

HP QuickTest Professional

Software Version: 11.00

User Guide

Document Release Date: October 2010

Software Release Date: October 2010



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.

Copyright Notices

© 1992 - 2010 Hewlett-Packard Development Company, L.P.

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

The title page of this document 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

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 online support 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 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 register for an HP Passport ID, go to:

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

To find more information about access levels, go to:

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

Table of Contents

Welcome to the HP QuickTest Professional	
User Guide	33
HP QuickTest Professional User Guide Overview	33
How Do I Find the Information That I Need?	34
Documentation Library Contents.....	36
Additional Online Resources.....	40
PART I: INTRODUCING QUICKTEST PROFESSIONAL	
Chapter 1: QuickTest Introduction	43
QuickTest Professional Overview	44
Testing Process Overview	45
Testing Process Workflow	46
Additional Testing Capabilities and Tools	54
Integration with Other HP Products	55
QuickTest Program Management.....	59
Chapter 2: QuickTest at a Glance	63
Concepts	
QuickTest Main Window Overview	64
QuickTest Panes.....	69
Tasks	
How to Start QuickTest	78

Reference

QuickTest Main Window - User Interface.....	79
QuickTest Commands.....	82
QuickTest Professional Program Folder Structure.....	106
Start Page	111
Add-in Manager Dialog Box.....	113
About QuickTest Professional Dialog Box.....	117
Product Information Window	119
Troubleshooting and Limitations - QuickTest Program Management	121

PART II: WORKING WITH TEST OBJECTS

Chapter 3: The Test Object Model.....125

Concepts

Test Object Model - Overview	126
How QuickTest Applies the Test Object Model Concept.....	130
Object Repository Types - Overview	137
Deciding Whether to Use Local or Shared Object Repositories.....	140

Tasks

How to Use the Object Spy to View Object Properties and Operations or Add an Object to a Repository	144
---	-----

Reference

Comparison of Object Repository Window and Object Repository Manager	147
Object Identification Process Workflow	149
Object Spy Dialog Box.....	151
Object Selection Dialog Box.....	155
Tips for Using the Pointing Hand	157
Troubleshooting and Limitations - Object Spy.....	158

Chapter 4: Managing Test Objects in Object Repositories159

Concepts

Adding and Deleting Test Objects in a Local or Shared Object Repository.....	160
Guidelines for Copying, Pasting, and Moving Objects	163
Locating Objects.....	165
Maintaining Identification Properties - Overview	167
Visual Relation Identifiers	174

Tasks

How to Add a Test Object to an Object Repository	176
How to Copy, Paste, Move, or Delete Objects in the Object Repository.....	180
How to Locate an Object in an Object Repository	183
How to Maintain Test Objects in Object Repositories.....	184
How to Define a Visual Relation Identifier for a Specific Test Object - Use-Case Scenario	190

Reference

Add Properties Dialog Box.....	197
Define New Test Object Dialog Box	200
Define Object Filter Dialog Box	202
Find and Replace Dialog Box	206
Ordinal Identifier Dialog Box.....	208
Visual Relation Identifier Dialog Box.....	210
Troubleshooting and Limitations - Managing Test Objects	217

Chapter 5: Working with Your Test's Object Repositories219**Concepts**

Object Repository Window - Overview.....	220
Exporting Local Objects to a Shared Object Repository	221
Local Copies of Objects from Shared Object Repositories	222
Repository Parameter Value Mappings	223
Working with Test Objects During a Run Session	225

Tasks

How to Export Local Objects to a Shared Object Repository.....	226
How to Copy an Object to the Local Object Repository	227
How to Modify Identification Properties During a Run Session	228

Reference

Associate Repositories Dialog Box	229
Map Repository Parameters Dialog Box	232
Object Properties Dialog Box	234
Object Repository Window	237
Troubleshooting and Limitations - Object Repositories	247

Table of Contents

Chapter 6: Shared Object Repositories.....	249
Concepts	
Shared Object Repositories Overview	250
Considerations for Working with Shared Object Repositories	255
Tasks	
How to Manage Shared Object Repositories	258
How to Manage Objects in Shared Object Repositories.....	263
Reference	
Object Repository Manager Main Window	266
Manage Repository Parameters Dialog Box	278
Chapter 7: Configuring Object Identification	283
Concepts	
Object Identification Configuration - Overview.....	284
Smart Identification	292
Test Object Mapping for Unidentified or Custom Classes	299
Tasks	
How to Configure Object Identification for a Test Object Class	300
How to Manage Identification Properties of a Test Object Class	301
How to Map an Unidentified or Custom Class to a Standard Windows Class	303
Reference	
Object Identification Dialog Box	304
Object Mapping Dialog Box.....	310
Smart Identification Properties Dialog Box	312
Chapter 8: Object Repository Comparison Tool.....	315
Concepts	
Object Repository Comparison Tool Overview	316
Tasks	
How to Compare Two Object Repositories	318
Reference	
Object Repository Comparison Tool Main Window	320
New Comparison Dialog Box.....	335

Chapter 9: Object Repository Merge Tool.....337**Concepts**

Object Repository Merge Tool Overview	338
Object Conflicts.....	340

Tasks

How to Merge Two Shared Object Repositories.....	344
How to Update a Shared Object Repository From a Local Object Repository	347

Reference

New Merge Dialog Box	350
Object Repository Merge Tool Main Window	352
Settings Dialog Box (Object Repository Merge Tool).....	368
Object Repository Merge Tool - Multiple Merge Window.....	374
Update from Local Repository Dialog Box.....	378

