab).
-

This section describes the general settings and options displayed in the Table Output Value Properties dialog box. Most of the options described in this section are available regardless of whether the Table Output Value Properties dialog box contains tabs.

Identifying the Output Value

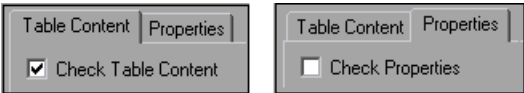
The top part of the Table Output Value Properties dialog box contains the following options:



Name	<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:</p> <ul style="list-style-type: none">➤ is unique➤ does not begin or end with a space➤ does not contain " (double quotation mark)➤ does not contain the following character combinations: := @@
Class	<p>Specifies the type of object (read-only). This may be a table-type object or a list view-type object.</p>
Find in Repository button 	<p>Displays the output value in its repository.</p> <p>Note: This option is not available when creating a new output value. It is available only when editing an existing output value.</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.

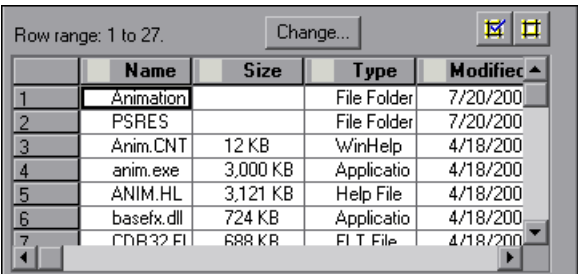


Check Table Content check box	(Table Content tab) Selecting the Check Table Content check box instructs QuickTest to output the values of the selected cells in the table object. (Selected by default)
Check Properties check box	(Properties tab) Selecting the Check Properties check box instructs QuickTest 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.

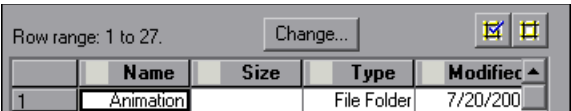
Choosing Cells for Output Values

The top part of the dialog box displays a grid representing the cells in the captured table. The column header names are captured from the table you selected for your output value step. You can output the values for one or more cells in the grid.



Tip: You can change the column widths and row heights of the grid by dragging the column and row header dividers.

Some environments and objects support selecting a row range. This enables you to specify which rows are displayed in the grid area. If row range selection is supported, the row range you specify when creating the output value is displayed above the grid:



Clicking the **Change** button enables you to modify the row range. (Depending on the environment, your application may need to be open and the relevant table displayed to modify the row range.) For more information, see “Modifying a Table Output Value” on page 711.

To choose a cell for a value to output:

Double-click the cell or select it and click the **Add Output Value** button (located above the grid, on the right). An output value icon is added to the cell.

To remove a cell from an output value:

Double-click the cell again or select it and click the **Remove Output Value** button (located above the grid, on the right). The output value icon is removed from the cell.

Specifying the Settings for the Output Value

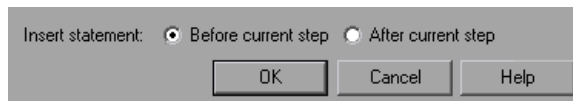
When a value in a table cell is first selected for output, the default output definition for the value is displayed in the **Configure value** area. For more information on default output definitions, see “Understanding Default Output Definitions” on page 683.

When you select a value in a table cell, you can:

- accept the displayed output definition by selecting another cell or by clicking **OK**.
- change the output type and/or settings for the selected value by clicking the **Modify** button. The Output Options dialog box opens and displays the current output type and settings for the value. For more information, see “Specifying the Output Type and Settings” on page 683.

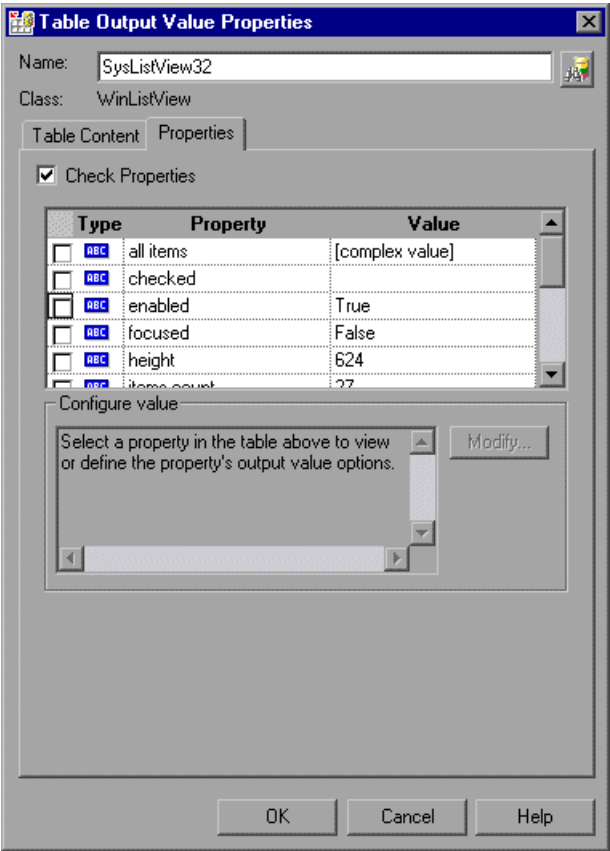
Specifying the Location for the Output Value Step

If the **Insert statement** area is displayed at the bottom of the dialog box, you can specify where the new output value step should be inserted in your test. For more information, see “Selecting the Location for the Output Value Step” on page 687.



Outputting Table Properties

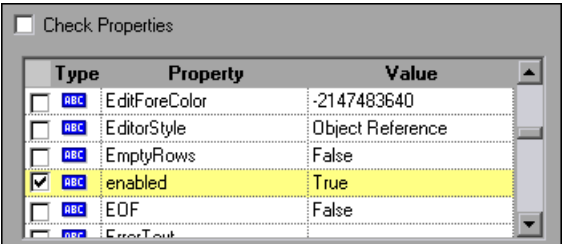
For certain environments, you can specify which object property values you want to output. By default, when you create a table output value on an object, QuickTest captures all the object's properties, but does not select any properties to output.



Note: For information on general table output value options, such as **Name** and **Class**, and on the options available in the Table Content tab, see “Outputting Table Content” on page 703.






Selecting Properties to Output

When you create a table output value, the Properties pane displays the object's default properties, including the properties, their values, and their types.



You instruct QuickTest to output specific properties by selecting the **Check Properties** check box. (This check box is cleared by default.)

The Properties pane for the object contains the following:

Check box	To output a property, select the corresponding check box. To remove a property from the output, clear the corresponding check box.
Type	The  icon indicates that the value of the property is currently a constant. The  icon indicates that the value of the property is currently a test or action parameter. The  icon indicates that the value of the property is currently a Data Table parameter. The  icon indicates that the value of the property is currently an environment variable parameter. The  icon indicates that the value of the property is currently a random number parameter.
Property	The name of the property.
Value	The expected value of the property. For more information on modifying the value of a property, see “Setting Values in the Configure Value Area” on page 757.

Modifying a Table Output Value

You can modify the table output value options, which specify where an output value is stored and how it can be used during the run session. You can also modify the number of rows for which QuickTest can output the values of specific table cells.

To modify the table output value options:

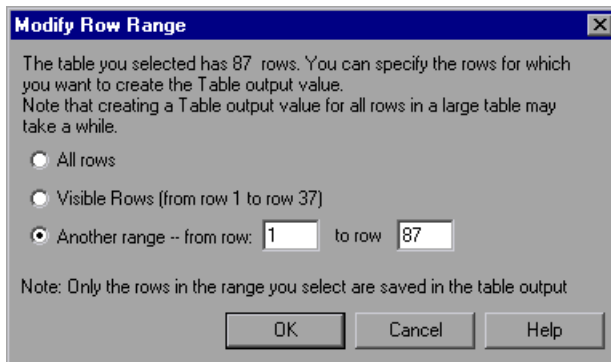
- 1** In the Keyword View or Expert View, right-click the Output CheckPoint step for the table whose output options you want to modify and select **Output Value Properties**. Alternatively, select the step containing the Output CheckPoint and select **Edit > Step Properties > Output Value Properties**. The Table Output Value Properties dialog box opens.
- 2** Perform one of the following:
 - If the Table Output Value Properties dialog box does not contain tabs, click the **Modify** button. The Output Options dialog box opens.
 - If the Table Output Value Properties dialog box contains tabs:
 - To modify the output options for the table content, make sure the Table Content tab is displayed and click the **Modify** button.
 - To modify the output options for the object properties, select the Properties tab and click the **Modify** button.The Output Options dialog box opens.
- 3** Modify the output value, as needed. For more information, see “Specifying the Output Type and Settings” on page 683.
- 4** You can also rename the output value, if needed. For more information, see “Identifying the Output Value” on page 680.

To modify the range or number of rows in an existing table output value:

- 1 Open the application containing the table or list view object from which you want to output a value and display the object in the application.
- 2 In the Keyword View or Expert View, right-click the Output CheckPoint step for the table whose row range you want to modify and select **Output Value Properties**. Alternatively, select the step containing the Output CheckPoint and select **Edit > Step Properties > Output Value Properties**. The Table Output Value Properties dialog box opens, displaying the currently selected row range.



- 3 In the Table Content tab, click the **Change** button at the top of the dialog box (above the grid area). The Modify Row Range dialog box opens.



- 4 Select the range of rows you want to include in your output value step. You can include all the rows, only the visible rows, or another range that you specify.

Note: The **Visible Rows** option may not be available for some environments or object types.

- 5** Click **OK**. The Modify Row Range dialog box closes, and the Table Output Value Properties dialog box displays the rows you specified in the Modify Row Range dialog box.
- If your modified row range includes new rows, you can select the cells from which you want to output values from the newly selected rows. The cells whose values you select will be outputted during the run session.
 - If your modified row range includes some or all of the rows that you specified previously, the cells whose values you selected to output will be outputted during the run session.
 - If your modified row range excludes some or all of the rows that were selected previously, the values of any previously selected cells in those rows will not be outputted during the run session.

Note: You can output values only from cells that are included in the specified row range.

Outputting Database Values

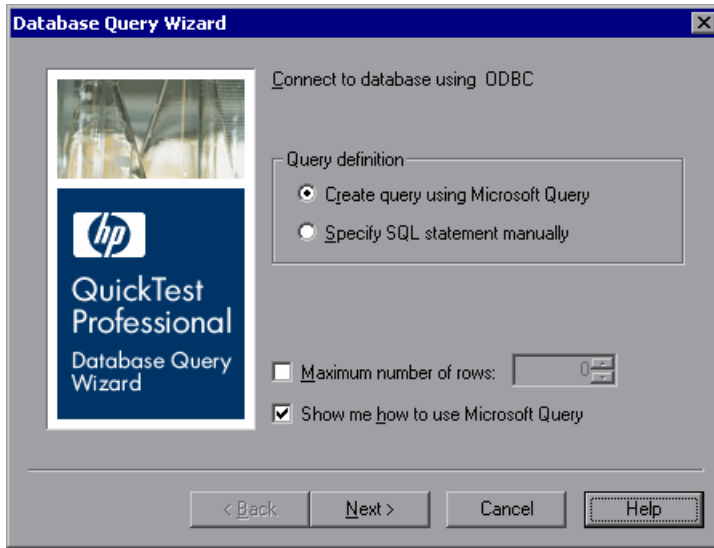
You can create database output values by defining a query to retrieve data from the database and selecting the values you want to output from the query result set. You can then specify the output settings for the selected values. During the run session, QuickTest captures the current data from the database and outputs the values according to the specified settings.

You can create database output values while recording or editing your test.

To create database output values:



- 1 Select **Insert > Output Value > Database Output Value**. The Database Query Wizard opens.



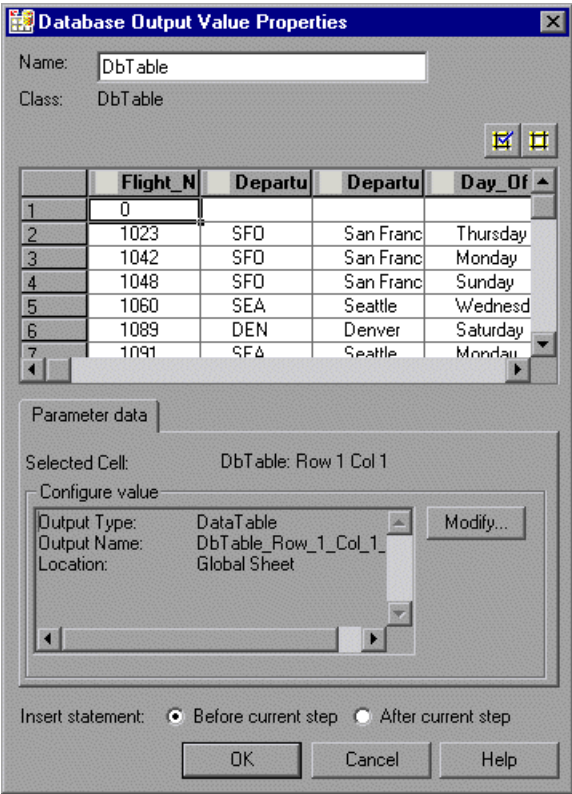
- 2 Use the wizard to define a query to retrieve the data that you want to output. Follow the instructions for creating a database checkpoint in “Creating a Check on a Database” on page 576.

After you finish defining your query, the Database Output Value Properties dialog box opens.

- 3 Specify the values to output and their settings. For more information, see “Defining Database Output Values” on page 715.
- 4 When you finish defining the output value details, click **OK**. QuickTest inserts an output value step in your test.

Defining Database Output Values

The Database Output Value Properties dialog box enables you to select the database cells for the values you want to output. You can define the output settings for each value that you select.



Identifying the Database Output Value

The top part of the Database Output Value Properties dialog box contains the following options:



Name	<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:</p> <ul style="list-style-type: none">➤ is unique➤ does not begin or end with a space➤ does not contain " (double quotation mark)➤ does not contain the following character combinations: := @@
Class	<p>Specifies the type of test object (read-only) for which you are performing the output value step.</p>
Find in Repository button	<p>Displays the output value in its repository.</p> <p>Note: This option is not available when creating a new output value. It is available only when editing an existing output value.</p>

Choosing Cells for Output Values

The top part of the dialog box displays a grid representing the cells in the captured database query results set. You can output the values for one or more cells in the grid.

Tip: You can change the width of the columns and the height of the rows in the grid by dragging the boundaries of the column and row headers.

To choose a cell for a value to output:

Double-click the cell or select it and click the **Add Output Value** button (located above the grid, on the right). An output value icon is added to the cell.

To remove a cell from an output value:

Double-click the cell again or select it and click the **Remove Output Value** button (located above the grid, on the right). The output value icon is removed from the cell.

Specifying the Settings for the Output Value

When a value in a database cell is first selected for output, the default output definition for the value is displayed in the **Configure value** area. For more information on default output definitions, see “Understanding Default Output Definitions” on page 683.

When you select a value in a database cell, you can:

- accept the displayed output definition by selecting another cell or by clicking **OK**.
- change the output type and/or settings for the selected value by clicking the **Modify** button. The Output Options dialog box opens and displays the current output type and settings for the value. For more information, see “Specifying the Output Type and Settings” on page 683.

Specifying the Location for the Output Value Step

If the **Insert statement** area is displayed at the bottom of the dialog box, you can specify where the new output value step should be inserted in your test. For more information, see “Selecting the Location for the Output Value Step” on page 687.

Outputting XML Values

You can create XML output value steps from any XML document contained in an XML Web page or frame, directly from an XML file, or from test objects that support XML. You can output element and/or attribute values in an XML output value step.

You can insert XML Web page or frame output value steps only while you are recording. You can create XML output value steps from an XML file or from a test object while recording or editing your test.

Note: XML Output Values are compatible with namespace standards and a change in namespace between nodes stored in the Output Properties dialog box XML tree and the actual values will result in a failed output value step.

For more information on XML standards, see <http://www.w3.org/XML/>

For more information on namespace standards, see <http://www.w3.org/TR/1999/REC-xml-names-19990114/>

To create XML output values from an XML Web page or frame:

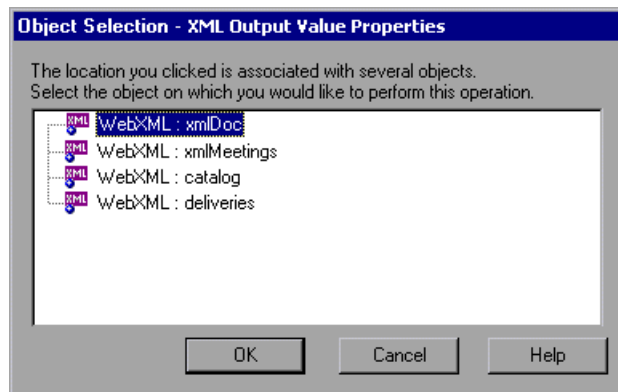


- 1 While recording, select **Insert > Output Value > XML Output Value (From Application)**, or click the **Insert Checkpoint or Output Value** button and select **XML Output Value (From Application)**. The pointer changes into a pointing hand. For more information on using the pointing hand, see “Tips for Using the Pointing Hand” on page 726.

Note: The **XML Output Value (From Application)** option is available only when the Web Add-in is installed and loaded. For more information on loading add-ins, see the section on working with QuickTest add-ins in the *HP QuickTest Professional Add-ins Guide*.

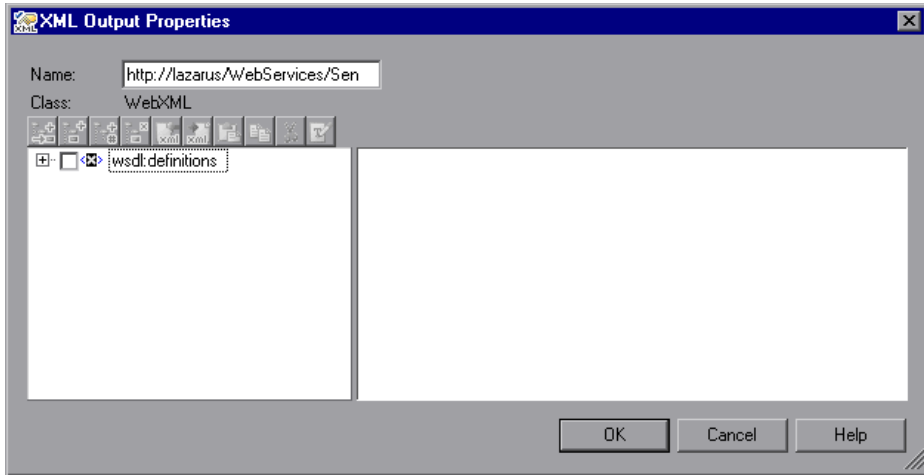
You can also insert a Web page or frame output value step using the **XML (From Resource)** option by selecting an existing WebXML test object. For more information, see creating XML output values from a test object that supports XML on page 723.

- 2** Click the XML object for which you want to specify an output value. If the location you clicked is associated with more than one object, the Object Selection - XML Output Value Properties dialog box opens.



- 3** Select the XML item you want to specify for the output value step.

- 4 Click **OK**. The XML Output Properties dialog box opens.



The XML Output Properties dialog box displays the element hierarchy and values (character data) of the selected XML document.

In the **Name** box, either accept the name that QuickTest assigns to the output value step or specify another name for it. By default, the output value name is the name of the test object on which the output value step is being performed.

If you rename it, make sure that the name:

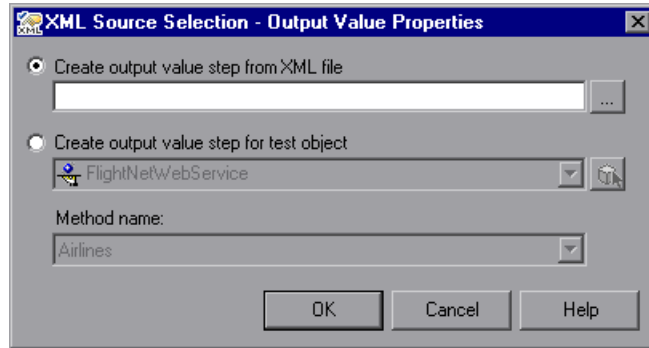
- is unique
- does not begin or end with a space
- does not contain " (double quotation mark)
- does not contain the following character combinations:
:=
@@

- 5 Select the items to output. For more information, see “Understanding the XML Output Properties Dialog Box” on page 727.
- 6 When you finish defining the output value details, click **OK**. QuickTest inserts an output value step in your test.

To create an XML output value step from an XML file:



- 1 Select **Insert > Output Value > XML Output Value (From Resource)**, or click the **Insert Checkpoint or Output Value** button and select **XML Output Value (From Resource)**. The XML Source Selection - Output Value Properties dialog box opens.



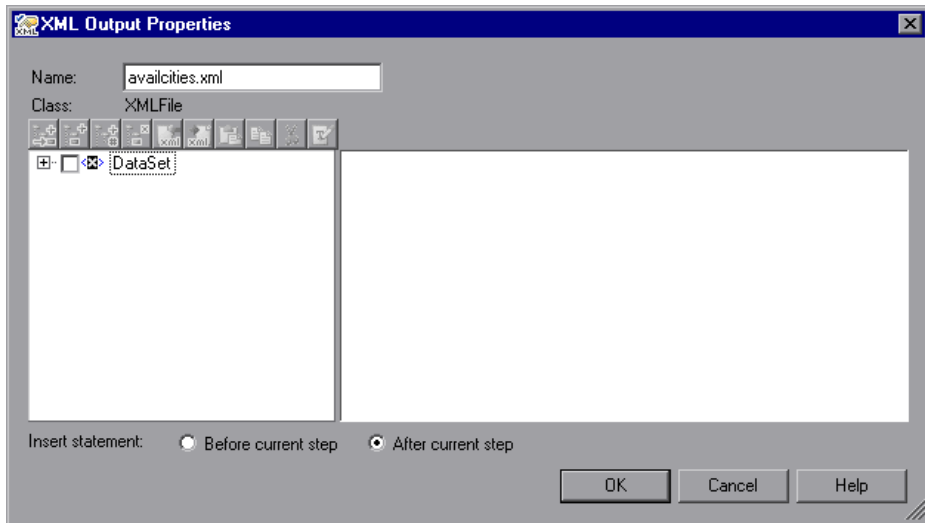
Tip: You can also insert an XML File output value step by selecting an existing XMLFile test object. For more information, see creating XML output values from a test object that supports XML on page 723.

- 2 Select **Create output value step from XML file**. Enter the Internet address or file path of the XML file.

Alternatively, click the browse button to open the Open XML File dialog box. In the sidebar, select the location of the XML file, and then navigate to the XML file for which you want to create an output value. You can specify an XML file either from your file system or from Quality Center. Select the file and click **Open**. The file path and name are entered in the box.

Note: You can enter a relative path and QuickTest will search for the XML file in the folders listed in the Folders pane of the Options dialog box. Once QuickTest locates the file, it saves it as an absolute path and uses the absolute path during the test run. For more information, see “Setting Folder Testing Options” on page 1237.

- 3 Click **OK**. The XML Output Properties dialog box opens.