PART III: DESIGNING TESTS**Chapter 10: Test Creation Overview.....383****Concepts**

Methodologies for Creating Tests	384
Enhancing Your Tests.....	387
Sample Test.....	389
Relative Paths in QuickTest.....	391
Portable Copies of Tests	395
Opening and Saving Tests with Locked Resources	396

Tasks

How to Perform File Operations on Test Files	398
--	-----

Table of Contents

Reference	
Open Test Dialog Box.....	401
Open <Resource> Dialog Box	406
Save Test Dialog Box	412
Save <Resource> Dialog Box.....	417
Save Test with Resources Dialog Box	422
Export to Zip File Dialog Box	426
Import from Zip File Dialog Box	427
Print Dialog Box	428
Troubleshooting and Limitations - Opening and Saving Testing Documents	430
Chapter 11: Test Creation - Keyword-Driven Methodology	435
Concepts	
Keyword-Driven Methodology - Overview	436
Test a Flight Application Using the Keyword-Driven Methodology - Use-Case Scenario	446
Tasks	
How to Create a Test Using the Keyword-Driven Methodology	455
Chapter 12: Test Creation - Recording Mechanism	463
Concepts	
Recording Tests - Overview	464
Recording Modes.....	467
Tasks	
How to Record a Test Using Normal Recording Mode	470
How to Record a Test Using Analog Recording	473
How to Record a Test Using Low Level Recording.....	474
Reference	
Analog Recording Settings Dialog Box	476
Record and Run Settings Dialog Box.....	479
Troubleshooting and Limitations - Recording Tests.....	481

Chapter 13: Keyword View	483
Concepts	
Keyword View Overview	484
Comments in the Keyword View	486
Conditional and Loop Statements in the Keyword View	486
Standard Steps After a Conditional or Loop Block	487
Tasks	
How to Add a Standard Step to Your Test.....	487
How to Add a Standard Step After a Conditional or Loop Block	493
How to Move an Action or Step	494
How to Delete a Step	496
How to Navigate in the Keyword View and Other Tips	497
How to Insert and Remove Breakpoints in the Keyword View	501
How to View Properties of Step Elements in the Keyword View	502
Reference	
Keyboard Shortcuts in the Keyword View	503
Keyword View User Interface	504
Columns Tab (Keyword View Options Dialog Box).....	508
Fonts and Colors Tab (Keyword View Options Dialog Box).....	511
Select Test Object Dialog Box.....	513
Password Encoder Tool.....	519
Troubleshooting and Limitations - Keyword View.....	520
Chapter 14: Actions.....	521
Concepts	
Actions Overview	523
Action Types.....	525
Action and Test Iterations Using the Data Table	526
Calls to Existing Actions and Copies of Actions.....	529
Action Parameters.....	531
Sharing Action Information	533
Action Syntax in the Expert View	535
Considerations for Working with Actions	538
Tasks	
How to Use Actions in Your Test	542
How to Nest Actions - Use-Case Scenario	547
How to Use Action Parameters - Use-Case Scenario	548

Table of Contents

Reference	
Action Call Properties Dialog Box	550
Action Properties Dialog Box	557
Action Toolbar in the Keyword View.....	576
Insert Call to New Action Dialog Box	578
Rename Action Dialog Box	580
Select Action Dialog Box	582
Split Action Dialog Box	585
Troubleshooting and Limitations - Actions	587
 PART IV: ENHANCING TESTS	
Chapter 15: Checkpoints Overview	591
Concepts	
Checkpoints Overview	592
Checkpoint Types.....	593
Tasks	
How to Insert a Checkpoint Step in a Test	598
Reference	
Add Existing Checkpoint Dialog Box	601
Troubleshooting and Limitations - Creating Checkpoints.....	603
Chapter 16: Standard Checkpoints.....	605
Concepts	
Standard Checkpoints Overview.....	606
Tasks	
How to Create or Modify a Standard Checkpoint Step	607
Reference	
Checkpoint Properties Dialog Box	609
Image Checkpoint Properties Dialog Box	615
Chapter 17: Bitmap Checkpoints	619
Concepts	
Bitmap Checkpoints Overview	620
Fine-Tuning the Bitmap Comparison	621

Tasks

How to Create or Modify a Bitmap Checkpoint Step 624

Reference

Bitmap Checkpoint Properties Dialog Box 626

Chapter 18: Table Checkpoints.....635**Concepts**

Table Checkpoints Overview 636

Tasks

How to Create or Modify a Table Checkpoint Step 637

Reference

Table Checkpoint Properties Dialog Box (Table Content Tab).....639

Table Checkpoint Properties Dialog Box (Properties Tab) 649

Define/Modify Row Range Dialog Box 652

Chapter 19: Text Checkpoints.....655**Concepts**

Checking Text Overview 656

Tasks

How to Create or Modify a Text or Text Area Checkpoint Step 657

Reference

Text / Text Area Checkpoint Properties Dialog Box 660

Chapter 20: Database Checkpoints.....673**Concepts**

Database Checkpoints Overview 674

Tasks

How to Create or Modify a Database Checkpoint Step 676

Table of Contents

Reference	
Database Checkpoint Properties Dialog Box	679
Connect to Database Using ODBC Page	
(Database Query Wizard)	689
Specify SQL Statement Page (Database Query Wizard).....	691
Troubleshooting and Limitations - Database Checkpoints	692
Chapter 21: XML Checkpoints	693
Concepts	
XML Checkpoints - Overview	694
XML Checkpoint Types.....	696
Using XML Objects and Methods to Enhance Your Test	697
Tasks	
How to Create or Modify an XML Checkpoint Step	698
How to Update the XML Hierarchy for XML Test Object Operation Checkpoints (WebService Test Objects Only)	700
Reference	
XML Checkpoint Properties Dialog Box	703
Edit XML as Text Dialog Box	710
XML Source Selection - Checkpoint / Output Value Properties Dialog Box.....	712
Schema Validation Dialog Box.....	716
Troubleshooting and Limitations - XML Checkpoints.....	722
Chapter 22: Parameterizing Values	723
Concepts	
Parameterizing Values Overview	725
Test and Action Input Parameters.....	727
Data Table Parameters.....	731
Environment Variable Parameters	734
When to Choose Global or Action Data Table Parameters	737
Automatically Parameterizing Steps.....	738
Data Driver	744
Example of a Parameterized Test.....	744

Tasks

How to Parameterize Values for Operations or Local Objects	750
How to Parameterize a Checkpoint Property Value.....	751
How to Use User-Defined External Environment Variables	753
How to Create an External Environment Variables File	755

Reference

Default Parameter Values	757
Built in Environment Variables.....	758
Parameter Options Dialog Box (Test/Action Parameter)	760
Parameter Options Dialog Box (Data Table)	763
Parameter Options Dialog Box (Environment)	766
Edit Complex Value Dialog Box.....	769
Parameter Options Dialog Box (Random Number)	770
Data Driver Dialog Box	773
Data Driver Wizard - Select Parameterization Type Page	776
Data Driver Wizard - Parameterize the Selected Step Page	778

Chapter 23: Output Values.....781**Concepts**

Output Values Overview	782
------------------------------	-----

Tasks

How to Create or Modify a Standard Output Value Step	790
How to Create or Modify a Table Output Value Step	793
How to Create or Modify a Text or Text Area Output Value Step....	794
How to Create or Modify a Database Output Value Step	797
How to Create or Modify an XML Output Value Step.....	798
How to Update the XML Hierarchy for XML Test Object Operation Output Value Steps (For WebService Test Objects Only)	801

Reference

Output Value Properties Dialog Box	803
Table Output Value Properties Dialog Box (Table Content Tab).....	812
Table Output Value Properties Dialog Box (Properties Tab).....	819
Text / Text Area Output Value Properties Dialog Box	822
Database Output Value Properties Dialog Box.....	830
XML Output Properties Dialog Box	835
Add Existing Output Value Dialog Box.....	841

Table of Contents

Chapter 24: Text Recognition for Windows-Based Objects	847
Concepts	
Text Recognition for Windows-Based Objects Overview	848
Checking Text in an Image - Use-Case Scenario.....	848
Tasks	
How to Configure Text Recognition Settings	852
Reference	
Guidelines for Text Recognition	854
Text Recognition and Development Environments.....	857
Chapter 25: Value Configuration and Regular Expressions	861
Concepts	
Value Configuration Overview	862
Regular Expressions Overview.....	863
Tasks	
How to Configure Constant and Parameter Values.....	866
Reference	
Configure Value Area	867
Constant Value Options Dialog Box.....	870
Value Configuration Options Dialog Box.....	872
Regular Expression Characters and Usage Options	875
Regular Expression Evaluator	882
Smart Regular Expression List	884
Chapter 26: User Interface-Based Programming Operations	887
Concepts	
Programming Statements Overview.....	888
Test Synchronization.....	894
Message Statements.....	897
Tasks	
How to Insert Steps Using the Step Generator.....	900
How to Insert Conditional Statements from the Keyword View.....	901
How to Insert Loop Statements In the Keyword View	904
How to Generate With Statements for Your Test	906

Reference

Add Synchronization Point Dialog Box	910
Comment Properties Dialog Box.....	912
Insert Comment Dialog Box	912
Insert Report Dialog Box	914
Step Generator Dialog Box	916
Storage Location Options Dialog Box	924

PART V: DEFINING FUNCTIONS AND OTHER PROGRAMMING TASKS**Chapter 27: Working in the Expert View and
Function Library Windows.....929****Concepts**

The Expert View and Function Library Windows Overview	931
Expert View and Keyword View - A Comparison	932
Generating Statements in the Expert View or in a Function Library	934
Bookmarks in an Action or Function Library	945
Programmatic Descriptions	946
Opening and Closing Applications Programmatically	959
Comments, Control-Flow, and Other VBScript Statements	960
Retrieving and Setting Identification Property Values	961
Native Properties and Operations	962
Running DOS Commands.....	964
Choosing Which Steps to Report During the Run Session	964
Windows API	964

Tasks

How to Navigate in the Expert View and Function Libraries	965
How to Enhance Your Tests and Function Libraries Using the Windows API.....	966

Reference

Checkpoint and Output Statements	969
Basic VBScript Syntax	970
Report Modes.....	987
Expert View and Function Library Window User Interface.....	988