The XML Output Properties dialog box displays the element hierarchy and values (character data) of the selected XML document.

In the **Name** box, either accept the name that QuickTest assigns to the output value step or specify another name for it. By default, the output value name is the name of the test object on which the output value step is being performed.

If you rename it, make sure that the name:

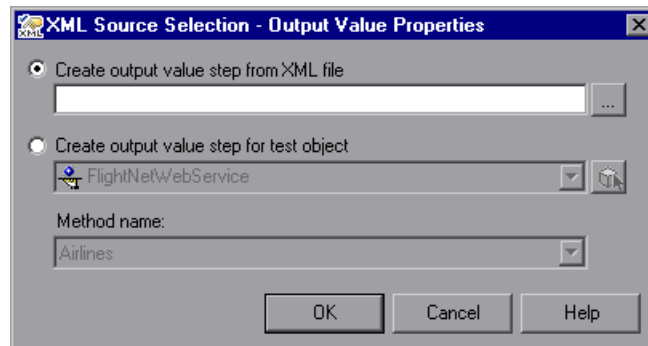
- is unique
- does not begin or end with a space
- does not contain " (double quotation mark)
- does not contain the following character combinations:
:=
@@

- 4 Select the items to output and the location for the output value step. For more information, see “Understanding the XML Output Properties Dialog Box” on page 727.
- 5 When you finish defining the output value details, click **OK**. QuickTest inserts an output value step in your test.

To create an XML output value step from a test object that supports XML:



- 1 Select **Insert > Output Value > XML Output Value (From Resource)**, or click the **Insert Checkpoint or Output Value** button and select **XML Output Value (From Resource)**. The XML Source Selection - Output Value Properties dialog box opens.



- 2 Select **Create output value step for test object** and select the test object from which you want to output values.



To select an object that is not displayed in the list, click **Object from Repository**. Then select an XML test object from the object repository on which to create a new output value step. The selected object must support XML.

You can select an existing WebXML or XMLFile test object type or you can select a WebService test object.

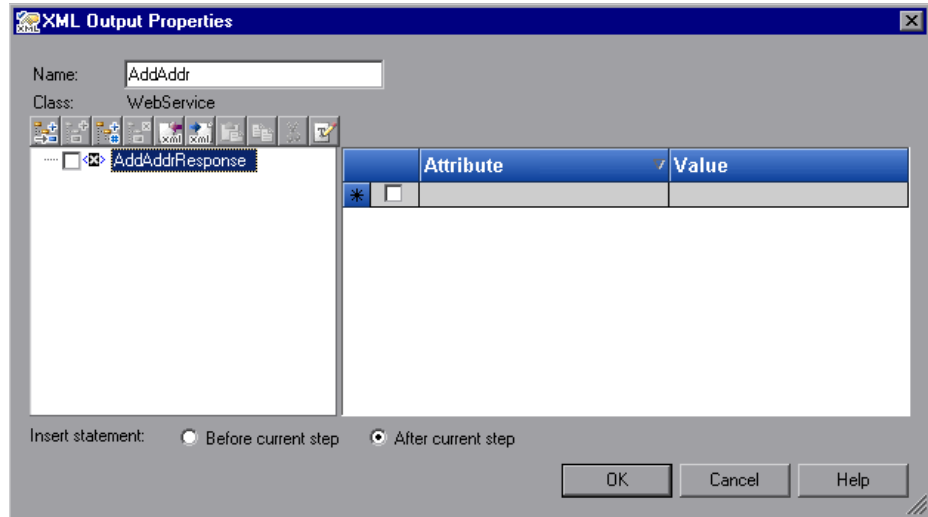
Note: Selecting a WebXML or XMLFile test object is identical to using the **XML Output Value (From Application)** or **XML Output Value (From Resource)** options, but may be faster than browsing to these objects and can be inserted while recording or editing. However, to use this option, the XML source must be available when you select the test object (the Web page must be open or the file must exist in the same location as when the test object was defined).

- 3 If you select a WebService test object, then the **Method name** box is enabled. Select the Web service operation whose return values you want to check.

Notes:

- The **Method name** box is available only if the Web Services Add-in is installed and loaded. The **Method name** box is enabled only if you select a WebService test object.
 - XML output value steps on Web service operations retrieve the values returned from the last Web service operation performed on the test object. If a different Web service operation step is performed prior to the output value step, then the output value step will fail.
-

- 4 Click **OK**. The XML Output Properties dialog box opens.



The XML Output Properties dialog box displays the element hierarchy in an XML tree, and the attributes and values (if any) of the selected XML output.

When you create an XML output value for an operation return value, only a generic XML tree is created and shown in the XML Output Properties dialog box. Before you can select which element or attribute values you want to output, you must populate the XML tree with the actual elements, attributes, and values. For more information, see “Updating the XML Hierarchy for XML Test Object Operation Output Value Steps (For WebService Test Objects Only)” on page 732.

- 5 In the **Name** box, either accept the name that QuickTest assigns to the output value step or specify another name for it. By default, the output value name is the name of the test object on which the output value step is being performed.

If you rename it, make sure that the name:

- is unique
- does not begin or end with a space
- does not contain " (double quotation mark)
- does not contain the following character combinations:
 - :=
 - @ @

Select the items to output and the location for the output value step. For more information, see “Understanding the XML Output Properties Dialog Box” on page 727.

- 6 When you finish defining the output value details, click **OK**. QuickTest inserts an output value step in your test.

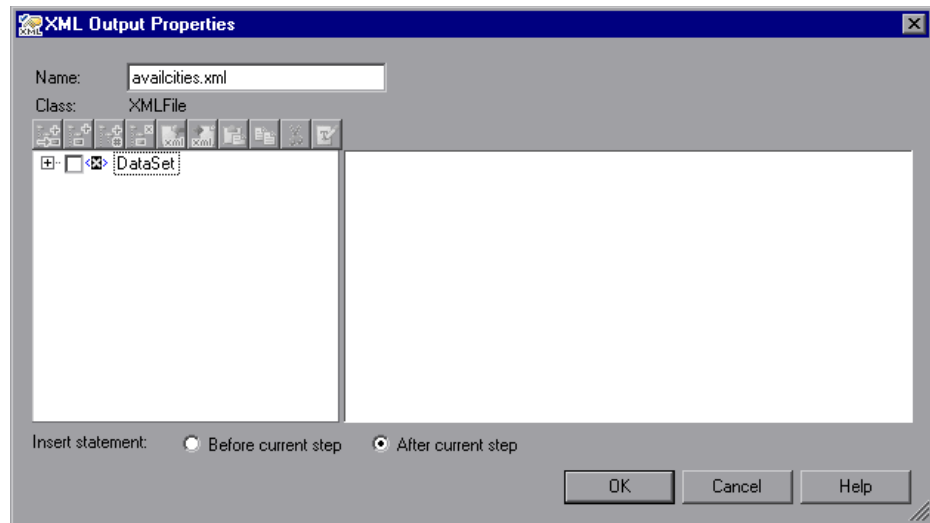
Tips for Using the Pointing Hand

- You can hold the left CTRL key to change the pointing hand to a standard pointer. You can then change the window focus or perform operations in QuickTest or in your application, such as right-clicking, using the scroll bars, or moving the pointer over an object to display a context menu.
- If the window containing the object you want to select is partially hidden by another window, hold the pointing hand over the partially hidden window for a few seconds until it comes to the foreground. Then point to and click the required object. You can configure the length of time required to bring a window into the foreground using the General pane of the Options dialog box.
- If the window containing the object you want to select is fully hidden by another window, or if a dialog box is hidden behind a window, press the left CTRL key and arrange the windows as needed.
- If the window containing the object you want to select is minimized, you can display it by holding the left CTRL key, right-clicking the application in the Windows task bar, and choosing **Restore** from the context menu.

- If the object you want to select can be displayed only by performing an event (such as right-clicking or moving the pointer over an object to display a context menu), hold the left CTRL key. The pointing hand temporarily turns into a standard pointer and you can perform the event. When the object you want to select is displayed, release the left CTRL key. The pointer becomes a pointing hand again.

Understanding the XML Output Properties Dialog Box


The XML Output Properties dialog box enables you to choose which element and/or attribute values to output and to define the output settings for each value that you select.



Identifying the Object











The top part of the XML Output Properties dialog box displays information on the test object for which you are creating an output value step:



Item	Description
Name	<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:</p> <ul style="list-style-type: none">➤ is unique➤ does not begin or end with a space➤ does not contain " (double quotation mark)➤ does not contain the following character combinations: := @@
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>
Find in Repository button 	<p>Displays the output value in its repository.</p> <p>Note: This option is not available when creating a new output value. It is available only when editing an existing output value.</p>



Modifying the XML Tree

The following commands are available according to the node you select in the tree:

Command	Icon	Description
Add Child		Adds a child node below the selected node in the tree.
Insert Sibling		Adds a sibling node at the same level as the selected node in the tree.
Add Value		Enables you to assign a constant or parameterized value to the selected element.
Delete		Deletes the selected node. Note that you cannot delete the root node of the output value step.
Import XML		Enables you to browse to and select a file structure from an existing XML file. The new file overrides the selected node's current sub-tree.
Export XML		Enables you to save the file structure of the selected node to an XML file.
Paste		Pastes a cut or copied node as a child node below the selected node in the XML tree. Note: You cannot paste an XML element node as its own descendant.
Copy		Makes a copy of the selected node, which you can then paste in another location in the XML tree.
Cut		Prepares the selected node to be cut and copies it to the clipboard. When you paste the node in the new location, it is removed from the original location in the XML tree.
Edit XML as Text		Opens the Edit XML as Text dialog box, enabling you to modify the XML text of the selected node and its subnodes in a text editor. For more information, see “Understanding the Edit XML as Text Dialog Box” on page 613.

Command	Icon	Description
Duplicate		<p>Adds a new node, identical to the selected one, as a sibling node at the same level as the selected node in the XML tree.</p> <p>Note: This command is available only from the context menu (right-click menu).</p>

XML Tree


The XML tree displays the hierarchical relationship between each element and value in the XML tree, enabling you to select the element and/or attribute values that you want to output. Each element node is displayed with a  icon. Each value node is displayed with a  icon.

Note: When you create an XML output value for an operation return value, only a generic XML tree is created and shown in the XML Output Properties dialog box. Before you can select which element or attribute values you want to output, you must populate the XML tree with the actual elements, attributes, and values. For more information, see “Updating the XML Hierarchy for XML Test Object Operation Output Value Steps (For WebService Test Objects Only)” on page 732.

Select an element node in the XML tree to display or set output options for its attributes and values on the right of the XML Output Properties dialog box. Select a value node in the XML tree to display or set output options for its value on the right of the XML Output Properties dialog box.

Tip: The XML tree pane and the **Attribute** and **Value** columns in the right pane are resizable.

To set output XML options:

- 1** Select the check box for an element or value node in the XML tree to indicate that you want to output a value for that node.
- 2** Select the element or value node to display or set output options for its attributes and/or values.
- 3** If you are outputting an element attribute, select the check box of the attributes for which you want output values.
- 4** Click in the **Value** column of an attribute, or click in the cell of an element value, and then click the **Output Options** button  to display the Value Configuration Options dialog box, which enables you to select or define the parameter in which you want to store the retrieved value.
- 5** In the Value Configuration Options dialog box, select the parameter type. Additional options are available for the output parameter type that you select. For more information on the options available for each parameter type, see:
 - **Data Table.** “Using Data Table Parameters” on page 639.
 - **Environment.** “Using Environment Variable Parameters” on page 645.
 - **Random Number.** “Using Random Number Parameters” on page 655.

Insert Statement Options

If you are inserting an output value step while editing your test, the bottom part of the XML Output Properties dialog box displays **Insert statement** options, enabling you to choose whether you want to insert the output value step before or after the step that you selected. Select **Before current step** if you want to insert the step before the highlighted step is performed. Select **After current step** if you want to insert the step after the highlighted step is performed.

Note: The **Insert statement** options are not available if you are adding a new output value step while recording or if you are modifying an existing output value step. They are available only if you are adding a new output value step while editing steps.

Updating the XML Hierarchy for XML Test Object Operation Output Value Steps (For WebService Test Objects Only)

This section is relevant only when working with XML output value steps on WebService test object operations (with the QuickTest Professional Web Services Add-in).

When you create an XML output value step for a test object operation (for a WebService test object), the XML tree of the operation return value data cannot be generated. Therefore, only a generic XML tree is created. To select the elements and attributes to output, you must first populate the XML tree with the actual elements, attributes, and values that the operation is expected to return.

You can use one of the three methods below to populate the XML tree:

- Updating the XML Tree Manually
- Importing an XML Tree from a File
- Updating the XML Tree Using Update Run Mode

Updating the XML Tree Manually

You can update the XML tree by adding elements, attributes, and values manually in the XML Output Properties dialog box.

To update the XML tree manually:

- 1** In the Keyword View, select the output value step whose XML tree you want to update. Click in the **Value** cell.



- 2** Click the **Output Properties** button or right-click and select **Output Value Properties**. The XML Output Properties dialog box opens.

- 3** Select a node in the XML tree and then click a toolbar button or choose an option from the context (right-click) menu to:



- Add an element at the same level as the selected node
- Add an element below the selected node
- Add a value to the selected node
- Edit the XML text of the selected node



- Copy the selected node



- Cut the selected node (the selected node is removed only after you paste it in another location)



- Paste a cut or copied node as a child node below the selected node



- Delete the selected node

- Duplicate the selected node, adding an identical node as a sibling node at the same level (this command is available only from the right-click menu)

For more information on the available tools in the XML Output Properties dialog box, see “Understanding the XML Output Properties Dialog Box” on page 727.

Importing an XML Tree from a File

You can import an XML tree from an existing file for a specific element in the XML tree hierarchy or for the whole tree.

To import an existing XML tree from a file:

- 1 In the Keyword View, select the output value step whose XML tree you want to update.



- 2 Click in the **Value** cell and then click the **Output Properties** button. The XML Output Value Properties dialog box opens.

- 3 If you want to import an XML hierarchy for the whole XML tree, select the root node. If you want to import an XML hierarchy for a specific element, select the element in the XML tree hierarchy.



- 4 Click the **Import XML** button. A message warns you that the imported hierarchy overwrites the selected node and its sub-tree. Click **Yes** to close the message.

- 5 In the Import XML from File dialog box, browse to the required XML file and click **Open**. The XML hierarchy is imported from the file.

Updating the XML Tree Using Update Run Mode

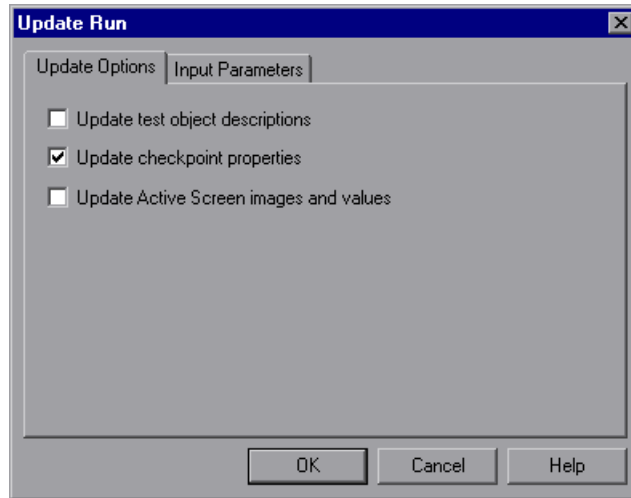
QuickTest cannot generate the return values of an operation when you insert an XML output value step on a Web service operation, but it can generate this information after it runs the operation. Therefore, you can run your Web service test in Update Run mode to automatically populate or update the elements, attributes and values in your XML tree.

To generate a new XML tree based on the current return values of the Web service operation, ensure that none of the node, attribute, or value check boxes are selected in the XML tree of the Output Value Properties dialog box.

Note: XML Output Value steps on Web service operations retrieve the values returned from the last Web service operation performed on the test object. If a different Web service operation step is performed prior to the output value step, then the output value step will fail.

To update an XML tree using Update Run mode:

- 1** Open a test containing an XML test object output value step for a Web service operation.
- 2** Click the down arrow next to the **Run** button in the toolbar and select **Update Run Mode**, or select **Automation > Update Run Mode**. The Update Run dialog box opens.



- 3** Select **Update checkpoint properties** and click **OK**. QuickTest runs the test and updates the XML hierarchy and values for each blank XML checkpoint and XML output value step in the test. It updates values only for XML checkpoints or output value steps that have one or more nodes selected.
- 4** If you want to confirm that QuickTest successfully updated your output value step, expand the tree in the Test Results window and select the XML output value step. Then check that **Update done** is displayed in the pane on the right. (If the Test Results window did not open automatically at the end of the run, click the **Results** button or select **Automation > Results**.)



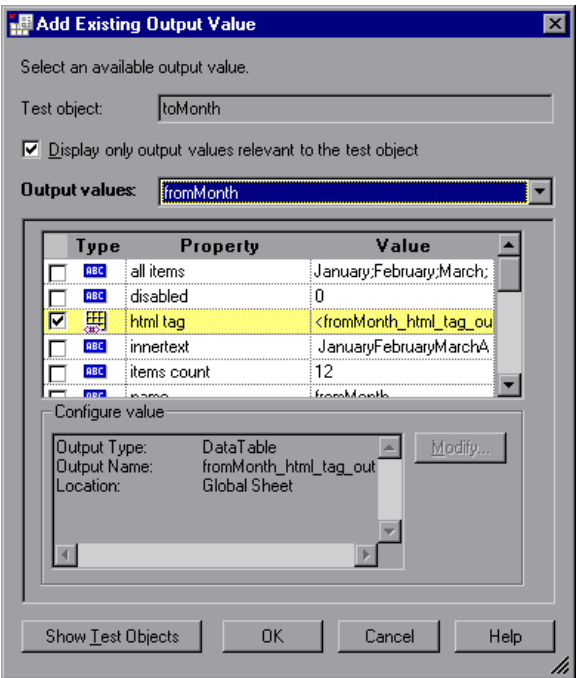
Adding Existing Output Values to a Test

QuickTest enables you to insert existing output values into your test.

When you insert an existing output value in your test, consider which output values should be used in multiple locations in your test. Each time an output value step is performed, the value contained in the output value is overwritten with the new output value. You should insert an existing output value into your test only if the stored value will no longer be needed by your test when the output value object is used again.

Understanding the Add Existing Output Value Dialog Box

You open the Add Existing Output Value dialog box by selecting **Insert > Output Value > Existing Output Value**. This option is available only if at least one of the object repositories associated with the current action (including the local object repository) contains at least one Output object.



If a step is highlighted in the Keyword View or the cursor is located in a step in the Expert View, the Add Existing Output Value dialog box opens with the **TestObjects** tree hidden.

The test object displayed in the **Test object** box is the object from the highlighted step in the Keyword View or the specific object where the cursor is located in the Expert View.

You can display or hide the **TestObjects** tree by clicking the **Show/Hide Test Objects** button.

The Add Existing Output Value dialog box contains the following options:

Option	Description
Test object	Specifies the test object for which you are adding an output value.
TestObjects tree	Displays the objects stored in the object repositories associated with the current action.
Show/Hide Test Objects	Shows or hides the TestObjects tree.
Display only output values relevant to the selected test object	<p>When selected, QuickTest determines which output value objects from the current action's object repositories are relevant for the selected object (based on the output value type and the properties selected to output in the output value object) and displays only those output value objects in the Output Values list.</p> <p>When using this option, it is recommended to open your application and display the selected object so that QuickTest can accurately determine all of the checkpoints that can apply to that object.</p>
Output values	<p>Lists the checkpoints available for insertion.</p> <p>If the Display only output values relevant to the selected test object option is cleared, this list includes all output value objects from all object repositories associated with the current action.</p> <p>If the Display only output values relevant to the selected test object option is selected, this list displays only the relevant output value objects as described above.</p>
Output Value Properties Area	Displays the output value options for the selected output value object in read-only format.

To insert an existing output value in your test:

- 1** Select the step after which you want to insert the checkpoint.
- 2** Select **Insert > Output Value > Existing Output Value**. The Add Existing Output Value dialog box opens.
- 3** If the **TestObjects** tree is displayed, select the object for which you want to insert an Output Value. Otherwise proceed to step 4.
- 4** From the **Output values** list, select the output value that you want to insert for the object displayed in the **Test object** box.
- 5** Click **OK**. The output value step is inserted after the current step.

26

Working with Text Recognition for Windows-Based Objects

QuickTest uses various mechanisms to identify the text strings in your Windows-based objects. This chapter describes how to configure QuickTest to optimize the results of text recognition for your Window-based objects.

This chapter includes:

- About Working with Text Recognition for Windows-Based Objects on page 742
- The Options Dialog Box: General > Text Recognition Pane on page 742
- Guidelines for Text Recognition on page 746
- Text Recognition and Development Environments on page 748
- Use-Case Scenario: Checking Text in an Image on page 750

About Working with Text Recognition for Windows-Based Objects

QuickTest identifies text in your application using either a Windows API-based mechanism or an OCR (optical character recognition) mechanism. You can use the text and text area checkpoint or output value commands to verify or retrieve text in your Windows-based objects. Alternatively, you can use the *testobject.GetText* (for Terminal Emulator objects), *testobject.GetVisibleText*, or *testobject.GetTextLocation* test object methods, or the **TextUtil.GetText** or **TextUtil.GetTextLocation** reserved object methods to capture the text you need.

By default, QuickTest tries to retrieve the text directly from the object using a Windows API-based mechanism. If QuickTest cannot capture the text this way (for example, because the text is part of a picture), it tries to capture the text using an OCR (optical character recognition) mechanism. You use the Text Recognition pane to specify the preferred text recognition mechanism and OCR-specific settings.

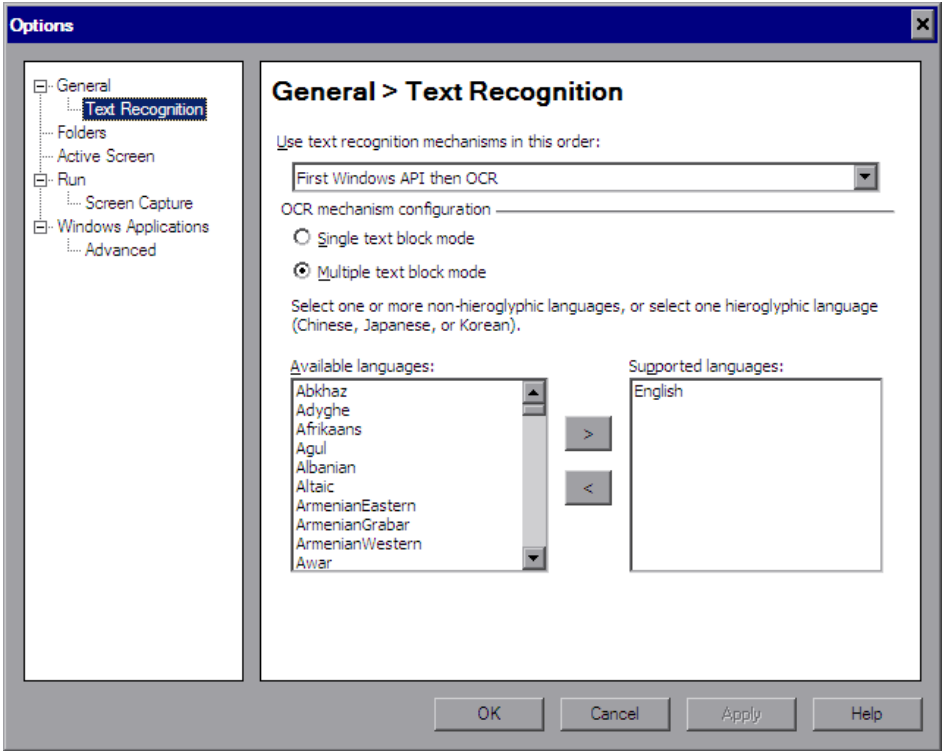
Before you insert a text / text area checkpoint or output value, review the “Guidelines for Text Recognition” on page 746.