Table of Contents

Chapter 28: Customizing the Expert View and Function Library Windows.....	1001
Concepts	
Expert View and Function Library Window Customization Options	1002
Reference	
General Tab (Editor Options Dialog Box)	1004
Fonts and Colors Tab (Editor Options Dialog Box)	1007
Key Binding Tab (Editor Options Dialog Box).....	1009
Chapter 29: User-Defined Functions and Function Libraries	1011
Concepts	
Function Library Overview	1013
Associated Function Libraries.....	1015
User-Defined Functions.....	1017
User-Defined Function Storage and Access.....	1019
User-Defined Function Registration.....	1020
Executing Externally-Defined Functions from Your Test	1026
Tasks	
How to Manage Function Libraries.....	1028
How to Edit a Function Library.....	1033
How to Manage Function Library Associations	1036
How to Work with a User-Defined Function	1039
How to Create and Register a User-Defined Function Using the Function Definition Generator.....	1042
How to Execute an Externally-Defined Function from Your Test	1048
Reference	
Function Definition Generator Dialog Box	1050
Troubleshooting and Limitations - Function Libraries.....	1059

PART VI: RUNNING AND ANALYZING TESTS

Chapter 30: QuickTest Run Sessions	1063
Concepts	
Run Sessions - Overview	1064
Optional Steps	1066
Tasks	
How to Run Your Test	1067
How to Set Optional Steps	1071
Reference	
Default Optional Steps	1072
Run Dialog Box: Results Location Tab (For Tests Stored in the File System)	1073
Run Dialog Box: Results Location Tab (For Tests Stored in Quality Center)	1075
Run Dialog Box: Input Parameters Tab	1078
Test Batch Runner	1080
Troubleshooting and Limitations - Run Sessions	1082
Chapter 31: Run Results Viewer	1085
Concepts	
Run Results Viewer Overview	1086
Run Results File Location	1091
Tasks	
How to Install the Run Results Viewer as a Standalone Application	1092
How to Open Run Results	1093
How to Navigate the Run Results Tree	1094
How to Customize the Run Results Viewer	1096
How to Jump to a Step in QuickTest	1097
How to Manually Submit Defects to Quality Center	1098
How to Automatically Submit Defects to a Quality Center Project	1100
How to Export Run Results	1102
How to Play a Screen Recorder Movie in the HP Micro Player	1104
How to Delete Run Results	1105

Table of Contents

Reference	
Run Results Viewer User Interface	1107
Run Results Viewer Commands	1109
Run Results Viewer Panes.....	1113
Run Results Viewer Dialog Boxes	1140
Run Results Deletion Tool.....	1160
Troubleshooting and Limitations - Viewing Run Results	1168
Chapter 32: Run Results - Understanding Step Results.....	1169
Concepts	
Smart Identification in the Run Results.....	1170
Checkpoint and Output Value Results	1174
Parameterized Values in the Run Results	1188
QuickTest Tests Containing Calls to Service Test Tests	1190
Reference	
XML Checkpoint Results Window (Run Results Viewer)	1191
XML Output Value Results Window (Run Results Viewer)	1199
PART VII: MAINTAINING AND DEBUGGING TESTS	
Chapter 33: Debugging Tests and Function Libraries.....	1205
Concepts	
Debugging Overview	1206
Considerations for Debugging	1207
Debug Session Speed	1208
Single Step Commands	1208
Running to a Step and Debugging from a Step	1209
Modifying and Watching the Values of Variables and Properties of Objects During a Run Session	1212
Breakpoints.....	1213
Run Errors	1214
Tasks	
How to Debug Your Test or Function Library.....	1215
How to Use Breakpoints	1219
How to Debug an Action or a Function - Exercise.....	1220
How to Step Into, Out of, or Over a Specific Step - Exercise	1224

Reference

Debug Viewer Pane.....	1227
Watch Tab (Debug Viewer Pane)	1229
Variables Tab (Debug Viewer Pane)	1232
Command Tab (Debug Viewer Pane).....	1234
Run Error Message Box.....	1236
Troubleshooting and Limitations - Debugging.....	1238

Chapter 34: Maintaining and Updating Tests	1239
---	-------------

Concepts

Why Tests Fail	1240
Maintenance Run Mode.....	1242
Updating Your Test Object Descriptions, Checkpoints, Output Values, or Active Screen Captures.....	1244

Tasks

How to Use Maintenance Run Mode to Update Your Test When Your Application Changes.....	1248
How to Update Test Object Descriptions, Checkpoints, Output Values, or Active Screens	1252

Reference

Maintenance Run Wizard.....	1255
Update Options Tab (Update Run Dialog Box)	1271
Troubleshooting and Limitations - Maintenance Mode.....	1274

PART VIII: WORKING WITH THE QUICKTEST PANES

Chapter 35: QuickTest Window Layout	1279
--	-------------

Concepts

QuickTest Window Layout Customization - Overview	1280
--	------

Tasks

How to Customize the QuickTest Window	1281
---	------

Table of Contents

Reference

Tips and Considerations for Customizing the QuickTest Window Layout	1291
Button Appearance Dialog Box	1294
Customization Mode - Context Menu Options.....	1296
Customize Dialog Box	1297
Windows Dialog Box.....	1304
Troubleshooting and Limitations - QuickTest Window Layout....	1305

Chapter 36: Active Screen Pane.....1307

Concepts

Active Screen Overview	1308
------------------------------	------

Tasks

How to Use the Active Screen in Your Test.....	1312
How to Modify the Active Screen Settings.....	1314

Reference

Active Screen Pane User Interface	1316
---	------

Chapter 37: Available Keywords Pane1319

Concepts

Available Keywords Pane Overview	1320
--	------

Tasks

How to Work with the Available Keywords Pane	1321
--	------

Reference

Available Keywords Pane User Interface	1322
--	------

Chapter 38: Data Table Pane1325

Concepts

Data Table Overview	1326
---------------------------	------

Data Table Sheets	1327
-------------------------	------

Data Table - Save Options	1330
---------------------------------	------

Data Table Objects, Methods, and Properties	1331
---	------

Formulas in Data Tables	1332
-------------------------------	------

Tasks

How to Define a Data Table in Your Test	1334
How to Manage Data Tables in Your Test.....	1336
How to Insert Formulas into Data Tables for Use in Checkpoints	1338

How to Import Data Using Microsoft Query	1339
--	------

Reference

Data Table Pane User Interface	1340
Database Query Wizard	1351
Troubleshooting and Limitations - Data Table.....	1354

Chapter 39: Information Pane1357**Concepts**

Information Pane Overview	1358
---------------------------------	------

Tasks

How to Resolve VBScript Syntax Errors in the Information Pane	1359
--	------

Reference

Information Pane User Interface.....	1360
--------------------------------------	------

Chapter 40: Missing Resources Pane1363**Concepts**

Missing Resources Overview	1364
----------------------------------	------

Tasks

How to Handle a Missing Resource.....	1367
How to Locate a Missing Action	1368

Reference

Missing Resources Pane User Interface.....	1371
--	------

Chapter 41: Process Guidance Panes.....1375**Concepts**

Process Guidance Overview	1376
---------------------------------	------

Tasks

How to Manage Process Guidance	1377
--------------------------------------	------

Table of Contents

Reference	
Process Guidance Management Dialog Box.....	1379
Process Guidance Panes User Interface	1380
Chapter 42: Resources Pane.....	1385
Concepts	
Resources Pane Overview	1386
Tasks	
How to Modify Associations Between Resources and Your Test or Action	1386
Reference	
Resources Pane User Interface	1391
Chapter 43: Test Flow Pane	1395
Concepts	
Test Flow Pane Overview.....	1396
Tasks	
How to Manage Actions in the Test Flow Pane	1397
Reference	
Test Flow Pane User Interface	1401
Chapter 44: To Do Pane	1405
Concepts	
To Do Pane Overview.....	1406
Tasks	
How to Manage Tasks and TODO Comments.....	1407
Reference	
Task Editor Dialog Box	1408
To Do Pane User Interface.....	1410

PART IX: CONFIGURING QUICKTEST SETTINGS**Chapter 45: Global Testing Options1419****Concepts**

Global Testing Options Overview1420

Reference

Options Dialog Box1420

General Pane (Options Dialog Box)1422

Folders Pane (Options Dialog Box)1431

Active Screen Pane (Options Dialog Box)1434

Run Pane (Options Dialog Box)1447

Chapter 46: Individual Test Settings.....1455**Concepts**

Test Settings Overview1456

Add-in Associations in Your Test1458

Associated Function Libraries.....1459

Local System Monitor.....1460

Log Tracking1461

Tasks

How to Manually Configure Log Tracking Settings.....1462

Reference

Properties Pane (Test Settings Dialog Box).....1466

Run Pane (Test Settings Dialog Box)1471

Resources Pane (Test Settings Dialog Box)1475

Parameters Pane (Test Settings Dialog Box)1479

Environment Pane (Test Settings Dialog Box)1483

Recovery Pane (Test Settings Dialog Box)1490

Local System Monitor Pane (Test Settings Dialog Box)1495

Log Tracking Pane (Test Settings Dialog Box).....1498

Chapter 47: Setting Testing Options During the Run Session.....1503**Concepts**

Setting Testing Options During the Run Session - Overview1504

Tasks

How to Set Testing Options During a Run Session1506

PART X: WORKING WITH ADVANCED TESTING FEATURES

Chapter 48: Virtual Objects.....	1513
Concepts	
Virtual Objects - Overview	1514
How Virtual Objects are Defined and Recognized.....	1515
Tasks	
How to Define Virtual Objects for Unsupported Objects in Your Test	1516
Reference	
Virtual Object Manager Dialog Box	1518
Virtual Object Wizard	1519
Chapter 49: Recovery Scenarios.....	1523
Concepts	
Recovery Scenarios Overview.....	1524
Tasks	
How to Create and Manage Recovery Scenarios.....	1528
How to Manage Recovery Scenario Associations.....	1530
Reference	
Add Recovery Scenario Dialog Box	1532
Recovery Scenario Manager Dialog Box.....	1534
Recovery Scenario Properties Dialog Box.....	1537
Recovery Scenario Wizard	1539
Troubleshooting and Limitations - Recovery Scenarios	1569
Chapter 50: QuickTest Script Editor	1571
Concepts	
QuickTest Script Editor Overview	1572
Reference	
QuickTest Script Editor Window.....	1575
Flow Pane (Script Editor).....	1582
Resources Pane (Script Editor).....	1584
Customize Dialog Box (Script Editor)	1587

Chapter 51: QuickTest Automation Scripts1589**Concepts**

QuickTest Automation Object Model Overview.....	1590
When to Use QuickTest Automation Scripts	1592
Application Object	1593
QuickTest Automation Object Model Reference.....	1594
Generated Automation Scripts	1594

Tasks

How to Create a QuickTest Automation Script.....	1596
How to Run Automation Scripts on a Remote Computer	1600

PART XI: WORKING WITH QUALITY CENTER**Chapter 52: Quality Center Integration.....1605****Concepts**

Quality Center Integration Overview	1607
Template Tests.....	1610
Quality Center Test Run Preferences.....	1611
QuickTest Remote Agent Preferences.....	1613
Data Awareness in HP ALM.....	1613