The Options Dialog Box: General > Text Recognition Pane

Description	Enables you to configure how QuickTest identifies text in your application. You can use this pane to modify the default text capture mechanism, OCR (optical character recognition) mechanism mode, and the language dictionaries the OCR mechanism uses to identify text.
Accessed by	Tools menu > Options item > General node > Text Recognition node

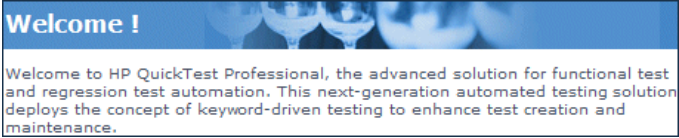
Important Information	The General > Text Recognition options are relevant only for Windows-based objects, such as Standard Windows, .NET WinForms, WPF, SAP Gui for Windows, Visual Basic, and ActiveX.
Learn More	Conceptual Overview: “About Working with Text Recognition for Windows-Based Objects” on page 742 Additional related topics: “Additional References” on page 746

Below is an image of the General > Text Recognition pane of the Options dialog box:



The General > Text Recognition pane options include:

Option	Description
Use text recognition mechanisms in this order	<p>Specifies the text recognition mechanism that QuickTest uses when capturing text.</p> <p>Possible values:</p> <p>First Windows API then OCR. (Default) Instructs QuickTest to first try to retrieve text directly from the object using the Windows API-based mechanism. If no text can be retrieved (for example, because the text is part of a picture), QuickTest tries to retrieve text using the OCR (optical character recognition) mechanism. (This setting is highly recommended when working with CJK (Chinese, Japanese, Korean) languages.)</p> <p>First OCR then Windows API. Instructs QuickTest to first try to retrieve text from the object using the OCR mechanism. If no text can be retrieved, then QuickTest uses its Windows API-based mechanism to retrieve text from the object.</p> <p>Use Only Windows API. Instructs QuickTest to use only the Windows API-based mechanism (and not the OCR mechanism) to retrieve text from the object.</p> <p>Use Only OCR. Instructs QuickTest to use only the OCR mechanism (and not the Windows API-based mechanism) to retrieve text from the object. (Required when working with Windows Vista.)</p> <p>For more information on text recognition support in Windows-based environments, see the <i>HP QuickTest Professional Readme</i>.</p>
Single text block mode	<p>Select this radio button if the text on the object is uniform in font, size, color, and background. For example:</p> <div data-bbox="505 1234 674 1274"><p>Welcome !</p></div> <p>The single text block mode instructs the OCR mechanism to focus on the area and treat it as a single text block. This is especially useful when trying to capture text on small objects or in a small text area.</p>

Option	Description
Multiple text block mode	<p>Select this radio button only if the text on the object comprises different fonts, font sizes, colors, and/or backgrounds. For example:</p>  <p>The multiple text block mode instructs the OCR mechanism to handle each text area in the object that has a different background font and size. The OCR mechanism decides where to divide the text blocks according to an internal algorithm.</p>
Available languages	<p>Lists all of the language dictionaries that the OCR mechanism can potentially use when retrieving text from the object.</p> <p>To specify the language dictionaries used by the OCR mechanism: Move a language to the Supported languages list box by selecting a language and clicking the right arrow button (>).</p>
Supported languages	<p>Lists the language dictionaries that the OCR mechanism uses when capturing text. The Supported languages list box can contain either:</p> <ul style="list-style-type: none"> ➤ One CJK (Chinese, Japanese, Korean) language. (Note: By default, English is also supported when capturing text in CJK languages.) ➤ One or more non-CJK languages. <p>To remove a language dictionary from the Supported languages list: Select the language and click the left arrow button (<).</p>

Additional References

Related Use Case Scenarios	“Use-Case Scenario: Checking Text in an Image” on page 750
Related Tasks	<ul style="list-style-type: none">➤ “Creating a Text Checkpoint” on page 552➤ “Creating a Text Area Checkpoint” on page 554➤ “Outputting Text Values” on page 688
Related Concepts	<ul style="list-style-type: none">➤ “Guidelines for Text Recognition” on page 746➤ “Checking Text” on page 551

Guidelines for Text Recognition

- When using the OCR mechanism, the larger the text, the better the text recognition.
- Try to keep the dimensions of the selected text area as small as possible, as this helps prevent additional unwanted characters in recognized text.

At the same time, consider the potential movement (change of coordinates) of the object within the window. For example, the screen resolution is often different on different computers, and this can affect the coordinates of the object in the application. Also, during the design and development stages of an application, an object may be moved to make room for other objects or for aesthetic purposes.

Consider that the operating system, installed service packs, installed toolkits, and so on, can all affect the size and location of an object in an application. Make sure that the dimensions of the selected text area are large enough for different system configurations.

The dimensions of the selected text area need to be large enough to take these issues into account.

- If you are not sure which text block mode to use, first use the single text block mode, as text captures performed on single text blocks are generally more accurate than text captures on multiple text blocks. If the results are not what you expect, then try using the multiple text block mode. For an example of when to use different text block modes, see “Use-Case Scenario: Checking Text in an Image” on page 750.

Tip: If you want to use the text recognition mechanism for a large area containing different fonts and backgrounds, it is recommended to create several steps to capture the text for each single text block instead of creating one step to capture a multiple text block.

- Windows provides various themes. When working with text recognition, try to apply themes in the following order:
 - Windows Vista theme (for best results)
 - Windows XP theme
 - Windows Classic theme
- If the text recognition mechanism retrieves unwanted text information (such as hidden text and shadowed text that appears as multiple copies of the same string), when using the multiple text block mode, use the single text block mode option.
- If your text recognition options are set to use the Windows API mechanism, then, when running a step that uses text recognition, the Windows API may cause a "blinking effect" in your application as it captures the text. If your test contains consecutive steps that utilize the text recognition mechanism, the "blinking effect" in one step may cause the subsequent text recognition step (or other step that relies on the appearance of the application, such as a bitmap checkpoint) to fail.

To address this, you can insert a **Wait** statement prior to each such step. This enables you to delay the performance of the next text recognition step until the Windows API capture of the previous step is complete.
- It is highly recommended to check text from your application window by inserting a standard checkpoint for the object containing the desired text, using its **text** (or similar) property.

Note: If you are creating text area checkpoints, see “Considerations for Defining the Text Area” on page 556 for additional guidelines.

Text Recognition and Development Environments

The following table lists the development environments supported by QuickTest (via its add-ins), and specifies what is supported for text recognition.

Development Environment	Text Recognition	
	Supported	Not Supported
ActiveX	Full text recognition support	N/A
Delphi	Full text recognition support	N/A
Java	<ul style="list-style-type: none">➤ Text checkpoints➤ Text output values➤ Text area checkpoints➤ Text area output values	<ul style="list-style-type: none">➤ GetTextLocation method➤ GetVisibleText method
.NET WebForms	<ul style="list-style-type: none">➤ Text checkpoints for Page object only➤ Text output values for Page object only	<ul style="list-style-type: none">➤ Text checkpoints for all other objects➤ Text output values for all other objects➤ Text area checkpoints for all other objects➤ Text area output values for all other objects➤ GetTextLocation method➤ GetVisibleText method
.NET WinForms	Full text recognition support	N/A
Oracle	N/A	No text recognition support

Development Environment	Text Recognition	
	Supported	Not Supported
PeopleSoft	<ul style="list-style-type: none"> ➤ Text checkpoints for PSFrame object only ➤ Text output values for PSFrame object only 	<ul style="list-style-type: none"> ➤ Text checkpoints for all other objects ➤ Text output values for all other objects ➤ Text area checkpoints for all other objects ➤ Text area output values for all other objects ➤ GetTextLocation method ➤ GetVisibleText method
PowerBuilder	Full text recognition support	N/A
SAP Gui for Windows	N/A	No text recognition support
SAP Web	<ul style="list-style-type: none"> ➤ Text checkpoints ➤ Text output values 	<ul style="list-style-type: none"> ➤ Text area checkpoints ➤ Text area output values ➤ GetTextLocation method ➤ GetVisibleText method
Siebel	N/A	No text recognition support
Standard Windows	Full text recognition support	N/A
Stingray	Full text recognition support	N/A
Terminal Emulators	Text output values for TeScreen and TeTextScreen objects only	<ul style="list-style-type: none"> ➤ Other text checkpoints ➤ Other text output values ➤ Text area checkpoints ➤ Text area output values ➤ GetTextLocation method ➤ GetVisibleText method
VisualAge	Full text recognition support	N/A
Visual Basic	Full text recognition support	N/A

Development Environment	Text Recognition	
	Supported	Not Supported
Web	<ul style="list-style-type: none">➤ Text checkpoints for Page object only➤ Text output values for Page object only	<ul style="list-style-type: none">➤ Text checkpoints for all other objects➤ Text output values for all other objects➤ Text area checkpoints for all other objects➤ Text area output values for all other objects➤ GetTextLocation method➤ GetVisibleText method
Web Services	N/A	No text recognition support
WPF	Full text recognition support	N/A

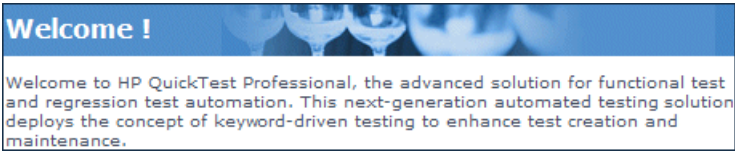
Use-Case Scenario: Checking Text in an Image

Ben and George are quality assurance engineers who are experienced QuickTest users. George is also familiar with text recognition and has a basic understanding of how text recognition mechanisms work.

Ben often uses bitmap checkpoints to test the appearance of different icons or pictures in the user interface he is testing.

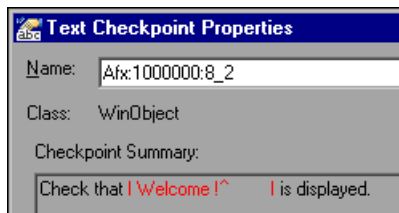
For one of his projects, Ben also needed to verify the text in the graphics, so he decided to use text checkpoints.

Ben decided to begin the verification process by inserting a text checkpoint to check that the text **Welcome !** was displayed correctly in the following graphic.



Before inserting the text checkpoint, Ben opened the Text Recognition pane and configured the text recognition settings. Ben set the text recognition mechanism to **Use Only OCR** because the text was part of a graphic. Ben also knew that single text block mode usually works best, so he selected the **Single text block mode** option.

Ben then inserted a text checkpoint on the entire area shown above and received the following results in the Text Checkpoint Properties dialog box:



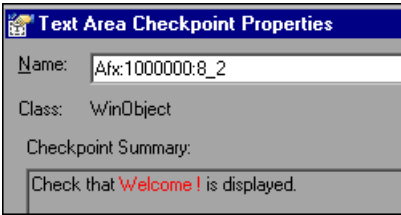
Ben noticed that there were extra characters in the **Checkpoint Summary** area of the text checkpoint, but he did not know why.

Ben asked his colleague, George, for help. George explained to him that the text recognition mechanism sometimes adds extra characters to the text checkpoint when it does not recognize the text correctly.

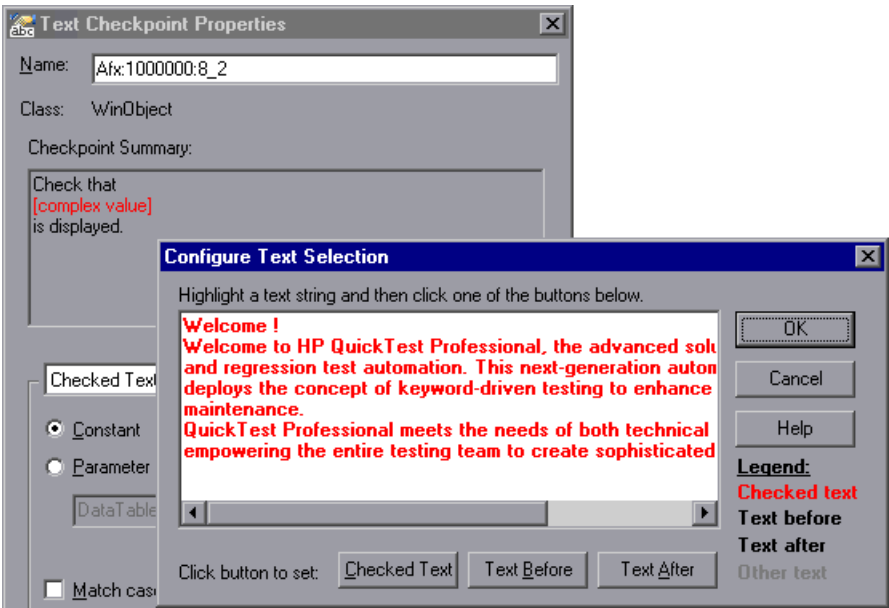
George also pointed out that the area Ben defined for the text checkpoint consisted of multiple text blocks because the text was not uniform in font size, color, or background. The title area consisted of white characters on a blue-gray background, while the remaining text was smaller and consisted of blue text on a white background.



Ben remembered that he had selected the **Single text block mode** option in the General > Text Recognition pane and understood that if he wanted to use single text block mode, he would have to create a text checkpoint only on the Welcome ! area of the graphic, and not on the entire graphic. Ben tried this, and the OCR mechanism correctly identified the text, as shown below:



Ben was pleased with the results, but he wanted to explore other possibilities, so he inserted another text checkpoint—this time on the entire graphic. He selected the **Multiple text block mode** option in the Text Recognition pane, which resulted in the following:



Ben was pleased that the OCR mechanism correctly recognized all of the text in the graphic. But he needed to test only the title, **Welcome !**, so he finalized this checkpoint by marking all of the text after **Welcome !** as **Text After**.

Even though both checkpoints passed, Ben needed only one text checkpoint. He decided to keep the first checkpoint (that used **Single text block mode**), and he deleted the second one. He selected the **Single text block mode** option in the Text Recognition pane to help ensure that the checkpoint would pass in future test runs.

27

Configuring Values

QuickTest enables you to configure the values for properties and other items by defining a value as a constant or a parameter. You can also use regular expressions in values to increase the flexibility and adaptability of your tests.

This chapter includes:

- About Configuring Values on page 755
- Configuring Constant and Parameter Values on page 756
- Understanding and Using Regular Expressions on page 762
- Defining Regular Expressions on page 765

About Configuring Values

Some dialog boxes, such as the Checkpoint Properties dialog boxes, include a **Configure value** area, in which you can define the value for a selected item as a constant or a parameter. In other contexts, such as the Keyword View, Step Generator, and Object Repository window, you can select a value directly and parameterize it or define it as a constant.

- **Constant.** A value that is defined directly in the step and remains unchanged for the duration of the test.
- **Parameter.** A value that is defined or generated separately from the step and is retrieved when the specific step runs. For example, a parameter value may be defined in an external file or generated by QuickTest.


When you define a value as a parameter, you can also specify other settings according to the parameter type. For more information on using parameters in your tests, see Chapter 24, “Parameterizing Values.”

You can edit a constant value in the **Configure value** area. In certain contexts, you can define a constant value using a regular expression.

A regular expression is a string that specifies a complex search phrase. Regular expressions are used to identify objects and text strings with varying values. For example, if the name of a window’s title bar changes according to a file name, you can use a regular expression in a test object description to identify a window whose title bar has the specified product name, followed by a hyphen, and then any other text.

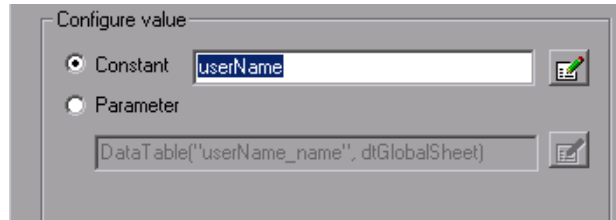
Configuring Constant and Parameter Values

You can define a value as a constant or a parameter in several ways:

- In the Value Configuration Options dialog box, you can click the parameterization button  for a selected value, for example, in the Keyword View, Step Generator, or Object Repository window. For more information, see “Configuring a Selected Value” on page 760.
- In the **Configure value** area of a dialog box, you can select a property or argument, for example, in the Checkpoint Properties dialog box.

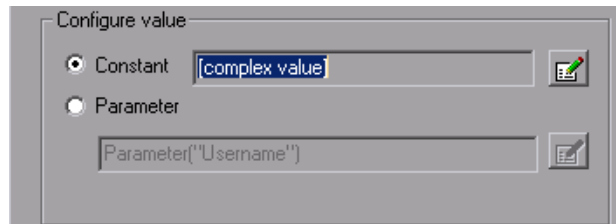
Setting Values in the Configure Value Area

When you select an item in a dialog box containing a **Configure value** area, such as the Checkpoint Properties dialog box, you can select **Constant** or **Parameter** to set the value. The default is **Constant**.



If you select **Constant**, you can edit a single-line value directly in the **Constant** box. If it is a **string** value, you can also click the **Constant Value Options** button to define the value as a regular expression. For information on regular expressions, see “Understanding and Using Regular Expressions” on page 762.

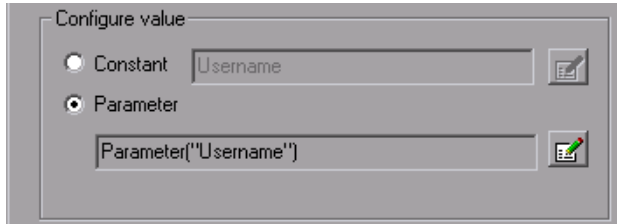
If the entire value cannot be displayed in the **Constant** box, it is shown as **[complex value]**. For example, the value of a list’s **all items** property is a multi-line value, where each line contains the value of an item in the list.



You can view or edit a complex value by clicking the **Constant Value Options** button. You can also define a complex value as a regular expression. For more information on editing constant values, see “Setting Constant Value Options” on page 759.

Configuring a Parameter Value

If you select **Parameter** for a value that is already parameterized, the **Parameter** box displays the current parameter definition for the value. If you select **Parameter** for a value that is not yet parameterized, the **Parameter** box displays the default parameter definition for the value.



For more information on default parameter definitions, see “Understanding Default Parameter Values” on page 634.



You can click the **Parameter Options** button to select a different parameter type or modify the parameter settings for the value.

The Parameter Options dialog box opens for the displayed parameter type. For more information on defining values for specific parameter types, see:

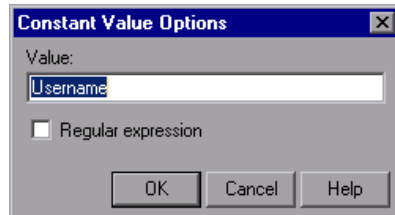
- “Setting Test and Action Parameter Options” on page 636
- “Setting Data Table Parameter Options” on page 641
- “Setting Environment Variable Parameter Options” on page 652
- “Using Random Number Parameters” on page 655

For more information on using parameters in your tests, see Chapter 24, “Parameterizing Values.”

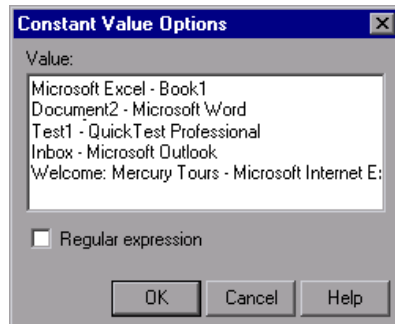
Setting Constant Value Options



When you click the **Constant Value Options** button in the **Configure value** area, the Constant Value Options dialog box opens.




For a complex value (a value that cannot be displayed entirely in the **Constant** box), the Constant Value Options dialog box expands to show the entire contents of the value.

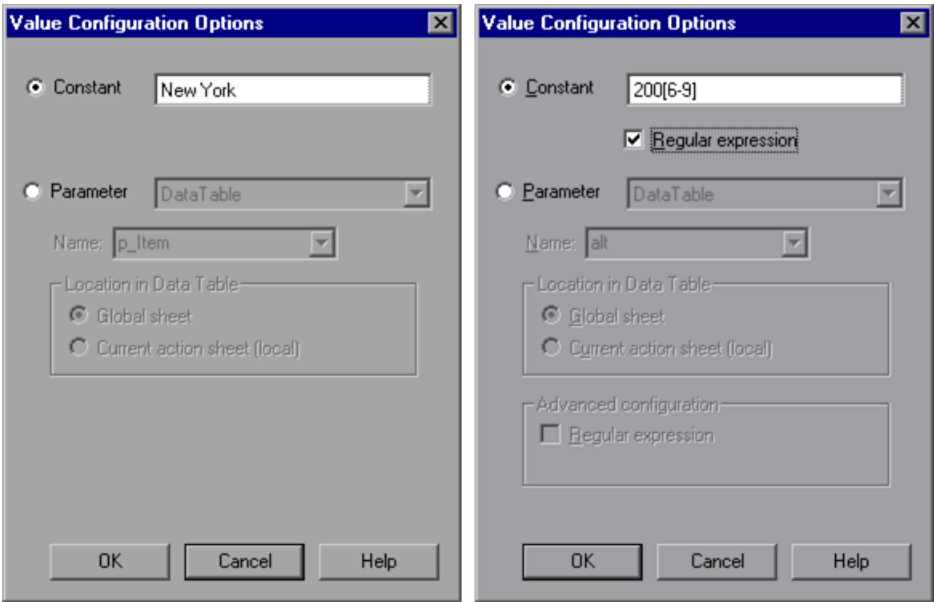


You can update the following options to edit the value of the constant:

- **Value.** Specifies the value for the constant.
- **Regular expression.** Sets the defined value as a regular expression:
 - For general information on regular expressions, see “Understanding and Using Regular Expressions” on page 762.
 - For information on defining a regular expression, see “Defining Regular Expressions” on page 765.


Configuring a Selected Value

When you click the parameterization button  for a selected value, the Value Configuration Options dialog box opens. In some situations, you can also define the constant or parameter using a regular expression. (The following examples illustrate the Value Configuration Options dialog box with and without the **Regular expression** check box.)



Note: The parameter options shown in this dialog box change according to the parameter type selected in the **Parameter** box.

You can select one of the following options:

- **Constant.** Defines a value that remains unchanged for the duration of the test. You can edit the value directly in the **Constant** box. This box offers the same editing options as the **Value** cell in which you clicked the parameterization button  to open this dialog box. For more information on these options, see “Defining Values for Your Step Arguments” on page 404.