Tasks

How to Work with Tests in Quality Center	1618
How to Data Drive a Test in HP ALM	1621
How to Enable Quality Center to Run Tests on a QuickTest Computer.....	1627
How to Use the Quality Center Connectivity Add-in	1628
How to Create a Template Test	1629
How to Create a Test in Quality Center Using a Template Test	1631
How to View or Modify Remote Agent Settings	1633

Reference

HP ALM Data Awareness - Task Breakdown.....	1634
HP ALM Connection Dialog Box	1635
Remote Agent Settings Dialog Box.....	1642
Troubleshooting and Limitations - Quality Center Integration	1648

Table of Contents

Chapter 53: Resources and Dependencies Model	1651
Concepts	
Resources and Dependencies Model Overview.....	1652
Asset Dependencies - Advantages	1653
Reference	
Relative Paths and Quality Center	1656
Resources and Dependencies Model Terminology	1657
Quality Center Resources-Related User Interface.....	1659
Troubleshooting and Limitations -	
Resources and Dependencies	1666
Chapter 54: Viewing and Comparing Versions of QuickTest Assets	1669
Concepts	
Asset Comparison Tool and Asset Viewer - Overview	1670
Tasks	
How to Open the QuickTest Asset Comparison Tool	1673
How to Open the QuickTest Asset Viewer	1676
How to Work with the Asset Comparison Tool and Asset Viewer	1679
Reference	
Asset Comparison Tool	1681
Asset Viewer.....	1690
Troubleshooting and Limitations - Asset Comparison Tool.....	1693
Chapter 55: Version Control in Quality Center 10.00 or HP ALM	1695
Concepts	
Managing Versions of Assets in Quality Center Overview	1696
Tasks	
How to Check In the Currently Open Asset	1703
How to Check Out the Latest Version of an Asset.....	1704
How to Cancel a Check-Out Operation	1705

Reference

Version Management Commands	1706
Check Out Dialog Box.....	1707
Check In Dialog Box	1708
Version History Dialog Box.....	1709
Baseline History Dialog Box	1712
Troubleshooting and Limitations - Quality Center	
Version Control	1715

Chapter 56: Version Control in Quality Center 9.21717**Concepts**

Version Control for QuickTest Tests in Quality Center 9.2	
Overview	1718

Tasks

How to Add Tests to the Quality Center 9.2 Version Control	
Database	1721
How to Perform a Version Control Operation on Tests Stored in	
Quality Center 9.2	1722

Reference

Quality Center 9.2 Check In Dialog Box	1724
Quality Center 9.2 Check Out Dialog Box.....	1725
Quality Center 9.2 Version History Dialog Box.....	1726

Chapter 57: HP ALM Sprinter1729**Concepts**

HP ALM Sprinter Overview	1730
--------------------------------	------

PART XII: WORKING WITH OTHER HP PRODUCTS**Chapter 58: Service Test Integration1735****Concepts**

Service Test Integration Overview.....	1736
--	------

Tasks

How to Integrate with Service Test	1739
--	------

Reference

Call to Service Test Test Dialog Box	1741
--	------

Table of Contents

Chapter 59: Business Process Testing	1743
Concepts	
Business Process Testing Overview.....	1743
Business Process Testing Workflow.....	1748
Business Process Testing Methodology	1750
Reference	
Quality Center Business Components Module.....	1755
Chapter 60: HP Performance Testing and Business Service Management Products	1757
Concepts	
HP Performance Testing and Business Service Management Products Overview	1758
Designing Tests for HP Performance Testing Products	1761
Running Tests from HP Performance Testing Products	1762
Designing Tests for HP Business Process Monitor	1763
Running Tests from HP Business Process Monitor	1764
Measuring Transactions	1765
Silent Test Runner	1768
Tasks	
How to Insert and Run Tests in Performance Center and LoadRunner.....	1770
Reference	
End Transaction Dialog Box.....	1771
Start Transaction Dialog Box.....	1772
Silent Test Runner Dialog Box	1774
PART XIII: APPENDICES	
Appendix A: Naming Conventions	1779
Appendix B: Supported Checkpoints and Output Values Per Add-in	1783
Supported Checkpoints.....	1784
Supported Output Values	1786

Appendix C: Frequently Asked Questions.....	1789
Creating Tests	1790
Programming in the Expert View.....	1792
Working with Dynamic Content	1794
Advanced Web Issues	1796
Standard Windows Environment.....	1800
Test Maintenance	1802
Testing Localized Applications.....	1805
Improving QuickTest Performance	1806
Appendix D: Custom Process Guidance Packages.....	1811
Concepts	
Custom Process Guidance Packages - Overview	1812
Data Files for Process Guidance Packages	1813
Tasks	
How to Create a Custom Package Configuration File.....	1814
How to Install Custom Process Guidance Packages in QuickTest	1815
Reference	
XML Details for Custom Process Guidance Packages	1816
Appendix E: Bitmap Checkpoint Customization	1819
Concepts	
About Bitmap Checkpoint Customization	1820
Custom Comparer for Images Whose Location Changes in the Application - Use-Case Scenario	1821
Custom Bitmap Comparer Development	1823
Tasks	
How to Develop a Custom Comparer	1824
How to Implement the Bitmap Comparer Interfaces	1828
How to Install Your Custom Comparer and Register it to QuickTest	1832
How to Use the Bitmap Checkpoint Customization Samples	1836
How to Develop a Custom Comparer - Tutorial.....	1839
Reference	
The Bitmap Checkpoint Comparer Interfaces	1851

Table of Contents

Welcome to the HP QuickTest Professional User Guide

This chapter includes:

- "HP QuickTest Professional User Guide Overview" on page 33
- "How Do I Find the Information That I Need?" on page 34
- "Documentation Library Contents" on page 36
- "Additional Online Resources" on page 40

HP QuickTest Professional User Guide Overview

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.