In some situations, for example, when parameterizing an object identification property value, you can also specify a constant value using a regular expression (by using a regular expression in the **Constant** box and selecting the **Regular expression** check box). For information on regular expressions, see “Understanding and Using Regular Expressions” on page 762.

- **Parameter.** Specifies a value that is defined or generated separately from the step and is retrieved when the specific step runs.

If you select **Parameter** for a value that is already parameterized, the **Parameter** section displays the current parameter type and details for the value. If you select **Parameter** for a value that is not yet parameterized, the **Parameter** section displays the default parameter type and details for the value.

For more information on default parameter definitions, see “Understanding Default Parameter Values” on page 634.

You can change the default definition by selecting a different parameter type or modifying the parameter settings for the value. The options in the **Parameter** section change according to the parameter type you select.

Note: If you are using an environment variable to parameterize an argument that receives a predefined constant or number, only the environment variable parameters whose value is of type **integer** are shown in the **Name** list.

The **Parameter** section of the Value Configuration Options dialog box is very similar to the Parameter Options dialog box. For more information on configuring values for specific parameter types, see:

- “Defining the Settings for a Test or Action Parameter” on page 637
- “Defining the Settings for a Data Table Parameter” on page 642
- “Defining the Settings for an Environment Variable Parameter” on page 652
- “Defining Settings for a Random Number Parameter” on page 656

For more information on using parameters in your tests, see Chapter 24, “Parameterizing Values.”

Understanding and Using Regular Expressions

Regular expressions enable QuickTest to identify objects and text strings with varying values. You can use regular expressions in the following situations:

- When defining the property values of an object in dialog boxes or in programmatic descriptions
- When parameterizing a step
- When creating checkpoints with varying values

For example, you can use a regular expression if the text property of an object is a date value, but the displayed date changes according to the current date. If you define the date as a regular expression, QuickTest can identify the object that contains text with the expected date format, rather than the exact date value.

A regular expression is a string that specifies a complex search phrase. By using special characters, such as a period (.), asterisk (*), caret (^), and brackets ([]), you can define the conditions of a search.

Notes:

- You can use regular expressions only for values of type **string**.
- When any special character in a regular expression is preceded by a backslash (\), QuickTest searches for the literal character.

For more information and examples of the use of regular expressions, see:

- “Using Regular Expressions for Property Values” on page 763
- “Using Regular Expressions in Checkpoints” on page 764

For information on defining regular expressions, including regular expression syntax, see “Defining Regular Expressions” on page 765.

Using Regular Expressions for Property Values

If you expect the value of a property to change in a predictable way during each run session, you can use regular expressions when defining or parameterizing property values, for example, in the Object Repository window, or in programmatic descriptions. For more information on programmatic descriptions, see “Using Programmatic Descriptions” on page 863.

For example, your Web site may include a form in which the user inputs data and clicks the **Send** button to submit the form. When a required field is not completed, the form is displayed again for the user to complete. When resubmitting the form, the user clicks the **Resend** button. You can define the value of the button’s **name** property as a regular expression, so that QuickTest ignores variations in the button name when clicking the button.

Using Regular Expressions in Checkpoints

When creating a standard checkpoint to verify the property values of an object, you can set the expected value of an object's property as a regular expression so that an object with a varying value can be verified.

For example, suppose you want to check that every window and dialog box in your application contains the name of your application followed by a hyphen (-) and a descriptive title. You can add a checkpoint to each dialog box object in your test to check that the first part of the title contains the name of your application followed by a hyphen.

When creating a text checkpoint to check that a varying text string is displayed on your application, you can define the text string as a regular expression.

For example, when booking a flight in the Mercury Tours sample Web site, the total cost charged to a credit card number should not be less than \$300. You define the amount as a regular expression, so that QuickTest will ignore variations in the text string as long as the value is not less than \$300.

You can apply the same principles to any checkpoint type whose dialog box contains a **Configure Value** area similar to that described in "Configuring Constant and Parameter Values" on page 756.

For example, for table checkpoints you can set cell values as regular expressions, and for XML checkpoints you can set attribute or element values as regular expressions. For more information on specific checkpoint types, see the relevant chapter for the checkpoint type.

Defining Regular Expressions

You can define a regular expression for a constant value, a Data Table parameter value, an Environment parameter value, or a property value in a programmatic description. For more information on defining property values, see “Configuring Constant and Parameter Values” on page 756.

You can define a regular expression by entering the regular expression syntax for the string in the **Value** box in the Constant Value Options dialog box or the Parameter Options dialog box. You instruct QuickTest to treat the value as a regular expression by selecting the **Regular Expression** check box.

All programmatic description property values are automatically treated as regular expressions. For more information on programmatic descriptions, see “Using Programmatic Descriptions” on page 863.

Note: You can use regular expressions only for values of type **string**.

By default, QuickTest treats all characters in a regular expression literally, except for the period (.), hyphen (-), asterisk (*), caret (^), brackets ([]), parentheses (()), dollar sign (\$), vertical line (|), plus sign (+), question mark (?), and backslash (\). When one of these special characters is preceded by a backslash (\), QuickTest treats it as a literal character.

If you enter a special character in the **Value** box of the Constant Value Options or the Parameter Options dialog box, QuickTest asks you if you want to add a backslash (\) before each special character. If you click **Yes**, a backslash (\) is added before the special character to instruct QuickTest to treat the character literally. If you click **No**, QuickTest treats the special character as a regular expression character.

This section describes some of the more common options that can be used to create regular expressions:

- Using the Backslash Character (\)
- Matching Any Single Character (.)
- Matching Any Single Character in a List ([xy])
- Matching Any Single Character Not in a List ([^xy])
- Matching Any Single Character within a Range ([x-y])
- Matching Zero or More Specific Characters (*)
- Matching One or More Specific Characters (+)
- Matching Zero or One Specific Character (?)
- Grouping Regular Expressions (())
- Matching One of Several Regular Expressions (|)
- Matching the Beginning of a Line (^)
- Matching the End of a Line (\$)
- Matching Any AlphaNumeric Character Including the Underscore (\w)
- Matching Any Non-AlphaNumeric Character (\W)
- Combining Regular Expression Operators

Note: For a complete list and explanation of supported regular expressions characters, see the Regular Expressions section in the Microsoft VBScript documentation (select **Help > QuickTest Professional Help** to open the QuickTest Professional Help. Then select **VBScript Reference > VBScript > User's Guide > Introduction to Regular Expressions**).

Using the Backslash Character

A backslash (\) can serve two purposes. It can be used in conjunction with a special character to indicate that the next character be treated as a literal character. For example, \. would be treated as period (.) instead of a wildcard. Alternatively, if the backslash (\) is used in conjunction with some characters that would otherwise be treated as literal characters, such as the letters n, t, w, or d, the combination indicates a special character. For example, \n stands for the newline character.

For example:

- w matches the character w
- \w is a special character that matches any word character including underscore
- \\ matches the literal character \
- \(matches the literal character (

For example, if you were looking for a Web site called:

newtours.demoaut.com

the period would be mistaken as an indication of a regular expression. To indicate that the period is not part of a regular expression, you would enter it as follows:

newtours\.demoaut\.com

Note: If a backslash character is used before a character that has no special meaning, the backslash is ignored. For example, \z matches z.

Matching Any Single Character

A period (.) instructs QuickTest to search for any single character (except for \n). For example:

welcome.

matches welcomes, welcomed, or welcome followed by a space or any other single character. A series of periods indicates the same number of unspecified characters.

To match any single character including \n, enter:

(.|\n)

For more information on the () regular expression characters, see “Grouping Regular Expressions” on page 770. For more information on the | regular expression character, see “Matching One of Several Regular Expressions” on page 771.

Matching Any Single Character in a List

Square brackets instruct QuickTest to search for any single character within a list of characters. For example, to search for the date 1967, 1968, or 1969, enter:

196[789]

Matching Any Single Character Not in a List

When a caret (^) is the first character inside square brackets, it instructs QuickTest to match any character in the list except for the ones specified in the string. For example:

[^ab]

matches any character except a or b.

Note: The caret has this special meaning only when it is displayed first within the brackets.

Matching Any Single Character within a Range

To match a single character within a range, you can use square brackets ([]) with the hyphen (-) character. For instance, to match any year in the 1960s, enter:

196[0-9]

A hyphen does not signify a range if it is displayed as the first or last character within brackets, or after a caret (^).

For example, [-a-z] matches a hyphen or any lowercase letter.

Note: Within brackets, the characters ".", "*", "[", and "\" are literal. For example, [.*] matches . or *. If the right bracket is the first character in the range, it is also literal.

Matching Zero or More Specific Characters

An asterisk (*) instructs QuickTest to match zero or more occurrences of the preceding character. For example:

`ca*r`

matches `car`, `caaaaaar`, and `cr`.

Matching One or More Specific Characters

A plus sign (+) instructs QuickTest to match one or more occurrences of the preceding character. For example:

`ca+r`

matches `car` and `caaaaaar`, but not `cr`.

Matching Zero or One Specific Character

A question mark (?) instructs QuickTest to match zero or one occurrences of the preceding character. For example:

`ca?r`

matches `car` and `cr`, but nothing else.

Grouping Regular Expressions

Parentheses (()) instruct QuickTest to treat the contained sequence as a unit, just as in mathematics and programming languages.

Using groups is especially useful for delimiting the argument(s) to an alternation operator (|) or a repetition operator (*, +, ?, {}).

Matching One of Several Regular Expressions

A vertical line (|) instructs QuickTest to match one of a choice of expressions. For example:

`foo|bar`

causes QuickTest to match either foo or bar.

`fo(o|b)ar`

causes QuickTest to match either fooar or fobar.

Matching the Beginning of a Line

A caret (^) instructs QuickTest to match the expression only at the start of a line, or after a newline character.

For example:

`book`

matches book within the lines—book, my book, and book list, while

`^book`

matches book only in the lines—book and book list.

Matching the End of a Line

A dollar sign (\$) instructs QuickTest to match the expression only at the end of a line, or before a newline character. For example:

`book`

matches book within the lines—my book, and book list, while a string that is followed by (\$), matches only lines ending in that string. For example:

`book$`

matches book only in the line—my book.

Matching Any AlphaNumeric Character Including the Underscore

`\w` instructs QuickTest to match any alphanumeric character and the underscore (A-Z, a-z, 0-9, `_`).

For example:

`\w*` causes QuickTest to match zero or more occurrences of the alphanumeric characters—A-Z, a-z, 0-9, and the underscore (`_`). It matches `Ab`, `r9Cj`, or `12_uYLgeu_435`.

For example:

`\w{3}` causes QuickTest to match 3 occurrences of the alphanumeric characters A-Z, a-z, 0-9, and the underscore (`_`). It matches `Ab4`, `r9_`, or `z_M`.

Matching Any Non-AlphaNumeric Character

`\W` instructs QuickTest to match any character other than alphanumeric characters and underscores.

For example:

`\W` matches `&`, `*`, `^`, `%`, `$`, and `#`

Combining Regular Expression Operators

You can combine regular expression operators in a single expression to achieve the exact search criteria you need.

For example, you can combine the `'.'` and `'*'` characters to find zero or more occurrences of any character (except `\n`).

For example,

`start.*`

matches `start`, `started`, `starting`, `starter`, and so forth.

You can use a combination of brackets and an asterisk to limit the search to a combination of non-numeric characters. For example:

`[a-zA-Z]*`

To match any number between 0 and 1200, you need to match numbers with 1 digit, 2 digits, 3 digits, or 4 digits between 1000-1200.

The regular expression below matches any number between 0 and 1200.

`([0-9]?[0-9]?[0-9]|1[01][0-9][0-9]|1200)`

28

Adding Steps Containing Programming Logic

After creating a test, you can use special QuickTest tools to enhance it with programming statements, even if you choose not to program manually in the Expert View.

This chapter includes:

- About Adding Steps Containing Programming Logic on page 776
- Inserting Steps Using the Step Generator on page 777
- Using Conditional Statements on page 797
- Using Loop Statements on page 803
- Generating With Statements for Your Test on page 806
- Generating Messages on page 812
- Adding Comments on page 815
- Synchronizing Your Test on page 816

About Adding Steps Containing Programming Logic

When you design tests, you usually begin by adding steps that represent the operations an end-user would perform as part of the business process you want to test. Then, to increase the power and flexibility of your test, you can add steps that contain programming logic to the basic framework.

Programming statements can contain:

- **Test object operations.** These are methods and properties defined by QuickTest. They can be operations that a user can perform on an object, operations that can retrieve or set information, or operations that perform operations triggered by an event.
- **Native operations.** These are methods and properties defined within the object you are testing, and therefore are retrieved from the run-time object in the application.
- VBScript programming commands that affect the way the test runs, such as conditions and loops. These are often used to control the logical flow of a test.
- Supplemental statements, such as comments, to make your test easier to read, and messages that appear in the test results, to alert you to a specified condition.

This chapter shows you how to insert different types of statements, mostly from the Keyword View, aided by the Step Generator and other dialog boxes.

The Step Generator dialog box helps you add steps that use test object operations, Utility object operations, and function calls, so that you do not need to memorize syntax or to be proficient in high-level VBScript. You can use the Step Generator from the Keyword View and also from the Expert View.

For information on how to insert statements in the Expert View, see Chapter 29, “Working in the Expert View and Function Library Windows.”

You can incorporate decision-making into your test and define messages for the test results by using the appropriate dialog boxes.

In addition, you can improve the readability of your test using **With** statements. You can instruct QuickTest to automatically generate **With** statements as you record. But even after your basic test is recorded, you can convert its statements, in the Expert View, to **With** statements—by selecting a menu command.

You can handle synchronization issues between the run session and your application, using synchronization points.

When working with tests, you can also measure how long it takes certain parts of your test to run, using transaction statements.

Inserting Steps Using the Step Generator

The Step Generator enables you to add steps by selecting from a range of context-sensitive options and entering the required values. In the Step Generator dialog box you can define steps that use:

- test object operations (tests only)
- Utility object operations
- calls to library functions (tests only), VBScript functions, and internal script functions

For example, you can add a step that checks that an object exists, or that stores the returned value of a method as an output value or as part of a conditional statement. You can parameterize any of the values in your step.

Note: You can use the Step Generator to insert steps in tests and function libraries. However, in function libraries, you cannot use the Step Generator to access test object names or collections, or to access the list of library functions.

Before you open the Step Generator to define a new step, you first select where in your test the new step should be inserted. For more information on the hierarchy of steps and objects, see “Understanding the QuickTest Object Hierarchy” on page 391.

After you open the Step Generator, you first select the category for the step operation (test object, Utility object or function) and the required object or the function library source (for example, built-in or local script functions). You can then select the appropriate operation (method, property, or function) and define the arguments and return values, parameterizing them if required.

The Step Generator then inserts a step with the correct syntax into your test. You can continue to add further steps at the same location without closing the Step Generator.

You can open the Step Generator from the Keyword View, Expert View, or Active Screen.

To open the Step Generator from the Keyword View or Expert View:

- 1 While recording or editing, click the step which you want the new step to follow. (When you finish defining the new step, QuickTest will insert it after this step.)
- 2 Select **Insert > Step Generator** or right-click the step and select **Insert Step > Step Generator**. Alternatively, press F7.

The Step Generator dialog box opens and displays the object from the selected step in the **Object** box. For more information, see “Defining Steps in the Step Generator Dialog Box” on page 780.

To open the Step Generator from a function library:

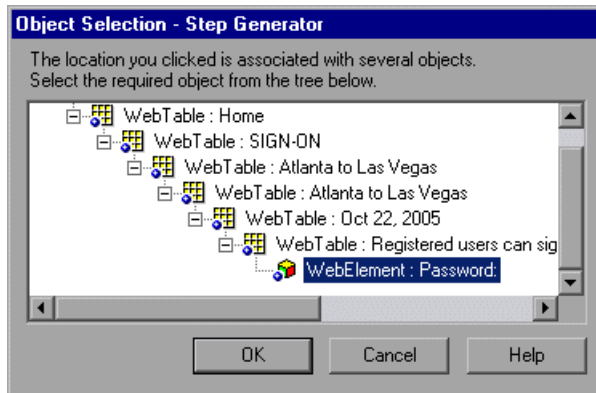
- 1 In the function library, click the location in which you want to insert the new step.
- 2 Select **Insert > Step Generator**, or right-click and select **Step Generator**. Alternatively, press F7.

The Step Generator dialog box opens. For more information, see “Defining Steps in the Step Generator Dialog Box” on page 780.

To open the Step Generator from the Active Screen while editing:

- 1** Confirm that the Active Screen is displayed. If it is not, select **View > Active Screen** or toggle the **Active Screen** toolbar button.
- 2** In the Keyword View or Expert View, click the step which you want the new step to follow. (When you finish defining the new step, QuickTest will insert it after this step.) The Active Screen displays the captured bitmap or HTML source corresponding to the selected step.
- 3** In the Active Screen, right-click the object for which you want to insert a step, and select **Step Generator**.

If the location you clicked is associated with more than one object, the **Object Selection - Step Generator** dialog box opens.



- 4** Select an object and click **OK**. The Step Generator dialog box opens and displays the object from the selected step in the **Object** box. For more information, see “Defining Steps in the Step Generator Dialog Box” on page 780.

Defining Steps in the Step Generator Dialog Box

The Step Generator dialog box enables you to add steps that perform operations, using test object methods (for tests only), Utility object methods, or function calls.

Step Generator

Category: Test Objects

Object: userName

☒ Test object operations ☐ Native operations

Operation: Set

Arguments:

Name	Type	Value
Text *	String	

* indicates a mandatory argument.

☐ Return value

Generated step:

```
Browser("Welcome: Mercury Tours").Page("Welcome: Mercury Tours").WebEdit("userName").Set Text
```

☐ 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 785.

In a function library, the Step Generator has a different title, contains only Utility objects and built-in and local script functions, and the box at the bottom of the dialog box shows a preview of the statement that will be inserted in the function library. For more information, see “Viewing the Generated Step in a Function Library” on page 785.

When the Step Generator dialog box opens, the object from the selected step is displayed in the **Object** box and the default method for the object is shown in the **Operation** box.

Defining a New Step

When you define a new step, you first select the type of step that you want to add to your test. You can then select the specific object and operation for the step, or the function that you want the step to use.

After you select the operation for the step, you can specify the relevant argument values and the location for the return value, if applicable. These values can be parameterized if required.

Finally, you can view the step documentation or statement syntax and add your new step or statement to your test or function library.

Note: Although the Step Generator shows information regarding the currently selected step, selections that you make in the Step Generator add a new step to your test; they do not modify the existing step.

Selecting the Type of Step to Add

In the **Category** list box, you can select one of the following options:


- **Test Objects.** Enables you to select a test object and operation for the step (for tests only). For more information, see “Specifying a Test Object and Operation for the Step” on page 786.
- **Utility Objects.** Enables you to select a Utility object and operation for the step. For more information, see “Specifying a Utility Object and Operation for the Step” on page 791.
- **Functions.** Enables you to select a function for the step from the available library functions (tests only), VBScript functions, and internal script functions. For more information, see “Specifying a Function for the Step” on page 793.

Specifying Argument Values

After you select the object and the operation (method, property, or function) for the step, you can specify the relevant argument values. These values can be parameterized if required.


If the selected operation has arguments, the **Arguments** area displays the name and type of each argument.

In the **Value** column, you can define the values for the arguments, as follows:

- **Mandatory arguments.** If the name of the argument is followed by a red asterisk (*), you must specify a value for the argument. You cannot insert the step or view the step documentation if the values have not been defined for all mandatory arguments.
- **Optional arguments.** If the name of the argument is not followed by a red asterisk (*), you can specify a value for the argument or leave the cell blank. If you do not specify a value, QuickTest uses the default value for the argument. (You can view the default value by moving the pointer over the cell).
- **Required arguments.** If you specify a value for an optional argument, then you must also specify the values for any optional arguments that are listed before this argument. If you do not specify these values, QuickTest uses the default values for all required arguments. You can see the default value for each argument in a tooltip, by moving the pointer over the **Value** column.
- **Parameterized arguments.** You can use a parameter for any argument value by clicking the parameterization button . For more information, see “Configuring a Selected Value” on page 760.
- **Predefined constants.** If an argument has a predefined list of values, QuickTest provides a drop-down list of possible values. If a list of values is provided, you cannot manually type a value in this box.

Specifying the Location for the Return Value

If the selected operation returns a value, you can specify that you want to store the value by selecting the **Return Value** check box. When this check box is selected, a default variable is displayed as the return value location.

You can supply a different variable definition by editing the value. You can select a different storage location for the return value by clicking the displayed value and then the output storage button . For more information, see “Storing Return Values and Action Output Parameter Values” on page 794.

Viewing the Step Documentation in the Keyword View

If you open the Step Generator from the Keyword View, the **Step documentation** box at the bottom of the Step Generator dialog box can display summary information on the current step in an easy-to-read sentence.

If you select either the **Test Object** or **Utility Object** category and you define all the mandatory and required values for the current operation, the **Step documentation** box describes the operation performed by the step. When the step is inserted into your test, this description is displayed in the **Documentation** column in the Keyword View.

If all the mandatory and required argument values have not been defined for the operation, the **Step documentation** box displays a warning message.

Note: If you select the **Functions** category, step documentation is available for user-defined functions, if you provided this information when defining them. For more information, see “Documenting the Function” on page 934.

Viewing the Generated Step in the Expert View

If you open the Step Generator from the Expert View, the **Generated step** box displays the defined statement for the step.

If all the mandatory and required argument values have not been defined for the operation, the names of the undefined arguments are highlighted in bold text. If you attempt to insert the step, an error message is displayed.

Viewing the Generated Step in a Function Library

If you open the Step Generator from a function library, the **Generated step** box displays the defined statement for the step.

If all the mandatory and required argument values have not been defined for the statement, the names of the undefined arguments are highlighted in bold text. If you attempt to insert the step, an error message is displayed.

Inserting Steps

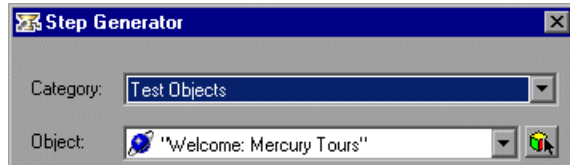
After you define all mandatory argument values for the current operation, the following options are available:

- To insert the current step and close the Step Generator, make sure the **Insert another step** check box is cleared. When you click **OK**, the step is added to your test and the Step Generator dialog box closes.
- To insert the current step and continue adding steps at the same location, select the **Insert another step** check box. The **OK** button changes to **Insert**. When you click **Insert**, the current step is added to your test and the Step Generator dialog box remains open, enabling you to define another step.

When you insert a new step using the Step Generator, it is added to your test after the selected step, and the new step is selected. For more information on the hierarchy of steps and objects and the positioning of new steps, see “Understanding the QuickTest Object Hierarchy” on page 391.

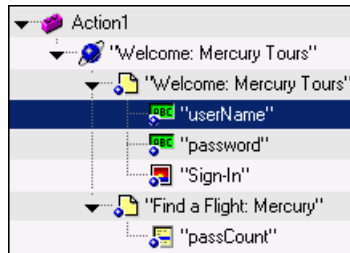
Specifying a Test Object and Operation for the Step

If you select **Test Objects** in the **Category** list in the Step Generator dialog box, you can select the object for the new step in the context of the currently selected step in your test. Alternatively, you can select any object from the object repository or from your application.

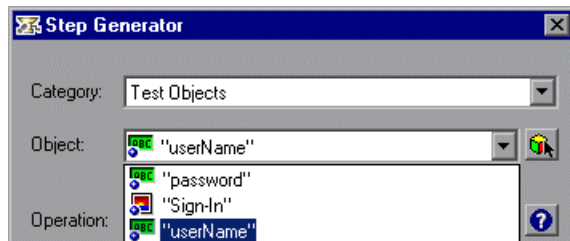


The list in the **Object** box contains all the objects in the object repository that are at the same hierarchical level and location as the currently selected step. You can select any of these objects for your new step.

For example, suppose that you selected the step for the **userName** object in the **Welcome: Mercury Tours** Web page, as shown below:



When you open the Step Generator, **Test Objects** is selected in the **Category** box, and the **Object** box lists the **userName**, **password** and **Sign-in** objects.



Note: The objects are listed by name in alphabetical order.



You can select an object from the object repository or from your application, by clicking the **Select Object** button. For more information, see “Selecting an Object from the Repository or Application” on page 788.

After you select the object for the step, you can select the required operation type (test object operation or, if available, native (run-time object) operation) and then you can select the operation for the step.

Selecting the Operation for a Test Object

If QuickTest can retrieve native (run-time object) operations for the selected test object, you can select the operation type—**Test object operation** or **Native operation**. (If QuickTest cannot retrieve native operations for the selected object, the **Native operations** option is not available.)

The **Operation** box displays the default operation for the selected object. You can select a different operation from the **Operation** box list, which contains all the operations that are available for the selected object.



For detailed information on a test object operation and its syntax, you can click the **Operation Help** button to open the *HP QuickTest Professional Object Model Reference* for the selected operation.

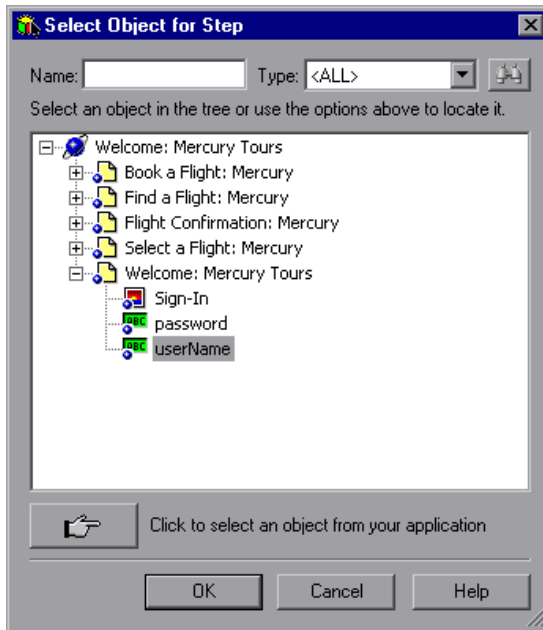
If you click the **Operation Help** button when a native operation is selected, the *HP QuickTest Professional Object Model Reference* opens for the selected test object. For more information on specific native operations, see the documentation for the environment or application you are testing.

Note: If you select a native operation, the Step Generator inserts a step using **.Object** syntax. For information on using the **Object** property, see “Accessing Native Properties and Operations” on page 887.

After you select the operation for your test object, you can define the relevant argument values. For more information, see “Specifying Argument Values” on page 783.

Selecting an Object from the Repository or Application

The Select Object for Step dialog box displays the object repository tree and enables you to select an object from the object repository or from your application.



You can select any object in the object repository tree for your new step. For more information on the object repository, see Chapter 5, “Managing Test Objects in Object Repositories.”

If the object that you want to use in the new step is not in the object repository, you can select an object in your application.

When you click **OK**, the selected object is displayed in the dialog box from which you opened the Select Object for Step dialog box.

To select an object in your application for the new step:

- 1** Click the pointing hand button. QuickTest is hidden, and the pointer changes to a pointing hand.
- 2** Use the pointing hand to click on the required object in your application. For more information about using the pointing hand feature, see “Tips for Using the Pointing Hand” on page 790.

If the location you clicked is associated with more than one object, the Object Selection dialog box opens.



- 3** Select the object for the new step and click **OK**. The object is displayed in the dialog box from which you opened the Select Object for Step dialog box.

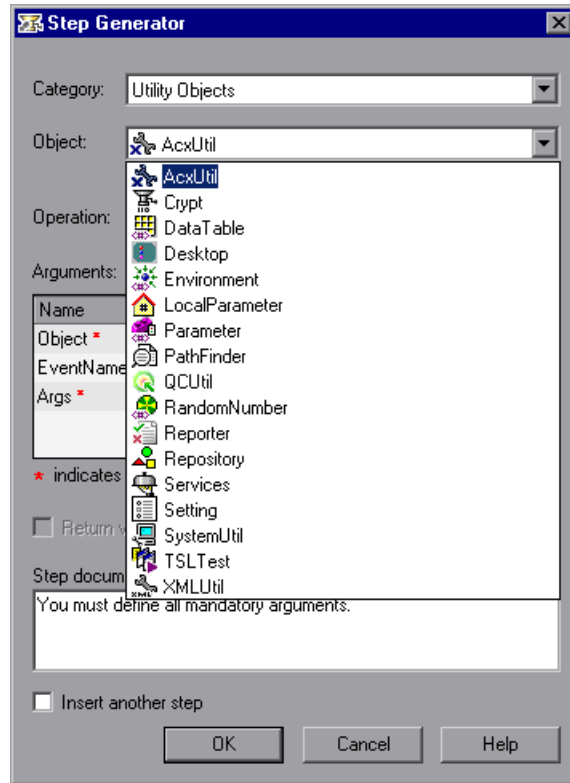
Tip: If you select an object in your application that is not in the object repository, a test object is added to the object repository when you insert the new step.

Tips for Using the Pointing Hand

- You can hold the left CTRL key to change the pointing hand to a standard pointer. You can then change the window focus or perform operations in QuickTest or in your application, such as right-clicking, using the scroll bars, or moving the pointer over an object to display a context menu.
- If the window containing the object you want to select is partially hidden by another window, hold the pointing hand over the partially hidden window for a few seconds until it comes to the foreground. Then point to and click the required object. You can configure the length of time required to bring a window into the foreground using the General pane of the Options dialog box.
- If the window containing the object you want to select is fully hidden by another window, or if a dialog box is hidden behind a window, press the left CTRL key and arrange the windows as needed.
- If the window containing the object you want to select is minimized, you can display it by holding the left CTRL key, right-clicking the application in the Windows task bar, and choosing **Restore** from the context menu.
- If the object you want to select can be displayed only by performing an event (such as right-clicking or moving the pointer over an object to display a context menu), hold the left CTRL key. The pointing hand temporarily turns into a standard pointer and you can perform the event. When the object you want to select is displayed, release the left CTRL key. The pointer becomes a pointing hand again.

Specifying a Utility Object and Operation for the Step

If you select **Utility Objects** in the **Category** box list, you can select the required Utility (reserved) object from the **Object** box list.



Tip: The above example shows the list of Utility objects that are available when you open the Step Generator from the Keyword View. When you open the Step Generator from the Expert View or a function library, the list includes a number of additional Utility objects. If you have one or more add-ins installed, the list may include additional Utility objects for those add-ins.

For more information on Utility objects, see the Utility Objects section of the *HP QuickTest Professional Object Model Reference*.

The **Operation** box displays the default operation for the selected Utility object. You can select a different operation from the **Operation** box list, which contains all the operations that are available for the selected object.

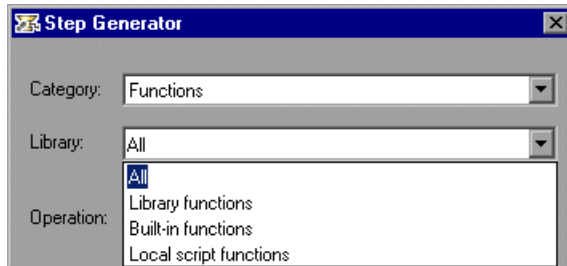


For detailed information on a Utility object operation and its syntax, you can click the **Operation Help** button to open the *HP QuickTest Professional Object Model Reference* for the selected operation.

After you select the operation for your Utility object, you can define the relevant argument values. For more information, see “Specifying Argument Values” on page 783.

Specifying a Function for the Step

If you select **Functions** in the **Category** box list, you can select one of the following options from the **Library** box list:



- **All.** Enables you to select a function from all the available functions and types.
- **Library functions.** Enables you to select a function from any function library associated with your test (for tests only). For more information on defining and using associated function libraries, see “Working with Associated Function Libraries” on page 919.
- **Built-in functions.** Enables you to select any standard VBScript function supported by QuickTest. For more information on working with VBScript, you can open the VBScript documentation from the QuickTest **Help** menu (**Help > QuickTest Professional Help > VBScript Reference**).
- **Local script functions.** Enables you to select any local function defined directly in the current action or function library.


You can select the required function from the **Operation** box list, which displays the functions available for the selected function type in alphabetical order.




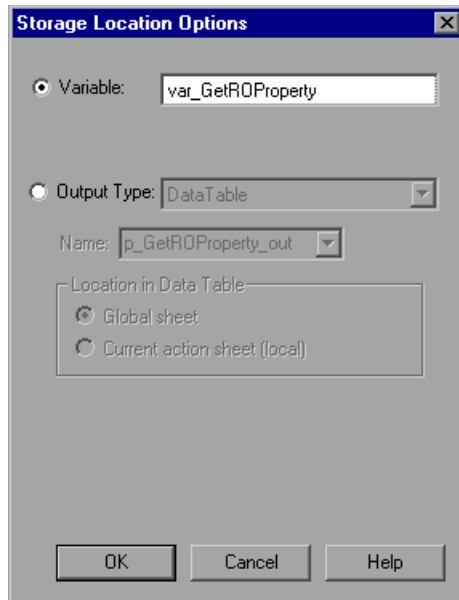
For detailed information on a selected built-in VBScript function, you can click the **Operation Help** button to open Microsoft's VBScript Reference or the *HP QuickTest Professional Object Model Reference*. This option is not available for library and local script functions.

After you select the function for the operation, you can define the relevant argument values. For more information, see “Specifying Argument Values” on page 783.

Storing Return Values and Action Output Parameter Values

The Storage Location Options dialog box enables you to specify how and where to store a return value for an operation that you have selected in the Step Generator dialog box. When you click the displayed return value and then the output storage button , the Storage Location Options dialog box opens.

The Storage Location Options dialog box also enables you to specify how and where to store the value for an output parameter for an action. When you select an output parameter in the Parameter Values tab of the Action Call Properties dialog box and click the output storage button  in the **Store in** column, the Storage Location Options dialog box opens.



The image shows the "Storage Location Options" dialog box. It has a title bar with a close button. Inside, there are two radio buttons: "Variable:" (selected) and "Output Type:". The "Variable:" option has a text field containing "var_GetROProperty". The "Output Type:" option has a dropdown menu showing "DataTable". Below this is a "Name:" label and a dropdown menu showing "p_GetROProperty_out". Underneath is a section titled "Location in Data Table" with two radio buttons: "Global sheet" (selected) and "Current action sheet ([local])". At the bottom are three buttons: "OK", "Cancel", and "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 (if any) or enter a different variable name.
- **Output Type.** Stores the value in an test or action output parameter, Data Table column or environment variable. You can specify the output type and settings as for any other output value.

When a return value or a test or action output parameter is first selected, the default output definition for the value is displayed. For more information on default output definitions for a return value, see “Understanding Default Output Definitions” on page 683.

For more information on default output definitions for output action parameter values, see “Understanding Default Output Definitions for Action Parameter Values” on page 796.

You can accept the default output definition by clicking **OK** or you can change the output type and/or settings.

The options for changing the output type and settings are identical to those in the Output Options dialog box. For more information, see:

- “Outputting a Value to an Action Parameter” on page 684
- “Outputting a Value to the Data Table” on page 685
- “Outputting a Value to an Environment Variable” on page 686

Understanding Default Output Definitions for Action Parameter Values

When you select **Output Type** for an output action parameter value for a nested action:

- If at least one output action parameter is defined in the action calling the nested action, the default output type is **Test/action parameter** and the default output name is the first output parameter displayed in the Action Properties dialog box of the calling action.
- If no output action parameters are defined in the calling action, the default output type is Data Table and QuickTest creates a new Data Table output name based on the selected value. The value is created in the Global sheet of the Data Table.

When you select **Output Type** for an output action parameter value for a top-level action:

- If at least one output action parameter is defined in the test, the default output type is **Test/action parameter** and the default output name is the first output parameter displayed in the Test Properties dialog box.
- If no output action parameters are defined in the test, the default output type is **Data Table** and QuickTest creates a new Data Table output name based on the selected value. The value is created in the Global sheet of the Data Table.

Using Conditional Statements

You can control the flow of your test with conditional statements. Using conditional **If...Then...Else** statements, you can incorporate decision-making into your tests.

The **If...Then...Else** statement is used to evaluate whether a condition is true or false and, depending on the result, to specify one or more statements to run. Usually the condition is an expression that uses a comparison operator to compare one value or variable with another. The following comparison operators are available: less than <, less than or equal to <=, greater than >, greater than or equal to >=, not equal <>, and equal =.

Your **If...Then...Else** statement can be nested to as many levels as you need. It has the following syntax:

If *condition* **Then** *statements* [**Else** *elasticsearch*] **End If**

Or, you can use the block form syntax:

```
If condition Then
    [statements]
[Elseif condition-n Then
    [elseifstatements] . . .
[Else
    [elasticsearch]
End If
```

For example:

'Set the focus on (activate) the Open Order dialog box
Window("Flight Reservation").Dialog("Open Order").Activate

'Insert a check mark in the Order No. check box
Window("Flight Reservation").Dialog("Open Order").WinCheckBox("Order No.").
Set "ON"

*Insert an order number in the displayed box and save the value as "OrderNo" for
'use later in the script. If the value is less than or equal to 0, generate a message
'box. (If the value is illegal and a message box is generated, end the run session
'when the user clicks OK.)*
OrderNo = InputBox("Enter Order Number")

If OrderNo <= 0 Then
 Msgbox "You entered an invalid order number."
 ExitAction
End If

'Insert the saved order number value in the Order No. box
Window("Flight Reservation").Dialog("Open Order").WinEdit("OrderNumber
Edit").Set OrderNo

'Click OK to close the Open Order dialog box
Window("Flight Reservation").Dialog("Open Order").WinButton("OK").Click

'Check if an error message opens and send a report to the test results
If Window("Flight Reservation").Dialog("Open Order").Dialog("Flight
Reservations").Exist Then
 Reporter.ReportEvent micFail, "Check that the value of the order
 number is legal", "The order number does not exist."
 Window("Flight Reservation").Dialog("Open Order").Dialog("Flight
 Reservations").WinButton("OK").Click
Else
 Reporter.ReportEvent micPass, "Check that the value of the order
 number is legal", "The order number exists."
End If

The above example checks whether the application being tested will identify whether a valid order number is being entered in the Order No. box in the Open Order dialog box.

To do this, QuickTest activates the Open Order dialog box (brings it into focus), selects the Order No. check box, and opens a box in which the user inserts a value—the relevant order number—and clicks **OK**. The first conditional statement instructs QuickTest to verify that the value entered by the user is greater than zero. **If** it is not greater than zero, QuickTest opens a message box stating that the value entered is invalid. When the user clicks **OK** to close the message box, QuickTest ends the run session.

Otherwise, if the value entered is greater than zero, QuickTest inserts the above value in the Order No. box.

The next **If** statement instructs QuickTest to check whether the order number exists in the application and to send a report to the Test Results indicating that the step passed or failed. If the step failed because the order number was invalid, the Flight Reservations error message opens, and QuickTest clicks **OK** to close this message box before ending the run session.

Note: You can insert conditional statements in the Expert View and in the Keyword View. You can also switch between the views, as needed. For information on working with conditional steps in the Expert View, see “Using Comments, Control-Flow, and Other VBScript Statements” on page 876, and the VBScript documentation (select **Help > QuickTest Professional Help > VBScript Reference**).

To insert a conditional statement in the Keyword View:

- 1 In the Keyword View, select the step that you want the conditional statement to follow.




The following example shows the userName row selected:

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 Select **Insert > Conditional Statement** and select **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 "43c1028da7b831801
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** property, it is not necessary to add a value to the **Value** cell.)
- 6 Insert a **Then** statement by selecting the **If** statement step and inserting a new statement (**Insert > New Step**) or by recording a new step. For example:

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 "43c1028da7b831801c87
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 select one of the following options:

To add:	Select:
an If statement	Insert > Conditional Statement > If...Then
an Elseif statement	Insert > Conditional Statement > Elseif...Then
an Else statement	Insert > Conditional Statement > Else

For example, the statements below check that the User Name edit box exists in the Mercury Tours site. **If** the edit box exists, **Then** a user name is entered; **Else** a message is sent to the Test Results.

Welcome: Mercury Tours			
Welcome: Mercury Tours			
If userName	Exist		Check whether the "userName" edit box exists. If :
userName	Set	DataTable("p_Us...	Enter <the value of the 'p_UserName' Data Tab
Else Statement			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 409.

Using Loop Statements

You can control the flow of your test with loop statements. Using loop statements, you can run a group of steps repeatedly, either while or until a condition is True. You can also use loop statements to repeat a group of steps a specific number of times.

The following loop statements are available in the Keyword View:





- **While...Wend.** Performs a series of statements as long as a specified condition is True.
- **For...Next.** Uses a counter to perform a group of statements a specified number of times.
- **Do...While.** Performs a series of statements indefinitely, as long as a specified condition is True.
- **Do...Until.** Performs a series of statements indefinitely, until a specified condition becomes True.

Note: For more information on loop statements, see the VBScript documentation (select **Help** > **QuickTest Professional Help** > **VBScript Reference**).

To insert a loop statement in the Keyword View:

- 1** Select the step that you want the loop statement to follow.
- 2** Select **Insert > Loop Statement** and select the statement type that you want to insert from the sub-menus. The new statement is added to the Keyword View below the selected step.

Each statement type is indicated by one of the following icons:

Icon	Type
	While...Wend statement
	For...Next statement
	Do...While statement
	Do...Until statement

- 3** In the **Value** column, enter the required condition, for example:
For i = 0 to ItemsCount - 1
- 4** To complete the loop statement, you can:
 - Select the loop statement step and record a new step to add it to your loop statement.
 - Select the loop statement step and select **Insert > New Step** or press F8 to insert a new step into your loop statement.

Note: For more information on working in the Expert View, see Chapter 29, “Working in the Expert View and Function Library Windows.”

The following example counts the number of items in a list and then selects them one by one. After each of the items has been selected, the test continues.

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 409.

Generating With Statements for Your Test

You can instruct QuickTest to automatically generate **With** statements when you record a test or to generate **With** statements for any existing action. You can also remove **With** statements from an action.

Note: Using **With** statements in your test has no effect on the run session itself, only on the way your test appears in the Expert View. Generating **With** statements for your test does not affect the Keyword View in any way.

Understanding With Statements

With statements make your script (in the Expert View) more concise and easier to read by grouping consecutive statements with the same parent hierarchy.

The **With** statement has the following syntax.

```
With object  
    statement  
    statement  
    statement  
End With
```

For example, you could replace this script:

```
Window("Flight Reservation").WinComboBox("Fly From:").Select "London"
Window("Flight Reservation").WinComboBox("Fly To:").Select "Los Angeles"
Window("Flight Reservation").WinButton("FLIGHT").Click
Window("Flight Reservation").Dialog("Flights Table").WinList("From").Select
"19097 LON "
Window("Flight Reservation").Dialog("Flights Table").WinButton("OK").Click
```

with the following:

```
With Window("Flight Reservation")
    .WinComboBox("Fly From:").Select "London"
    .WinComboBox("Fly To:").Select "Los Angeles"
    .WinButton("FLIGHT").Click
    With .Dialog("Flights Table")
        .WinList("From").Select "19097 LON "
        .WinButton("OK").Click
    End With 'Dialog("Flights Table")
End With 'Window("Flight Reservation")
```

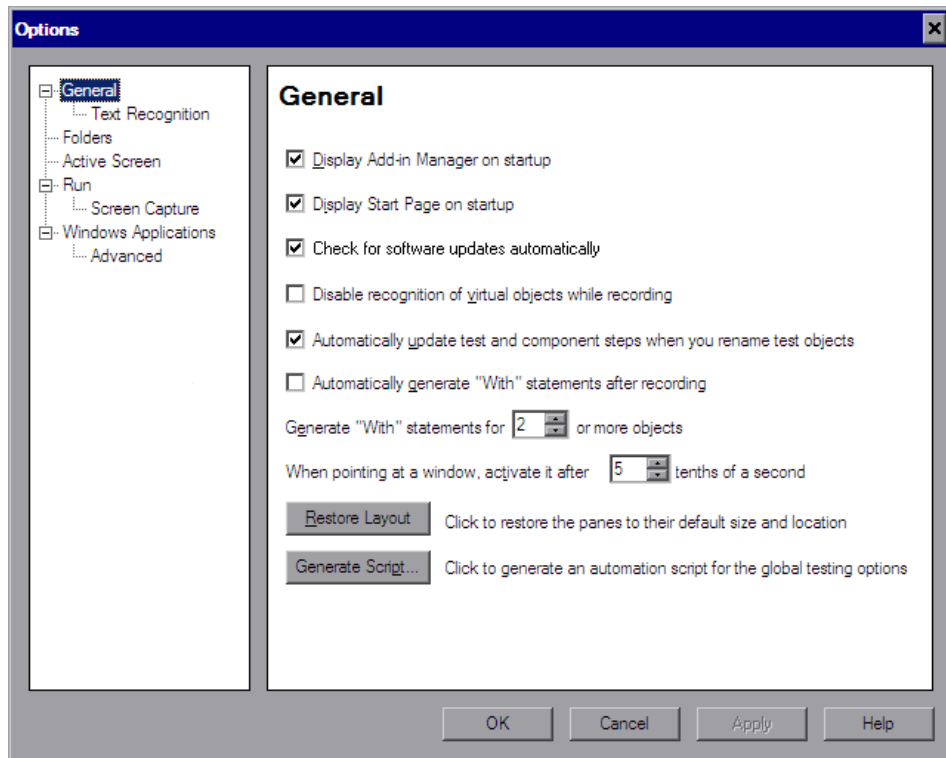
Automatically Generating With Statements

You can instruct QuickTest to automatically generate **With** statements for the steps you record. When you select this option, statements are displayed in their normal format while recording. When you stop recording, the statements in all actions recorded during the current recording session are automatically converted to the **With** format.

To generate With statements automatically when you record:



- 1 Select **Tools > Options** or click the **Options** toolbar button. The Options dialog box opens.



- 2 In the **General** pane, select **Automatically generate "With" statements after recording**.