Prerequisite Background

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.

How Do I Find the Information That I Need?

Within this guide, each subject area is organized into topics. A topic contains a distinct module of information for that subject. The topics are generally classified according to the type of information they contain.

This structure is designed to create easier access to specific information by dividing the documentation into the different types of information you may need at different times.

Three main topic types are in use: **Concepts**, **Tasks**, and **Reference**. The topic types are differentiated visually using icons.

Topic Types

Topic Type	Description	Usage
Concepts 	General Concepts. Background, descriptive, or conceptual information.	Learn general information about what a feature does.
	Use-case Scenario Concepts. Real-life examples of when or why to use a specific product area.	Learn why or when you may want to use the feature.

Topic Type	Description	Usage
Tasks 	<p>Instructional Tasks. Step-by-step guidance to help you work with the application and accomplish your goals. Some task steps include examples, using sample data.</p> <p>Task steps can be with or without numbering:</p> <ul style="list-style-type: none"> ➤ Numbered steps. Tasks that are performed by following each step in consecutive order. ➤ Non-numbered steps. A list of self-contained operations that you can perform in any order. 	<ul style="list-style-type: none"> ➤ Learn about the overall workflow of a task. ➤ Follow the steps listed in a numbered task to complete a task. ➤ Perform independent operations by completing steps in a non-numbered task.
	<p>Exercise Tasks. Step-by-step instructions for a task using a sample application or sample data.</p>	Follow the steps in these topics to practice the workflow of a task.
	<p>Use-case Scenario Tasks. Examples of how to perform a task for a specific situation.</p>	Learn how a task could be performed in a realistic scenario.

Topic Type	Description	Usage
 Reference	General Reference. Detailed lists and explanations of reference-oriented material.	Look up a specific piece of reference information relevant to a particular context.
	User Interface Reference. Specialized reference topics that describe a particular user interface in detail. Pressing F1 in the product area generally open the user interface topics.	Look up specific information about what to enter or how to use one or more specific user interface elements, such as a window, dialog box, or wizard.
 Troubleshooting and Limitations	Troubleshooting and Limitations. Specialized reference topics that describe commonly encountered problems and their solutions, and list limitations of a feature or product area.	Increase your awareness of important issues before working with a feature, or if you encounter usability problems in the software.

Documentation Library Contents

This guide is part of the QuickTest Professional Documentation Library. The Documentation Library provides a single-point of access for all QuickTest Professional documentation.

You can access the Documentation Library by using the following:

- Select **Help > QuickTest Professional Help**.
- In the Start menu, select **Program Files > HP QuickTest Professional > Documentation > HP QuickTest Professional Help**.
- Click in selected QuickTest windows and dialog boxes or press F1.
- View a description, syntax, and examples for a QuickTest test object, method, or property by placing the cursor on it and pressing F1.

The Documentation Library includes the following:

Type	Included Documentation
Getting Started Documentation	<ul style="list-style-type: none">➤ Readme provides the latest news and information about QuickTest. Select Start > Programs > HP Mercury Product > Readme.➤ HP Mercury Product Installation Guide explains how to install and set up QuickTest. Select Help > QuickTest Professional Help and click the link to the Installation Guide from the Documentation Library Home page.➤ HP Mercury Product Tutorial teaches you basic QuickTest skills and shows you how to design tests for your applications. Select Help > Mercury Product Tutorial.➤ Product Feature Movies provide an overview and step-by-step instructions describing how to use selected QuickTest features. Select Help > Product Feature Movies.➤ What's New provides an overview of the features, enhancements and supported environments that are new in the current version of <i>QuickTest</i>. Choose Help > What's New.

Type	Included Documentation
Feature Documentation	<p>Mercury Product Help includes:</p> <ul style="list-style-type: none">➤ Home provides links to the Documentation Library guides in each available format (Help, PDF, and/or HTML).➤ What's New in QuickTest Professional describes the newest features, enhancements, and supported environments in the latest version of QuickTest.➤ HP Mercury Product User Guide describes how to use QuickTest to test your application.➤ HP Mercury Product 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 Mercury Product Add-ins Guide describes how to work with supported environments using QuickTest add-ins, and provides environment-specific information for each add-in.➤ HP Mercury Product 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.

Type	Included Documentation
Reference Documentation	<p>► HP Mercury Product Advanced References contains documentation for the following QuickTest COM and XML references:</p> <ul style="list-style-type: none"> ► 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 Run Results Schema documents the run results XML schema, which provides the information you need to customize your run 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.

Additional Online Resources

Sample applications. The following sample applications are the basis for many examples in this guide:

- **Mercury Tours sample Web site.** The URL for this Web site is newtours.demoaut.com.
- **Mercury Flight application.** To access from the Start menu, select **Program Files > HP QuickTest Professional > Sample Applications > Flight.**

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>

HP Software Web site accesses the HP Software Web site. This 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. Choose **Help > HP Software Web site**. The URL for this Web site is www.hp.com/go/software.

Part I

Introducing QuickTest Professional

1

QuickTest Introduction

This chapter includes:

- QuickTest Professional Overview on page 44
- Testing Process Overview on page 45
- Testing Process Workflow on page 46
- Additional Testing Capabilities and Tools on page 54
- Integration with Other HP Products on page 55
- QuickTest Program Management on page 59

QuickTest Professional Overview

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.