- 3 Enter the minimum number of consecutive, identical objects for which you want to apply the **With** statement in the **Generate "With" statements for __ or more objects** box. The default is 2.

Note: This setting is used when you use the **Apply "With" to Script** option (see “Generating With Statements for Existing Actions” on page 809) as well as for the **Automatically generate "With" statements after recording** option.

For example, if you only want to generate a **With** statement when you have three or more consecutive statements based on the same object, enter 3.

- 4 Begin recording your test. While recording, statements are recorded normally. When you stop recording, the statements in all actions recorded during the current recording session are automatically converted to the **With** format.

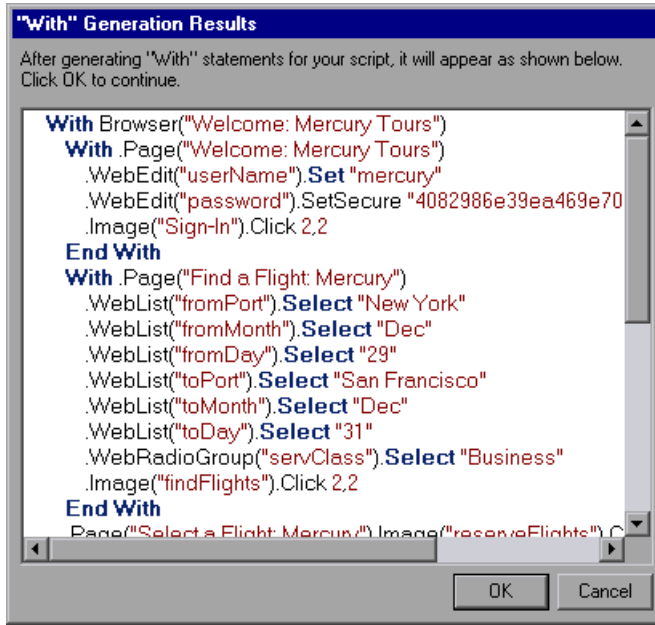
Generating With Statements for Existing Actions

You can instruct QuickTest to generate **With** statements for any action displayed in the Expert View, and to enable IntelliSense within existing **With** statements.

To generate With statements for existing actions:

- 1 Confirm that the proper number is set for the **Generate "With" statements for __ or more objects** in the General pane of the Options dialog box. (The default is 2.)
- 2 Display the action for which you want to generate **With** statements.

- 3 From the Expert View, select **Edit > Advanced > Apply "With" to Script**. The "With" Generation Results window opens.



Each **With** statement contains only one object

- 4 To confirm the generated results, click **OK**. The **With** statements are applied to the action.

Tips:

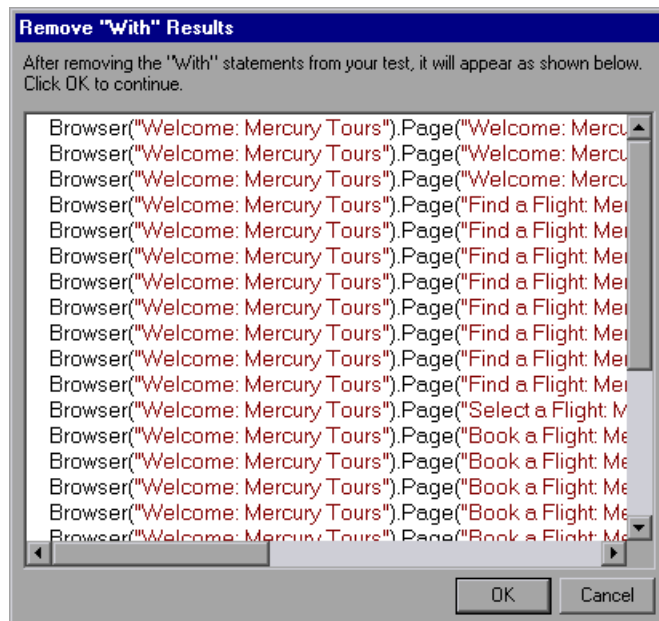
- You can search for text strings in the Generation Results window by pressing CTRL+F. For more information on the Find dialog box, see "Finding Text Strings" on page 847.
 - If you type a **With** statement (as opposed to creating it using the procedure described above), select **Edit > Advanced > Apply "With" to Script** to enable IntelliSense within the **With** statement.
-

Removing With Statements from an Action

You can remove all the **With** statements in an action displayed in the Expert View.

To remove With statements from an action:

- 1 Display the action for which you want to remove **With** statements.
- 2 From the Expert View, select **Edit > Advanced > Remove "With" Statements**. The Remove "With" Results window opens.



- 3** To confirm the results, click **OK**. The **With** statements are replaced with the standard statement format.

Generating Messages

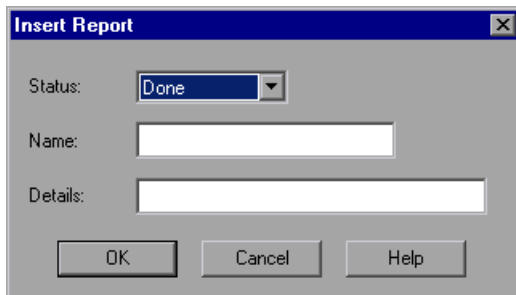
You can generate messages in your test that are displayed in the Test Results window. You can also choose to display messages on screen while running your test.

Sending Messages to the Test Results

You can define a message that QuickTest sends to your test results. For example, suppose you want to check that a password edit box exists in the Mercury Tours site. If the edit box exists, then a password is entered. Otherwise, QuickTest sends a message to the test results indicating that the object is absent.


To send a message to your test results:

- 1 In the Keyword View, select a step and select **Insert > Report** or right-click a step and select **Insert Step > Report**. The Insert Report dialog box opens.



- 2 Select the status that will result from this step from the **Status** list.

Status	Description
Passed	Causes this step to pass. Sends the specified message to the report.
Failed	Causes this step (and therefore the test itself) to fail. Sends the specified message to the report.
Done	Sends a message to the report without affecting the pass/fail status of the step.
Warning	Sends a warning status for the step, but does not cause the test to stop running, and does not affect its pass/fail status.

- 3 In the **Name** box, type a name for the step, for example, Password edit box.
- 4 In the **Details** box, type a detailed description of this step to send to your test results, for example, Password edit box does not exist.
- 5 Click **OK**. A report step is inserted into the Keyword View  and a **Reporter.ReportEvent** statement is inserted into your script in the Expert View. For example:

```
Reporter.ReportEvent micFail, "Password edit box", "Password edit box does not exist"
```

In this example, micFail indicates the status of the report (failed), Password edit box is the report name, and Password edit box does not exist is the report message.

For more information on test results, see Chapter 33, “Viewing Run Session Results.”

Note: After you add a report step, you can modify it in the Keyword View by right-clicking the step and choosing **Report Properties**, or by modifying any of the arguments in the **Value** column. (You can also modify the **Reporter.ReportEvent** statement directly in the Expert View.)

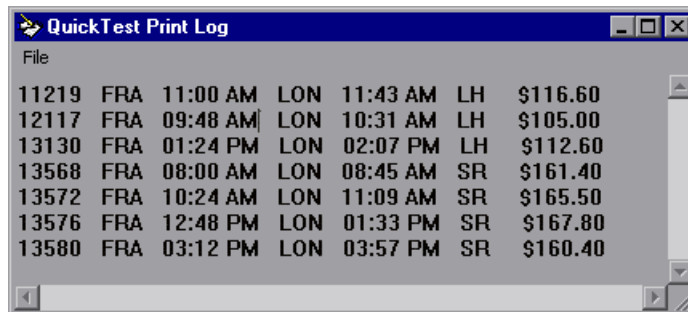
Displaying Messages During the Run Session

In addition to sending messages to the Test Results, you can generate messages in the following ways:

- Use the **MessageBox** VBScript function in your test to display information during the run session. The run session pauses until the message box is closed. For more information, see the VBScript documentation from the QuickTest **Help** menu (**Help > QuickTest Professional Help > VBScript Reference**).
- Use the **Print** Utility statement in your test to display information in the QuickTest Print Log window while still continuing the run session. For example, the following example iterates all the items in the **Flight Table** dialog (in the sample Flight application) and uses the **Print** Utility statement to print the content of each item to the QuickTest Print Log window.

```
Set FlightsList = Window("Flight Reservation").Dialog("Flights Table").
    WinList("From")
For i = 1 to FlightsList.GetItemsCount
    Print FlightsList.GetItem(i - 1)
Next
```

The QuickTest Print Log window remains open throughout the run session, until you close it.

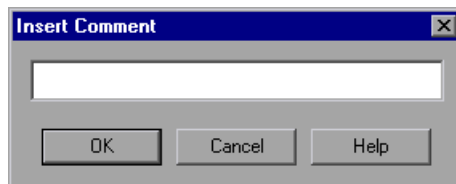


Adding Comments

While editing your test, you can add comments in the Keyword View or in the Expert View. You can also add comments to function libraries. A comment is an explanatory remark in a program. When you run a test, QuickTest does not process comments. You can use comments to explain sections of your tests to improve readability and to make them easier to update. You can add comments directly to the Keyword View or the Expert View, or you can use the Insert Comment dialog box. You can also modify comments at any time directly in the Keyword View or the Expert View, or using the Comment Properties dialog box.

To add a comment in the Keyword View:

- 1 If the **Comment** column is not visible, right-click any column header and select **Comment**.
- 2 Add a comment in one of the following ways:
 - To add a comment on the same line as a step, select the step and type your comment in the **Comment** column.
 - To add a comment on a separate line, select a step and select **Insert > Comment**, or right-click a step and select **Insert Step > Comment**. The Insert Comment dialog box opens. Type a comment and click **OK**. A comment statement is added to your test.



In the Keyword View, the  icon indicates a comment.

To add a comment in the Expert View or a function library:

Type an apostrophe (') and then type your comment. You can add a comment at the end of a line or at the beginning of a separate line.

To modify a comment:

- In the Keyword View, you can modify the comment text directly in the **Comment** column, or you can right-click any column in the step and select **Comment Properties** to open the Comment Properties dialog box (which is similar to the Insert Comment dialog box).
- In the Expert View, you can overwrite any existing comment.

Tip: If you want to add the same comment to every action that you create, you can add the comment to an action template. For more information, see “Creating an Action Template” on page 462.

Synchronizing Your Test

When you run a test, your application may not always respond with the same speed. For example, it might take a few seconds:

- for a progress bar to reach 100%
- for a status message to appear
- for a button to become enabled
- for a window or pop-up message to open

You can handle these anticipated timing problems by synchronizing your test to ensure that QuickTest waits until your application is ready before performing a certain step.

There are several options that you can use to synchronize your test:

- You can insert a **synchronization point**, which instructs QuickTest to pause the test until an object property achieves the value you specify. When you insert a synchronization point into your test, QuickTest generates a **WaitProperty** statement in the Expert View.

- You can insert **Exist** or **Wait** statements that instruct QuickTest to wait until an object exists or to wait a specified amount of time before continuing the test.
- You can modify the default amount of time that QuickTest waits for a Web page to load.
- When working with tests, you can increase the default timeout settings for a test to instruct QuickTest to allow more time for objects to appear.

Creating Synchronization Points

If you do not want QuickTest to perform a step or checkpoint until an object in your application achieves a certain status, you should insert a synchronization point to instruct QuickTest to pause the test until the object property achieves the value you specify (or until a specified timeout is exceeded).

For example, suppose you record a test on a flight reservation application. You insert an order, and then you want to modify the order. When you click the **Insert Order** button, a progress bar is displayed and all other buttons are disabled until the progress bar reaches 100%. Once the progress bar reaches 100%, you record a click on the **Update Order** button.

Without a synchronization point, QuickTest may try to click the **Update Order** button too soon during a test run (if the progress bar takes longer than the test's object synchronization timeout), and the test will fail.

You can insert a synchronization point that instructs QuickTest to wait until the **Update Order** button's **enabled** property equals 1.

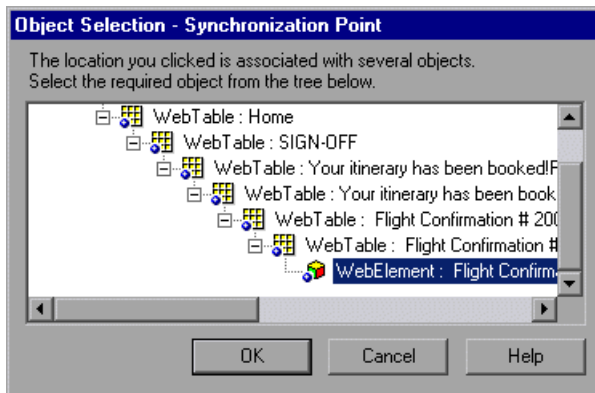
Tip: QuickTest must be able to identify the specified object to perform a synchronization point. To instruct QuickTest to wait for an object to open or appear, use an **Exist** or **Wait** statement. For more information, see “Adding Exist and Wait Statements” on page 821.

To insert a synchronization point:

- 1** Start a recording session.
- 2** Display the screen or page in your application that contains the object for which you want to insert a synchronization point.
- 3** In QuickTest, select **Insert > Synchronization Point**. The pointer changes to a pointing hand. For more information about using the pointing hand feature, see “Tips for Using the Pointing Hand” on page 820.
- 4** Click the object in your application for which you want to insert a synchronization point.

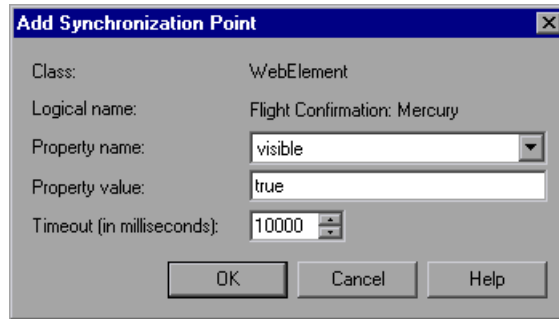
Note: It does not matter what property values the object has at the time that you insert the synchronization point.

If the location you click is associated with more than one object in your application, the Object Selection - Synchronization Point dialog box opens.



Select the object for which you want to insert a synchronization point, and click **OK**.

The Add Synchronization Point dialog box opens.



- 5 The **Property name** list contains the identification properties associated with the object. Select the property name you want to use for the synchronization point.
- 6 Enter the property value for which QuickTest should wait before continuing to the next step in the test.
- 7 Enter the synchronization point timeout (in milliseconds) after which QuickTest should continue to the next step, even if the specified property value was not achieved.
- 8 Click **OK**. A **WaitProperty** step is added to your test.

Because the WaitProperty step is a method of the selected object, it is displayed in the Keyword View with the icon for the selected object. For example, if you insert a synchronization point for the **Update Order** button, it may look something like this:

▼ Flight Confirmation: Mercury	Sync		Wait for the Web page to synchronize befo
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, see the *HP QuickTest Professional Object Model Reference*.

Tips for Using the Pointing Hand

- You can hold the left CTRL key to change the pointing hand to a standard pointer. You can then change the window focus or perform operations in QuickTest or in your application, such as right-clicking, using the scroll bars, or moving the pointer over an object to display a context menu.
- If the window containing the object you want to select is partially hidden by another window, hold the pointing hand over the partially hidden window for a few seconds until it comes to the foreground. Then point to and click the required object. You can configure the length of time required to bring a window into the foreground using the General pane of the Options dialog box.
- If the window containing the object you want to select is fully hidden by another window, or if a dialog box is hidden behind a window, press the left CTRL key and arrange the windows as needed.
- If the window containing the object you want to select is minimized, you can display it by holding the left CTRL key, right-clicking the application in the Windows task bar, and choosing **Restore** from the context menu.
- If the object you want to select can be displayed only by performing an event (such as right-clicking or moving the pointer over an object to display a context menu), hold the left CTRL key. The pointing hand temporarily turns into a standard pointer and you can perform the event. When the object you want to select is displayed, release the left CTRL key. The pointer becomes a pointing hand again.

Adding Exist and Wait Statements

You can enter **Exist** and/or **Wait** statements to instruct QuickTest to wait for a window to open or an object to appear. Exist statements return a boolean value indicating whether or not an object currently exists. Wait statements instruct QuickTest to wait a specified amount of time before proceeding to the next step. You can combine these statements within a loop to instruct QuickTest to wait until the object exists before continuing with the test.

For example, the following statements instruct QuickTest to wait up to 20 seconds for the Flights Table dialog box to open.

```
blnDone=Window("Flight Reservation").Dialog("Flights Table").Exist
counter=1
While Not blnDone
    Wait (2)
    blnDone=Window("Flight Reservation").Dialog("Flights Table").Exist
    counter=counter+1
    If counter=10 then
        blnDone=True
    End if
Wend
```

For more information on **While**, **Exist**, and **Wait** statements, see the *HP QuickTest Professional Object Model Reference*.

Modifying Timeout Values

If you find that, in general, the amount of time QuickTest waits for objects to appear or for a browser to navigate to a specified page is insufficient, you can increase the default object synchronization timeout values for your test and the browser navigation timeout values for your test.

Alternatively, if you insert synchronization points and **Exist** and/or **Wait** statements for the specific areas in your test where you want QuickTest to wait a longer time for an event to occur, you may want to decrease the default timeouts for the rest of your test.

- When working with tests, to modify the maximum amount of time that QuickTest waits for an object to appear, change the **Object Synchronization Timeout** in the **File > Settings > Run** pane. For more information, see “Defining Run Settings for Your Test” on page 1270.
- To modify the amount of time that QuickTest waits for a Web page to load, change the **Browser Navigation Timeout** in the **File > Settings > Web** pane. For more information, see the *HP QuickTest Professional Add-ins Guide*.

Index

This index contains entries from both volumes of the *HP QuickTest Professional User Guide*.

A

About QuickTest Professional window 101

absolute path 342

access permissions

- required for Quality Center 44

- required to run QuickTest 44

action calls

- iterations 508

- missing 1207

- parameter values 509

- properties 507

- run properties 508

action data sheets 455, 1228

Action List 461

action parameters 486, 502, 654, 663

- guidelines 505

- setting options 664

- storing output values 701, 712

Action tab, Data Table 455

Action toolbar, Keyword View 74, 461

action values, sharing

- using Dictionary objects 513

- using environment variables 513

- via the global Data Table 512

ActionIteration, environment variable

- 678

actions 451, 489

- adding to Keyword View 418

- calling using basic syntax 514

- creating 462

- deleting 486

- diagram 452, 490, 491

- external 454

- guidelines for working with 465

inserting

- call to 494

- copy of 492

- existing 490

mapping calls to missing 1212

missing calls to 1211

multiple, in tests 453

nesting 479, 502

non-reusable 454

overview 452, 490

parameterization data, location 477

parameters. *See* action parameters

properties 459

removing 486

removing calls to missing 1215

renaming 483

reusable 454

running from a step 984

setting parameters 471

setting properties 467

sharing values 512

- using Dictionary objects 513

- using environment variables 513

- via the global Data Table 512

splitting 481

syntax 514

syntax for parameters 515

syntax for storing return values 516

template 488

test flow 461

Test Flow pane 457

values. *See* action values, sharing

Active Screen 402

- defining capture settings 1272

- increasing/decreasing saved

- information 1590

- saving and deleting files 350, 352

- updating 406

Active Screen capture settings 1272

Index

- Active Server Page technology 1585
 - Add Existing Checkpoint dialog box 524
 - Add Existing Output Value dialog box 765
 - Add Object to Object Repository dialog box 167
 - Add Repository Parameter dialog box 256
 - Add Schema dialog box, XML checkpoint 649
 - Add Synchronization Point dialog box 846
 - Add/Remove dialog box, object identification 137, 153
 - Add/Remove Properties dialog box 199
 - add-ins
 - associating with a QuickTest test in Quality Center 1458
 - associating with a test 1295
 - QuickTest 33
 - Agent, Remote 1469
 - Allow other HP products to run tests and components option 1468
 - analog recording 394, 397
 - analyzing run results. *See* run results
 - API, using Windows 917
 - API-based text recognition 770
 - application
 - launch from QuickTest 1182
 - application areas
 - recovery scenarios, removing 1405
 - Application crash trigger 1368
 - Application Management, integrating with QuickTest 1555
 - application, sample 45
 - applications
 - closing 903
 - running 903
 - testing localized versions 1591
 - arguments, defining 955
 - ASCII 1230
 - ASP files 1585
 - Asset Comparison Tool 1493
 - Color Settings dialog box 1501
 - context menu commands 1500
 - legend 1499
 - opening 1493
 - options 1495
 - asset versions in QuickTest 1508
 - Asset Viewer 1502
 - long documents 1506
 - opening 1502
 - assets
 - adding to version control 1511
 - checking into in to version control 1514
 - checking out of version control 1511
 - definition of 1476
 - Assignment column, Keyword View 414
 - assistive properties, configuring 136
 - Associate Repositories dialog box 225
 - associating
 - add-ins with a test 1295
 - add-ins with test created in Quality Center 1460
 - function libraries 947, 949, 950
 - object repositories with actions 472
 - shared object repositories 225
 - attribute/<property name> notation 916
 - authentication
 - connecting to Quality Center 1446
 - auto-expand VBScript syntax 870, 927
 - AutoFill Lists dialog box 1239
 - automation
 - Application object 1437
 - definition 1432
 - development environment 1435
 - generating scripts for a test 1294
 - language 1435
 - object model 1431
 - object repository 270
 - type library 1435
 - Automation Engineer, role in Business Process Testing 1537
 - Automation toolbar, QuickTest window 72
 - Available Keywords pane 62, 1193
- ## **B**
- backslash (\) 795
 - baseline history
 - comparison to version history 1522
 - Baseline History dialog box 1519
 - baselines 1518

- bitmap checkpoint customization 1575
 - API 1585
 - sample 1600
 - tutorial 1589
 - Bitmap Checkpoint Properties dialog box 522
 - bitmap checkpoints 515
 - analyzing results for 1033
 - creating 518
 - customizing 516, 1575
 - modifying 518
 - pixel tolerance 516
 - RGB tolerance 516
 - bitmap comparer. *See* custom comparer
 - bookmarks 844
 - breakpoints
 - about 1078
 - deleting 1081
 - disabling/enabling 1080
 - setting 1079
 - using in Keyword View 423
 - built-in environment variables 650, 1283
 - business analyst
 - role in Business Process Testing 1508
 - business components, overview 15
 - Business Process Monitor, integrating with QuickTest 1527
 - Business Process Testing 1507
 - roles 1508
 - workflow 1511
 - business process tests 1512
 - overview 15
 - running 1515
 - button
 - add to toolbar or menu 1149
 - Button Appearance dialog box 1148
- C**
- calculations
 - in function libraries 878
 - in the Expert View 878
 - Call to WinRunner Function dialog box 1522
 - Call to WinRunner Test dialog box 1518
 - capture level options 1247
 - Java applications 1248
 - Oracle applications 1250
 - SAP Gui for Windows applications 1249
 - Terminal Emulator applications 1252
 - Windows applications 1251
 - Cell Identification tab, Database Checkpoint Properties dialog box 588
 - CGI scripts 1557
 - character set support, Unicode 4
 - check for updates 1234
 - Check In command 1483, 1486
 - Check In command, Quality Center 9.x 1497, 1499
 - Check Out command 1483
 - Check Out command, Quality Center 9.x 1497
 - Checkpoint Properties dialog box
 - for checking databases 535
 - for checking objects 508
 - checkpoints
 - about 495
 - adding existing 498
 - adding new 496
 - bitmaps 515
 - databases 575
 - definition 315, 495
 - fail 1101
 - images 512
 - in Expert View 830
 - modifying 512, 514
 - objects 506
 - parameterizing 659
 - standard, for checking text 570
 - supported by QuickTest 1546
 - tables 529, 530, 535
 - text 551, 552
 - text area 554
 - types 501
 - using formulas 1218
 - XML 591
 - Close method 875
 - closing application process 875, 1353, 1357

Index

- collection, properties. *See* programmatic descriptions
- collections, of virtual objects 1310
- Color Settings dialog box, Asset Comparison Tool 1473
- colors
 - setting in Keyword View 418
 - setting in Object Repository Comparison Tool 298
 - setting in Object Repository Merge Tool 265
- columns, displaying in Keyword View 416
- COM 1558
- command
 - add to toolbar or menu 1149
- command line options
 - deleting test results using 1007
 - Domain 1008
 - FromDate 1008
 - Log 1008
 - MinSize 1009
 - Name 1009
 - Password 1010
 - Project 1010
 - Recursive 1010
 - Server 1011
 - Silent 1011
 - Test 1011
 - UntilDate 1012
 - User 1012
- commands
 - Object Repository Comparison Tool 294
 - Object Repository Merge Tool 258
- Comment column, Keyword View 389
- comments
 - function libraries 877
 - the Expert View 877
 - the Keyword View 815
- Comments tab, To Do pane 1174
- CompareBitmaps method 1585
- comparing
 - shared object repositories 287
- comparing versions 1465
- Completing the Recovery Scenario Wizard screen 1364
- complex value 759
- components
 - run results. *See* run results
 - steps
 - adding 392
 - deleting 414
 - managing 412
 - moving 412
- conditional statements 797
 - using in Keyword View 423
- Configure Text Selection dialog box 561
- Configure value area 757
- configuring values 755
- conflict resolution
 - in merged object repository 280
 - settings, Object Repository Merge Tool 263
- connecting QuickTest to Quality Center 1418
- connection string, specifying for database checkpoints 580
- Constant Value Options button 759
- Constant Value Options dialog box 759
- constant value, defining 755
- content property check, on databases 576
- ControllerHostName, environment variable 650
- cookies 1557
- creation time identifier. *See* ordinal identifiers
- CreationTime property, using to identify an object 117
- currencies, setting custom format 1209
- Custom Active Screen Capture Settings dialog box 1244
 - Capture Level options 1247
 - General options 1247
 - Web options 1252
- custom comparer
 - See also* bitmap checkpoint
 - customization
 - creating 1579
 - documenting 1581
 - installing 1582
 - registering 1582
- custom number format, setting 1209

- custom objects, mapping 131
- custom settings
 - resetting 1246
- customize
 - toolbars and menus 1146
- Customize dialog box 1146
 - Commands tab 1149
 - Options tab 1157
 - Toolbars tab 1152
 - Tools tab 1155
- customizing bitmap checkpoints. *See* bitmap checkpoints, customizing

D

- Data Driver 662
- Data menu commands, Data Table 1209
- data sheets
 - action 1200
 - Global 1200
 - global/action, choosing 429
 - local 1200
- Data Table 25, 35, 1197
 - action data sheets 1200
 - Action tab 429
 - AutoFill Lists 1211
 - comparing versions 1465
 - Data menu commands 1209
 - data sheets 1200
 - design-time 1197
 - Edit menu commands 1207
 - editing tables 1202
 - File menu commands 1206
 - Format menu commands 1209
 - Global tab 429
 - importing data, in various formats 1202
 - iteration options for individual tests 1271
 - local data sheets 1200
 - location 1201
 - menu commands, using 1205
 - parameters, setting options for 641
 - run-time 1198
 - saving 1201

- scripting functions, using 1220
- Sheet menu commands 1207
- specifications 1204
- storing output values 674, 685
- table columns 639
- table rows 640
- using formulas 1216
- using with Quality Center 1212
- viewing results 1056
- worksheet functions 1216
- Database Checkpoint Properties dialog box 581
 - Cell Identification tab 588
 - Expected Data tab 585
 - Settings tab 586
- database checkpoints 575
 - about 575
 - analyzing results 1031
 - general information 583
 - modifying 590
 - specifying cell identification settings 588
 - specifying cells 583
 - specifying expected data 585
 - specifying value type 586
- Database Output Value Properties dialog box 715
- database output values 672, 713, 715
- Database Query wizard 577
- databases
 - connection string 580
 - creating a query in ODBC / Microsoft Query 1216
 - creating a query with Microsoft Query / SQL statement 579
 - creating checkpoints for 576
 - manually defining an SQL statement 577
 - result set 576
 - Specify SQL statement screen 580
- data-driven test 627, 674
- dates, setting custom format 1209
- Debug from Action 434
- Debug from Step 1076
- Debug toolbar, QuickTest window 23, 45

- Debug Viewer 36
 - Debug Viewer pane 36, 1082
 - Command tab 1092
 - Variables tab 1089
 - Watch tab 1083
 - debugging
 - accessing variables 1089
 - breakpoints
 - deleting 1081
 - disabling/enabling 1080
 - setting 1079
 - Debug from Action 434
 - Debug from Step 1076
 - function libraries 916, 1069
 - pausing runs 1078
 - Run to Action 434
 - Run to Step 1076
 - running commands 1092
 - tests 1069
 - tests, an example 1096
 - watching expressions 1083
 - default object identification settings 119
 - default optional steps 965
 - default parameter definition 758, 761
 - default properties, modifying 79, 162
 - defects, reporting 1013
 - automatically during test 1015
 - from Test Results 1013
 - Define Object Filter dialog box 144
 - delay, editing a step 833
 - deleting
 - actions 460
 - breakpoints 1081
 - objects from the object repository 153
 - repository parameters 233
 - test results 1004
 - dependencies
 - definition of 1448
 - Used By grid 1455
 - Using grid 1455
 - Dependencies tab, Quality Center 1454
 - Used By grid 1455
 - Using grid 1458
 - description, test objects 83
 - See also* test objects
 - descriptive programming. *See* programmatic descriptions
 - design-time Data Table 1197
 - development environment 1407
 - Dictionary object 487
 - difference types
 - Object Repository Comparison Tool 297
 - Dim statement
 - in the Expert View and function libraries 856
 - disconnecting from Quality Center 1422
 - disk space, saving 1564
 - display area
 - Script Editor 1391
 - Do...Loop statement
 - in the Expert View and function libraries 881
 - docked panes 1141
 - Documentation Only option 421
 - documenting a function 934
 - Domain command line option 1008
 - DOS commands, run within tests 889
 - dynamic list of values, for method argument 837
 - dynamic Web content 1555
 - dynamically generated URLs and Web pages 1556
- E**
- Edit menu commands, Data Table 1207
 - Edit Schema dialog box, XML checkpoint 621
 - Edit toolbar, QuickTest window 45
 - Edit XML dialog box, XML checkpoint 613
 - Editor Options dialog box 897
 - Element Value dialog box
 - XML checkpoint results 1047
 - encoding passwords 406
 - End Transaction button 45
 - End Transaction dialog box 1537
 - environment variables 645, 1283
 - built-in 650, 1283
 - files, with Quality Center 649

- storing output values 674, 686
- types 645
- environment variables, user-defined 1287
 - exporting 1289
 - external 647
 - internal 645
 - modifying 1287
 - viewing 1287
- environment, testing 5
- error behavior options for tests 1271
- errors in VBScript syntax 860
- Excel formulas
 - for parameterizing values 1217
 - in checkpoints 1218
 - in the Data Table 1216
- Excel. *See* Microsoft Excel
- ExecuteFile function 948
- ExecuteFile statement 920
- Exist property 1555
- Exist statement 821
- existing actions, inserting 464
- Expert View 825, 1553
 - about 827
 - basic action syntax 488
 - checkpoints 830
 - closing applications 875
 - customizing appearance of 895
 - finding text 847
 - general customization options 897
 - highlighting elements 900
 - replacing text 849
 - running applications 875
 - syntax for action parameters 489
 - syntax for action return values 490
 - understanding parameters 831
- export and replace local objects 193
- Export Report dialog box 1001
- exporting
 - local objects to shared object repository 193
 - object repository to XML file 243
 - Screen Recorder movies 996
 - tests to zip files 331
- expressions
 - using in the Expert View and function libraries 852

- extensibility, bitmap checkpoints. *See* bitmap checkpoint customization
- eXtensible Markup Language (XML) 1558
- external action
 - data location 451
 - definition 428
- external functions, executing from script 948
- external resources
 - saving to tests in Quality Center 326
- external user-defined environment variables 647

F

- FAQs 1551
- File menu commands, Data Table 1206
- File toolbar, QuickTest window 25
- filter
 - defining for objects 144
- Filter dialog box
 - Object Repository Comparison Tool 302
 - Object Repository Merge Tool 283
- filter properties (Smart Identification) 121
- filtering
 - repositories in Object Repository Comparison Tool 302
 - target repository 282
- Find & Replace dialog box, object repositories 154
- Find dialog box
 - Expert View 847
 - Object Repository Comparison Tool 304
 - Object Repository Merge Tool 284
 - Test Results 990
- Find in Repository button 510, 513, 524, 537, 561, 583, 608, 680, 694, 705, 716, 728
- floating panes 1142
- Flow pane
 - Script Editor 1386
- fonts, setting in Keyword View 418
- For...Each statement
 - in the Expert View and function libraries 880

- For...Next statement
 - in the Expert View and function libraries 879
- Format menu commands, Data Table 1209
- formulas
 - for parameterizing values 1217
 - in checkpoints 1218
 - in the Data Table 1216
- FromDate command line option 1008
- function arguments, passing parameters
 - from QuickTest to WinRunner 1525
- function calls
 - dragging and dropping 34, 1165
- Function Definition Generator 927
 - about 923
 - defining a function 927
 - documenting a function 934
 - opening 925
 - previewing function code 936
 - registering a function 928
- function libraries 905
 - about 14
 - associating current 921
 - closing in the Script Editor 1402
 - comparing versions 1465
 - creating 909
 - creating in the Script Editor 1400
 - customizing appearance of 895
 - customizing general options 897
 - debugging 916, 1069
 - description 30
 - editing 914
 - editing in the Script Editor 1400
 - finding text 847
 - general options 897
 - highlighting elements 900
 - in Script Editor 1397
 - managing 908
 - modifying associated 922
 - navigating 913
 - opening 909, 918
 - opening in the Script Editor 1398
 - pausing runs 1078
 - properties 1389
 - read-only, editing 916
 - replacing text 849

- saving 911
 - saving in the Script Editor 1401
 - specifying for a test 1274
 - working with 1397
 - working with associated 919
- functions
 - code
 - finalizing 937
 - inserting 937
 - user-defined 905

G

- General > Text Recognition pane 742
- general options 897
- General Text Recognition pane 742
- Generate Script option 1410
- GetDefaultConfigurationString method 1585
- GetHelpFilename method 1585
- GetROProperty method 886
- Global data sheet 429, 1200
- global Data Table parameter 643
- global testing options 1231
- global/action data sheets, choosing 429
- Go To dialog box 843
- GroupName, environment variable 650
- guidelines
 - user-defined functions 945

H

- hidden mode 1443
- History tab, Quality Center 1453
- HP Application Management, integrating
 - with QuickTest 1527
- HP Micro Player 996
- HP Quality Center. *See* Quality Center
- HP Software Support Web site xxv
- HP Software Web site xxv

I

- IBitmapCompareConfiguration interface 1585
- icons
 - display large or small 1157

- identification properties 79, 83
 - viewing 97
- If...Then...Else statement
 - in Expert View and function libraries 883
- Image Checkpoint Properties dialog box 512
- image checkpoints
 - comparing image contents 514
 - editing the property value 514
- importing
 - object repository from XML file 242
 - tests from zip files 331
- index identifier. *See* ordinal identifiers
- Index property
 - programmatically descriptions 871
 - using to identify an object 115
- Information pane 37
- initialization scripts 1405
- Input Parameters tab, Run dialog box 962
- Insert New Action dialog box 437
- Insert Report dialog box 812
- Insert toolbar, QuickTest window 45
- IntelliSense 833, 898
- internal user-defined environment variables 645
- Item cell 395
- Item column, Keyword View 387
- Item list 396
- item, selecting
 - from Item list 396
 - from shared object repository 396
 - from your application 399
- iterations 482, 639
 - options for individual tests 1271
- IVerifyBitmap interface 1585

J

- Java applications
 - capture level options 1248
- JavaScript 1407
- Jump to Step in QuickTest, from Test Results window 987

K

- key assignments
 - in Expert View 902
 - in function libraries 902
- key column 545, 589
- keyboard commands, sending to Web objects 1559
- keyboard shortcuts
 - in Expert View 902
 - in function libraries 902
 - in Keyword View 415
- Keyword View 28, 383, 385
 - columns, description of 386
 - columns, displaying 416
 - display options 416
 - fonts and colors 418
 - keyboard keys 415
 - steps, adding 392
 - steps, adding after block 409
 - steps, deleting 414
 - steps, modifying 410
- Keyword View tab 28
- keyword-driven testing
 - analyzing your application 342
 - automation infrastructure 341
 - configuring QuickTest 347
 - creating function libraries 345
 - creating test steps 348
 - creating tests 348
 - methodology 341
 - overview 336
 - running tests 350
 - setting up object repositories 343
 - troubleshooting tests 350
- Knowledge Base xxv

L

- language 1407
- language support, Unicode 4
- layout
 - customizing QuickTest window 1135
 - moving panes 1136
 - moving tabs 1136
 - restoring default 1144

Index

- learning objects 225
- Libraries tab, Quality Center 1453
- license information 17
 - modifying 17
- list of values, for method argument 388, 783, 837
- LoadRunner, integrating with QuickTest 1527
- local data sheet. *See* action data sheets
- local Data Table parameters 644
- local object repositories 89, 92
 - copying objects to 195
 - exporting and replacing 193
 - merging 269
- local objects, exporting to shared object repository 193
- local parameter 404
- Local System Monitor pane 1296
- local test 428
- LocalHostName, environment variable 650
- localization 645, 1201
- localized applications, testing 1563
- Locate Missing Actions dialog box 1184, 1187
- location identifier. *See* ordinal identifiers
- Location property, using to identify an object 115
- Log command line option 1008
- loop statements 803
 - using in Keyword View 423
- low-level recording 368, 375, 1552

M

- maintaining tests 1101
- Maintenance Run Mode 1104
- Maintenance Run Wizard
 - Smart Identification screen 1120
- Manage Repository Parameters dialog box 229
- Managing 1479
- mandatory properties, configuring 108
- manual steps 410
- manual tests 421
- Map Shared Object Repository Parameters dialog box 202

- mapping
 - calls to missing actions 1184
 - custom objects 131
 - missing actions 1183
 - repository parameters 202
 - unmapped object repositories 1191
 - unmapped repository parameters 1194
- mathematical formulas, in the Data Table 1216
- menu
 - create new 1149
- menu bar, QuickTest window 23
- Mercury Tours, sample application 17
- merging
 - local object repositories 269
 - shared object repositories 247
- messages
 - displaying during the run session 814
 - generating 812
 - sending to test results 812
- meta tags 1557
- methods
 - adding new or changing behavior of 939
 - native 887
 - run-time object 887
 - user-defined 939
 - See also* operations
- Microsoft Excel 1202, 1216
- Microsoft Query
 - choosing a database for a database checkpoint 579, 1216
- Microsoft Visual Basic scripting language 13
- MinSize command line option 1009
- missing resources 1179
- Missing Resources pane 38
 - about 1180
 - filtering 1181
 - unmapped repository parameters 1194
 - unmapped shared object repositories 1191
- Modify Row Range dialog box 711
- modifying
 - your license 17

- movies of your run session
 - capturing 1255
 - capturing and viewing 994
 - exporting 996
 - removing from the test results 995
 - setting options to capture 1255
 - viewing results in Quality Center 991
- moving a step 412
- multiple actions in tests 427
- multiple documents, working with 1159
- multiple text block mode, text recognition 745

N

- Name and Description screen 1363
- Name command line option 1009
- names
 - modifying for test objects 169
- native methods *See* native operations
- native operations 87
 - viewing 97
- native properties 87
 - viewing 97
- Navigate and Learn option 225
- nesting actions 453, 476
- New Merge dialog box 267
- node, Options dialog box 1232
- node, Test Settings dialog box 1262
- non-reusable action 428

O

- Object Configuration Screen 1323
- object identification
 - generating automation scripts 120
 - restoring defaults 119
- Object Identification dialog box 107
- Object Mapping dialog box 131
- object model
 - automation 1403
 - definition 1404
- object property values
 - restoring default 165, 168
 - specifying or modifying 163
 - viewing 197

- Object property, native methods 888
- Object property, run-time object methods 888
- object repositories
 - adding objects 136
 - associating with actions 446
 - choosing 92
 - closing 221
 - converting from earlier version 217
 - copying, pasting, and moving objects 150
 - creating 217
 - deleting objects 153
 - exporting local and replacing 193
 - exporting local objects 193
 - exporting to XML 243
 - importing from XML 242
 - local 92
 - locating objects 159
 - managing 208
 - managing associations 199
 - managing using automation 244
 - missing 1179
 - modifying 224
 - opening 217
 - saving 219
 - setting for tests 1274
 - shared 94
 - unmapped 1191
- Object Repository Comparison Tool 287
 - color settings 298
 - difference types 297
 - filtering the repositories 302
 - repository panes 290
 - statistics 301
 - synchronizing repositories 303
 - window 289
- Object Repository Manager 210
- Object Repository Merge Tool 247
 - changing the view 252
 - color settings 265
 - conflict resolution settings 263
 - conflicts 277
 - filtering the target repository 282
 - primary repository pane 254
 - resolution options pane 254

Index

- resolving conflicts 280
- secondary repository pane 254
- target repository pane 252
- window 250
- object repository types 89
- Object Repository window 183
 - buttons 185
 - Edit toolbar 185
 - Filter toolbar 187
 - Object details area 190
 - options 188
 - test object details 162
 - understanding 182
- Object Selection dialog box 399
- Object Spy 97, 100
- Object state trigger 1340
- objects
 - adding using navigate and learn 225
 - deleting from object repository 153
 - dragging and dropping 34, 1165
 - identification 105
 - identifying 79
 - methods, run-time 887
 - properties, run-time 887
 - viewing operations 79
 - See also* test objects
- OCR 742
- ODBC, choosing a database for a database checkpoint 1216
- online documentation xxii
- online resources xxiv
- Open QuickTest Test dialog box 322
- Open Test dialog box 1429
- Open Test from Quality Center dialog box 1496
- Open Test from Quality Center Project dialog box 1427
- operation
 - arguments 404
 - selecting for step 403
 - selecting from Item list 395, 396
- Operation cell 403
- Operation column, Keyword View 388
- operations
 - native 87
 - run-time object 87
 - test object 87
- Option Explicit statement 945
- optional steps 963
 - default 965
 - setting 964
- Options dialog box 1232
 - Active Screen pane 1240
 - Folders pane 1237
 - General > Text Recognition pane 742
 - General pane 1234
 - Generate Script option 1234, 1410
 - node 1232
 - Run > Screen Capture pane 1255
 - Run pane 1253
- Oracle applications
 - capture level options 1250
- ordinal identifiers 113
 - specifying for test objects 177
- OS, environment variable 650
- OSVersion, environment variable 650
- output types 683
 - action parameters 684
 - Data Table 685
 - environment variables 686
 - test parameters 684
- output value
 - adding existing 736
- output value categories
 - database output values 672
 - standard output values 671
 - text area output values 671
 - text output values 671
 - XML output values 672
- Output Value Properties dialog box 679
- output values
 - creating for text 690
 - database 713, 715, 717
 - definition 669
 - editing 675
 - object properties 679
 - standard 676

- storing in action or test parameters 673
 - storing in Data Table 674
 - storing in environment variables 674
 - supported by QuickTest 1548
 - tables 698, 703, 708
 - text 688, 692
 - text area 690
 - viewing 675
 - viewing results 1055
 - XML 718, 727
- output.txt log file 1541
- outputting
 - database values 713
 - property values 676
 - text values 688, 690
 - values 669
 - XML values 718
- Owner Description, Used By grid 1457
- Owner ID, Used By grid 1456
- Owner Name, Used By grid 1457
- Owner Type, Used By grid 1456
- P**
- panes
 - auto-hiding 1141
 - Available Keywords 34
 - customizing layout 1136
 - Debug Viewer 36
 - docked 1141
 - floating 1142
 - Information 37
 - Missing Resources 38
 - moving 1136
 - Process Guidance 39
 - Resources 40
 - Test Flow 41
 - To Do 42
- parameter definition, default 758, 761
- Parameter Options button 758
- Parameter Options dialog box 636
- Parameter reserved object 1283
- parameter types
 - action parameters 626
 - Data Table parameters 639
 - environment variable parameters 645
 - random number parameters 655
 - test parameters 626
- parameter values
 - action calls 483
 - defining 755
- parameterization example 657
- parameterization icon 630, 632, 760
- parameterized values, viewing in test results 1053
- parameterizing
 - methods 628
 - property values using repository parameters 235
 - tests, example 657
 - using the Data Driver 662
 - values 625
- parameters
 - action 460, 476, 637
 - action guidelines 479
 - environment variables, user-defined 1285, 1287
 - handling unmapped object repository 1194
 - in the Expert View 831
 - output from previous action call 637
 - parent action 637
 - passing to a WinRunner function 1525
 - passing to a WinRunner test 1520
 - repository 228
 - adding 230
 - deleting 233
 - managing 229
 - mapping 202
 - missing in 1179
 - modifying 232
 - setting for actions 445
 - specifying for tests 1280
 - syntax for calling action 489
 - test 637
- passing data between actions 429
- Password command line option 1010
- Password Encoder dialog box 406
- passwords, encoding 406
- PathFinder.Locate, statement 1240

- paths, absolute and relative 316
- pausing run sessions 1078
- percentages, setting custom format 1209
- performance testing products, integrating
 - with QuickTest 1527
- performance, improving 1564
- permissions
 - required for Quality Center 16
 - required to run QuickTest 16
- pixel tolerance, in bitmap checkpoints 516
- Pointing Hand
 - tips for working with 99
- Pop-up window trigger 1340
- possible values, for method argument 837
- post-recovery test run options 1330
- Post-Recovery Test Run Options screen 1361
- previewing function code 936
- primary repository 248
- primary repository pane 254
- Print dialog box
 - Test Results window 999
- Print Preview dialog box 997
- Print, utility statement 814
- printing
 - function libraries 917
 - tests 332
- priority
 - setting for recovery scenarios 1376
- Process Guidance 1225
 - panes 1222
 - starting 1224
- Process Guidance panes 39
- Product Information button 73
- Product Information window 73
- ProductDir, environment variable 650
- ProductName, environment variable 650
- ProductVer, environment variable 650
- programmatic descriptions 206, 863
 - description objects 868
 - Index property 871
 - performing checks on objects 872
 - statement 864
 - variables 864
 - With statement 867