QuickTest Professional provides the following functionality:

- 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. For details, see "Testing Process Workflow" on page 46.
- 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. For details, see "Integration with Other HP Products" on page 55.
- QuickTest Professional provides add-ins that enable you to test objects (controls) created in commonly used development environments. For details, see "Additional Testing Capabilities and Tools" on page 54.



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 "Troubleshooting and Limitations - Multilingual Applications" on page 1206.



Testing Process Overview

Basic Test Structure

- 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.
- 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.

Creating and Enhancing Your Test

- You can use QuickTest process guidance to guide you through the process of creating a test. For more information, see "Process Guidance Panes" on page 1375.
- 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 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.

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 "QuickTest Commands" on page 82.

Testing Process Workflow

This section describes the general workflow of the QuickTest testing process.

Testing with QuickTest involves the following main stages:

- "Stage 1: Analyzing Your Application" on page 47
- "Stage 2: Preparing the Testing Infrastructure" on page 47
- "Stage 3: Adding Steps to Your Actions" on page 48
- "Stage 4: Enhancing Your Test" on page 50
- "Stage 5: Running and Debugging Your Test" on page 51
- "Stage 6: Analyzing Run Results and Reporting Defects" on page 52

Stage 1: Analyzing Your Application

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

- **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.
- **Determine the functionality that you want to test.** To do this, you 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?
- **Decide how to divide these processes into smaller units that 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 4, "Managing Test Objects in Object Repositories" and Chapter 29, "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 51, "QuickTest Automation Scripts."

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 47.)

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 13, "Keyword View" and Chapter 37, "Available Keywords Pane." Advanced users can also add steps using the Expert View. For more information, see Chapter 27, "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 13, "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 15, "Checkpoints Overview."
- 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 23, "Output 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 22, "Parameterizing Values."
- Add user-defined functions by creating function libraries and calling their functions from your test. For more information, see Chapter 29, "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 26, "User Interface-Based Programming Operations."

Stage 5: Running and Debugging Your Test

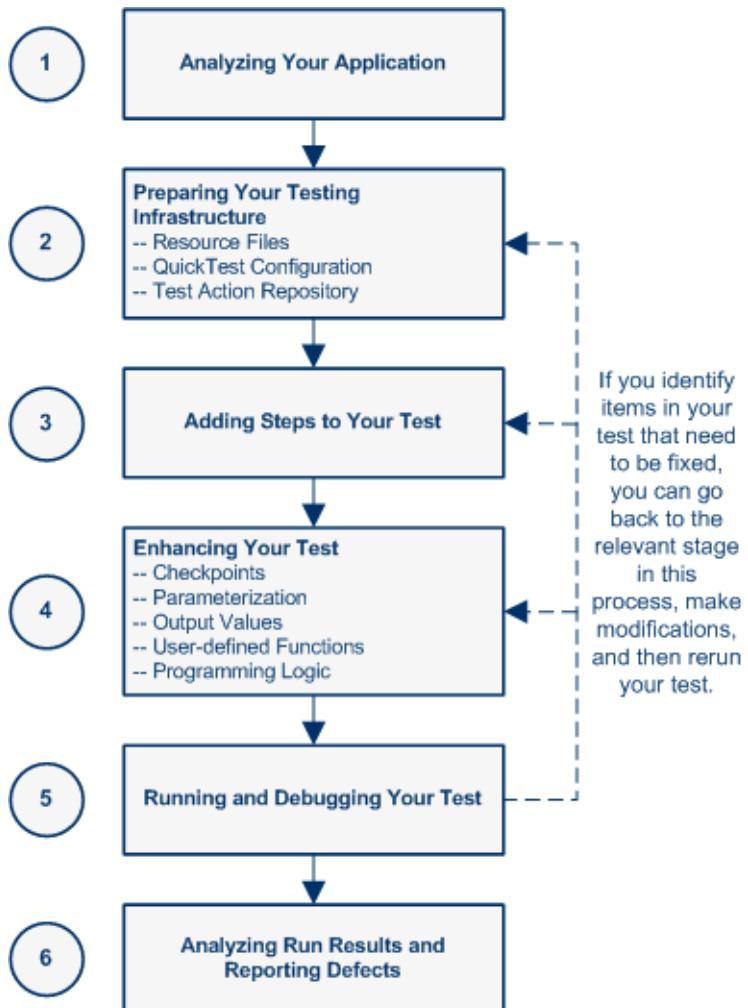
After you create your test, you can perform different types of runs to achieve different goals.

- **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 33, "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 30, "QuickTest Run Sessions."
- **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 34, "Maintaining and Updating 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 Run Results and Reporting Defects

After you run your test, you can view the results of the run in the Run Results Viewer. 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 Run Results Viewer. For more information, see Chapter 31, "Run Results Viewer." If you enabled local system monitoring for your test, you can view the results in the System Monitor tab of the Run Results Viewer. For more information, see "Run Results - Understanding Step Results" on page 1169.

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 Run Results Viewer. For more information, see Chapter 52, "Quality Center Integration Overview."



Additional Testing Capabilities and Tools

The following sections describe additional QuickTest Professional functionality:

- "Testing Objects from Different Environments" on page 54
- "Programming in the Expert View" on page 54
- "Functions and Function Libraries" on page 55

Testing Objects from Different Environments

- When you open QuickTest, you can load environment-specific QuickTest add-