- programming 1553
 - comments 815
 - conditional statements 797
 - displaying messages during the run session 814
 - Expert View and function libraries 825
 - function libraries 825
 - generating messages 812
 - loop statements 803
 - sending messages to test results 812
 - Step Generator 776, 777
 - VBScript 853
- project (Quality Center)
 - connecting to 1418
 - disconnecting from 1422
 - opening tests in 1426
 - saving tests to 1425
- Project command line option 1010
- properties 433, 1387, 1389
 - adding for test object descriptions 171
 - CreationTime 117
 - default 79, 162
 - defining new for test object 174
 - deleting from a test object description 177
 - Index 115
 - Location 115
 - native 87, 887
 - run-time object 887
 - setting for action calls 481
 - setting for actions 441
 - test object 87
 - viewing for recovery scenarios 1368, 1376
 - viewing for steps in Keyword View 422
 - See also* identification properties
- Properties tab
 - Table Checkpoint Properties dialog box 546
 - Table Output Value Properties dialog box 709
- property collection. *See* programmatic descriptions

property values
 specifying in the test object
 description 235
 synchronization points 817

Q

QA engineer. *See* Automation Engineer

QCUtil object 1424

Quality Center 1415

 associated function libraries 919

 connecting QuickTest to 1418

 Connectivity Add-in 1424

 Data Table 1212

 Dependencies tab 1454

 disconnecting from 1422

 environment variable files 649

 History tab 1453

 integrating with QuickTest 1424,
 1461

 Libraries tab 1453

 managing the testing process 14

 opening tests in 1426

 relative paths 1450

 reporting defects

 automatically 1015

 manually 1013

 running QuickTest tests remotely
 1440

 saving tests to 326

 saving tests to a project 1425

 using QuickTest with 14

 version control 1479

 version control management 1480

Quality Center 9.x 1495

 version control for 1496

Quality Center Connection - Server

 Connection dialog box 1418

Quality Center OTA 1424

query file, for a database checkpoint

 creating 579, 1216

 working with ODBC / Microsoft

 Query 1216

QuickTest

 about 3

 access permissions, required 16

 automation object model 1403

 getting started 19

 integrating with HP application

 management and performance

 testing products 1527

 layout 1135

 customizing 1135

 product information 73

 starting 20

 updating software 18

 window. *See* QuickTest window

QuickTest Asset Viewer 1474

QuickTest Automation Reference 1411

QuickTest Print Log window 814

QuickTest Professional Asset Upgrade Tool
 for Quality Center 323, 330, 1382,
 1426

QuickTest window

 Action toolbar 23, 46

 auto-hiding panes 1141

 Automation toolbar 44

 Available Keywords pane 34

 customizing layout 1135

 Data Table 25

 Debug toolbar 23

 Debug Viewer pane 36

 Edit toolbar 45

 File toolbar 25

 Information pane 37

 Insert toolbar 45

 look and feel 27

 menu bar 23

 Missing Resources pane 38

 moving panes 1136

 moving tabs 1136

 multiple documents 1159

 Process Guidance panes 39

 Resources pane 40

 restoring default layout 1144

 Standard toolbar 44

 status bar 25

 Test Flow pane 41

 theme 27

 title bar 25

 To Do pane 42

 Tools toolbar 45

View toolbar 46

R

random number parameters 655

Readme xxii

recording

analog 368

low-level 368, 1552

tests 364

time, improving 1564

recovery operations 1330

Close application process 1353

Function call 1353

Keyboard or mouse operation 1353

Restart Microsoft Windows 1353

Recovery Scenario Manager Dialog Box 1334

Recovery Scenario Wizard 1338

Click Button or Press Key screen 1355

Close Processes screen 1357

Completing the Recovery Scenario

Wizard screen 1364

Function screen 1358

Name and Description screen 1363

Post-Recovery Test Run Options

screen 1361

Recovery Operation - Click Button or

Press Key screen 1355

Recovery Operation - Close Processes

screen 1357

Recovery Operation - Function Call

screen 1358

Recovery Operation screen 1353

Recovery Operations screen 1352

Select Object screen 1345

Select Processes screen 1350

Select Test Run Error screen 1349

Select Trigger Event screen 1340

Set Object Properties and Values

screen 1348

Specify Pop-up Window Conditions

screen 1342

recovery scenarios 1329

associating with tests 1373

comparing versions 1465

copying 1371

deleting 1370

disabling 1377

files 1334

locating missing 1192

modifying 1370

removing from tests 1376

removing missing 1194

saving 1365

setting priority 1376

viewing properties 1368, 1376

Recursive command line option 1010

redirection of server 1557

registering functions 928

registering methods 939

RegisterUserFunc statement 928, 939, 941

regular expressions 762

backslash (\) 767

defining 765

for constants 757

for property values 763

in checkpoints 764

using in function libraries 852

using in the Expert View and function

libraries 852

Related Description, Using grid 1459

Related ID, Using grid 1458

Related Name, Using grid 1459

Related Type, Using grid 1458

relative path 316

relative paths

Quality Center 1450

remote access to QuickTest 1440

Remote Agent 1441

Remote Agent Settings dialog box 1443

Replace dialog box

Expert View 849

function libraries 849

report. *See* Test Results window

reporting defects

automatically 1013

manually 1013

reports, filter 893

repositories. *See* object repositories

Repository Parameter dialog box 235

repository parameters 228

adding 230

- deleting 233
- managing 229
- mapping 202
- modifying 232
- parameterizing values 235
- repository types 89
- reserved objects 919
- Resolution Options pane, Object Repository Merge Tool 254
- resolving conflicts, Object Repository Merge Tool 280
- resources
 - adding to version control 1483
 - checking into version control 1486
 - checking out of version control 1483
 - definition of 1448
 - managing 40, 1161
 - missing in test 1179
- Resources and Dependencies model
 - glossary 1448
 - overview 1449
- Resources pane 40, 1161, 1388
- restoring QuickTest default layout 1234
- Result Details tab, Test Results window 975
- result set 576
- ResultDir, environment variable 651
- Results Location tab, Run dialog box 960
- results. *See* run results
- reusable actions 428
- RGB tolerance, in bitmap checkpoints 516
- roles in Business Process Testing 1508
- Run > Screen Capture pane 1255
- Run dialog box 955
- Run from Action 434
- Run from Step 956
- run options, in the Options dialog box 1253
- run properties, setting for action calls 482
- run results 969
 - checkpoints 1028
 - customizing display 1019
 - deleting with command line options 1007
 - deleting with Test Results Deletion Tool 1004
 - enabling and filtering 893
 - exporting to a file 1001
 - finding 985, 990
 - output values 1055
 - parameterized values 1053
 - previewing before printing 997
 - printing 999
 - reporting defects automatically 1015
 - reporting defects manually 1013
 - Run-Time Data Table 1056
 - schema 1019
 - sending messages to 812
 - Test Results window 971
 - viewing for a selected run 981
 - viewing WinRunner steps 1017
- run sessions
 - creating test objects programmatically 206
 - disabling recovery scenarios 1377
 - modifying identification properties 206
 - pausing 1078
 - working with test objects 206
- Run to Action 434
- Run to Step 1076
- running tests 953
 - advanced issues 1552
 - from a Quality Center project 1437
 - from a step 956
 - from an action 434
 - Run dialog box 955
 - running WinRunner tests 1518
 - to update expected results 1125
 - Update Run dialog box 1128
 - using optional steps 963
 - viewing results 980
- run-time
 - Data Table 1198
 - objects 887
 - settings, adding and removing 1305
- Run-Time Data Table 1056
- run-time object methods. *See* native operations
- run-time object properties. *See* native properties

- run-time objects 79, 83
 - viewing properties and operations 97
- S**
- sample application, Mercury Tours 17
- SAP Gui for Windows applications
 - capture level options 1249
- Save QuickTest Test dialog box 324
- Save Shared Object Repository dialog box 285
- Save Test dialog box 1425
- Save Test with Resources dialog box 328
- ScenarioId, environment variable 651
- scenarios. *See* recovery scenarios
- Schema Validation dialog box, XML checkpoint 618
- schema, for run results 1019
- Screen Recorder tab, Test Results window 994
- ScreenTips
 - display 1157
- Script Editor 1381
 - customizing the window 1384
 - display area 1391
 - Flow pane 1386
 - function libraries 1397
 - main window 1383
 - Resources pane 1388
 - tests 1393
- scripts, test. *See* tests
- secondary object repository 248
- secondary repository pane 254
- Section 508, Web Content Accessibility Guidelines 6
- Select Action dialog box 466, 469
- Select Object for Step dialog box 396
- Select Object screen 1345
- Select Processes screen 1350
- Select Test Run Error screen 1349
- Select Trigger Event screen 1340
- selecting a test object
 - from Item list 396
 - from shared object repository 396
 - from your application 399

- server
 - Quality Center, disconnecting from 1422
 - redirections 1557
 - server-side connections 1557
- Server command line option 1011
- session IDs 1557
- Set Object Properties and Values screen 1348
- Set statement
 - in the Expert View and function libraries 856
- Setting object 1302
- settings 433
- Settings tab, Database Checkpoint Properties dialog box 586
- SetTOProperty method 206
- SGML 1558
- shared object repositories 89, 94
 - associating with actions 446
 - comparing 287
 - comparing versions 1465
 - managing associations 199
 - merging 247
 - unmapped 1191
 - Update from Local Repository 269
- shared object repository window 215
- Sheet menu commands, Data Table 1207
- shortcut keys
 - display 1157
 - in Keyword View 415
 - in QuickTest 46
- shortcuts
 - for menu items 46
 - in Expert View 902
 - in function libraries 902
 - in QuickTest 46
 - Object Repository Comparison Tool 294
 - Object Repository Merge Tool 258
- Silent command line option 1011
- Silent Test Runner 1538
 - dialog box 1539
- single text block mode, text recognition 744

- Smart Identification
 - analyzing information 1024
 - configuring 121
 - disabling during test runs 1272
 - enabling from the Object Identification dialog box 118, 120
 - Smart Identification Properties dialog box 126
 - software updates 18
 - specifications for Data Table 1204
 - Specify Pop-up Window Conditions screen 1342
 - Specify SQL statement screen, for creating database checkpoints 580
 - Split Action dialog box 456
 - splitting actions 455
 - Spy. *See* Object Spy
 - standard checkpoints
 - analyzing results 1029
 - specifying timeout 511
 - standard output values 671
 - creating 676
 - specifying 679
 - Standard toolbar, QuickTest window 44
 - Start Page 31
 - Start Transaction dialog box 1536
 - starting QuickTest 20
 - statement completion 833, 898
 - statements, using in Keyword View 410
 - Statistics dialog box 276
 - Comparison Tool 301
 - status bar
 - Object Repository Comparison Tool 292
 - Object Repository Merge Tool 255
 - QuickTest window 25
 - Step commands 1072
 - Step Generator 776, 777
 - Step Generator dialog box 780
 - steps
 - adding 392
 - adding after block 409
 - adding to Keyword View 392
 - deleting 414
 - deleting from Keyword View 414
 - inserting 777
 - manual 410
 - modifying in Keyword View 410
 - moving 412
 - optional 963
 - viewing properties in Keyword View 422
 - still images of your application, capturing and viewing 993
 - Stop command shortcut key 1255
 - Subject Matter Expert, role in Business Process Testing 1508
 - Summary column, Keyword View 389
 - synchronization points
 - creating 817
 - inserting 818
 - synchronization timeout
 - setting 1271
 - synchronizing repositories
 - Object Repository Comparison Tool 303
 - synchronizing tests 816
 - modifying timeout values 822
 - synchronization point 817
 - waiting for objects to appear 821
 - waiting for specified property values 817
 - syntax
 - actions 488
 - for action parameters 489
 - for action return values 490
 - syntax errors, VBScript 860
 - System Counters
 - enabling 1296
 - setting 1296
 - system counters, results 1063
 - System Monitor tab 1063
 - exporting results 1063
 - SystemTempDir, environment variable 651
 - SystemUtil.Run method 875
- ## T
- Table Checkpoint Properties dialog box 535
 - Expected Data tab 542
 - Properties tab 546
 - Table Content tab 536

- table checkpoints
 - about 529
 - analyzing results 1031
 - creating 530
 - general options 537
 - modifying 548
 - specifying cell identification settings 544
 - specifying cells 540
 - specifying expected data 542
 - specifying value type 543
 - Table Content tab 538
 - Table Properties tab 538
- Table Content tab
 - Table Checkpoint Properties dialog box 536
 - Table Output Value Properties dialog box 703
- Table Output Value Properties dialog box 703
 - Properties tab 709
 - Table Content tab 703
- table output values 703
 - modifying output options 711
 - modifying row range 711
 - Table Content tab 706
 - Table Properties tab 706
- table properties
 - specifying which to check 547
 - specifying which to output 710
- target repository 248
 - saving 285
- target repository pane 252
- Task Editor dialog box 1177
- Tasks tab, To Do pane 1171
- tasks, managing 42
- template tests 1430, 1432
- templates, actions 462
- Terminal Emulator applications
 - capture level options 1252
- test batches, running 966
- Test command line option 1011
- test database, maintaining 1405
- test flow (actions) 435
- Test Flow pane 41
 - actions 41, 431
- test object methods 87
- test object operations 87
- test object properties 87
 - See also* identification properties
- test object properties. *See* identification properties
- test objects 79, 83
 - adding
 - description properties 171
 - to object repository 136
 - copying to local repository 195
 - copying, pasting, and moving in object repository 150
 - creating in run sessions 206
 - creating using programmatic descriptions 206
 - defining new 147
 - defining new properties 174
 - deleting description properties 177
 - dragging and dropping 182, 225
 - finding 154
 - highlighting in an application 157
 - identifying 79
 - in run sessions 206
 - locating in object repository 154, 159
 - managing 135
 - modifying
 - in run sessions 206
 - names 169
 - properties 162
 - properties during run sessions 206
 - property values, replacing 154
 - property values, retrieving and setting 886
 - renaming 169
 - selecting
 - from application 399
 - from Item list 396
 - from shared object repository 396
 - specifying ordinal identifiers 177
 - viewing properties 197
 - viewing properties and operations 97

- test parameters 626, 635
 - setting options 636
 - storing output values 673, 684
 - using in steps 1283
- test resources, missing 1179
- Test Results Deletion Tool 1004
 - running from the command line 1007
- Test Results toolbar, Test Results window 977
- Test Results tree 974
- Test Results window 971
 - customize appearance 979
 - Jump to Step in QuickTest 987
 - Result Details tab 975
 - run results toolbar 977
 - run results tree 974
 - Screen Recorder tab 994
 - System Monitor tab 1063
 - theme 979
- test results. *See* run results
- Test run error trigger 1340
- Test Run Log 1541
- test run time, improving 1564
- test set 1438
- Test Settings dialog box 1262
 - Environment pane 1283
 - Generate Script option 1410
 - Local System Monitor pane 1296
 - node 1262
 - Parameters pane 1280
 - Properties pane 1265
 - Recovery pane 1291
 - Resources pane 1274
 - Run pane 1270
- test versions in QuickTest 1480
- TestDir, environment variable 651
- TestDirector. *See* Quality Center
- testing options
 - during a test run 1301
 - restoring 1305
 - retrieving 1304
 - run-time 1305
 - setting 1302
 - setting for all tests 1231
 - setting for an individual test 1261
- testing process 7
 - analyzing test results 13, 341
 - creating tests 8, 11, 336, 339
 - reporting defects 13, 341
 - running tests 12, 340
- TestIteration, environment variable 651
- TestName, environment variable 651
- tests
 - about test steps 313
 - adding to version control 1483
 - and components, a comparison 1516
 - associating recovery scenarios with 1373
 - checking into version control 1486
 - checking out of version control 1483
 - checkpoints. *See* checkpoints
 - closing in the Script Editor 1397
 - comparing versions 1465
 - creating 309, 321, 335
 - creating in Quality Center using a template test 1434
 - debugging 1069
 - diagram 426, 464, 465
 - disabling recovery scenarios 1377
 - editing in the Script Editor 1395
 - enhancing 315
 - local 428
 - maintaining 1101
 - managing 321
 - managing in Quality Center 14, 1415
 - opening in a Quality Center project 1426
 - opening in QuickTest 322
 - opening in the Script Editor 1393
 - opening tests from older versions 323
 - parameterizing, example 657
 - pausing runs 1078
 - printing 332
 - properties 1387, 1389
 - recording 361, 364
 - removing recovery scenarios from 1376
 - running 953
 - running from a step 956
 - running from an action 434
 - running using optional steps 963
 - save as 326
 - saving 324

- saving in the Script Editor 1396
 - saving to a Quality Center project 1425
 - saving to Quality Center 326
 - saving with external resources 326
 - settings 433
 - unzipping 331
 - updating 1125
 - viewing and comparing versions 1461
 - working with 1393
 - zipping 331
 - See also* run results
- Text Area Checkpoint Properties dialog box 557
- Text Area Output Value Properties dialog box 692
- text area output values 671
 - creating 690
- Text Checkpoint Properties dialog box 557
- text checkpoints 551, 552
 - analyzing results 1036
 - configuring the text selection 561
 - modifying 570
 - setting options 561
 - specifying the checked text 564
 - specifying the text after 567
 - specifying the text before 566
 - specifying timeout 569
 - standard checkpoints 570
 - types 551
- TEXT function in Data Table worksheet 1216
- Text Output Value Properties dialog box 692
- text output values 671
 - creating 688
 - specifying 692
- text recognition 742
 - guidelines 746
 - multiple text block mode 745
 - single text block mode 744
 - supported environments 748
 - use-case scenario 750
- Text Recognition pane, Options dialog box 742
- text values, outputting 688, 690
- text, checking
 - using text area checkpoints 554
- timeout
 - setting 1271
 - specifying for standard checkpoint 511
 - specifying for text checkpoints 569
- times, setting custom format 1209
- title bar, QuickTest window 25
- To Do pane 42, 1170
 - Comments tab 1174
 - Tasks tab 1171
- toolbar buttons
 - display text labels 1152
- toolbars
 - default settings 1152
 - Object Repository Comparison Tool 293
 - Object Repository Merge Tool 257
 - QuickTest window
 - Action 46
 - Automation 44
 - Debug 23, 45
 - Edit 45
 - File 25
 - Insert 45
 - Standard 44
 - Tools 45
 - View 46
 - show and hide 1152
- toolbars and menus
 - customize 1146
- Tools toolbar, QuickTest window 45
- ToolTips
 - display 1157
- transactions 1534
 - defining 1534
 - ending 1537
 - inserting 1536
 - measuring 1534
- Tree View. *See* Keyword View
- trigger
 - Application crash 1340
 - events 1330
 - Object state 1340
 - Pop-up window 1340
 - test run error 1340
- Troubleshooting and Knowledge Base xxv

TSL functions, calling from QuickTest 1522
 type library 1407
 typing delay, when editing a step 833

U

Unicode 4
 unregistering methods, using the
 UnregisterUserFunc statement 943
 UnregisterUserFunc statement 939
 UntilDate command line option 1012
 unzipping tests 331
 Update Run dialog box 1128
 UpdatingActiveScreen, environment
 variable 651
 UpdatingCheckpoints, environment variable
 651
 upgrading assets 323, 330, 1382, 1426
 Used By grid 1455
 User command line option 1012
 user-defined
 functions. *See* user-defined functions
 methods 939
 properties, accessing 888
 test objects, mapping 131
 user-defined functions 905
 adding a tooltip to 934
 documenting 934
 finalizing 937
 Function Definition Generator 923
 generating additional 936
 guidelines for 945
 previewing code in Function
 Definition Generator 936
 registering 928
 UserName, environment variable 651
 Using grid 1455

V

Value cell 404
 Value column, Keyword View 388
 Value Configuration Options dialog box
 630, 760
 VALUE function in Data Table worksheet
 1216

values
 configuring 755
 input 404
 outputting 669
 parameterizing 625
 restoring default for object properties
 165, 168
 specifying for object properties 163
 viewing for object properties 197
 variables
 environment 1283
 unique in global scope 946
 See also environment variables,
 user-defined
 VBScript 1407
 associated function libraries
 with Quality Center 919
 auto-expand syntax 842, 899
 documentation 876
 formatting text 859
 syntax 853
 syntax errors 860
 version control 1479, 1480
 adding assets to 1483
 baseline history 1491
 cancelling check out 1487
 checking assets out of 1483
 checking tests in to 1486
 commands 1482
 Quality Center 9.x 1496
 version history 1488
 version history
 comparison to baseline history 1494
 Version History dialog box 1488
 version manager 1480
 version manager, Quality Center 9.x 1496
 versions
 comparing 1465
 1465
 viewing and comparing 1461
 View toolbar 46
 Virtual Object Manager 1327
 Virtual Object Manager Dialog Box 1313
 Virtual Object wizard 1315

Index

- virtual objects 1309
 - defining 1314
 - removing 1327
- Visual Basic 1407
- Visual C++ 1407
- Visual Studio.NET 1407
- VuserId, environment variable 651

W

- W3C Web Content Accessibility Guidelines 6
- Wait statement 821
- WaitProperty statement 817
- Web
 - advanced issues, FAQ 1557
 - sending keyboard commands to Web objects 1559
- Web content accessibility checkpoints
 - in test results 1048
- Web content, dynamic 1555
- While statement, in the Expert View and function libraries 882
- Windows API 889
- Windows applications
 - capture level options 1251
- Windows command line options 1007
- Windows dialog box 1159
- WinRunner
 - calling tests from QuickTest 1518
 - calling TSL functions from QuickTest 1522
 - function arguments, passing
 - parameters from QuickTest 1525
 - tests, passing parameters from QuickTest 1520
 - viewing WinRunner steps in test results 1017
 - working with 1517
- With statements
 - entering manually 884
 - generating automatically, while recording 808
 - generating for existing actions 809

- in the Expert View 806
- removing 811
 - With Generation Results window 810
- workflow in Business Process Testing 1511
- worksheet functions in the Data Table 1216
- wscript.exe 1408

X

- XML
 - checkpoint results
 - attribute details 1041
 - checkpoint summary 1039
 - checkpoints 591
 - Add Schema dialog box 621
 - analyzing results 622, 1037
 - Edit Schema dialog box 621
 - for files 600
 - for test objects 603
 - for web page/frame 595
 - modifying 622
 - namespace 593, 623, 718
 - Schema Validation dialog box 618
 - XPath 623
 - Edit XML dialog box 613
 - exporting from object repository 243
 - importing as object repository 242
 - objects and methods 623
 - output value results
 - analyzing 1057
 - attribute details 1061
 - XML Checkpoint from File dialog box 600
 - XML Checkpoint Properties dialog box 607
 - XML checkpoint results
 - Element Value dialog box 1047
 - XML Checkpoint Results window 1038
 - XML Output Properties dialog box 727
 - XML Output Value Results window 1058
 - XML output values 672
 - XML structure
 - importing 614, 732
 - updating 614, 732
 - updating using Update Run mode 614, 732
 - XML values, outputting 718

Z

zip files

- exporting tests to 331

- importing tests from 331

zipping tests 331

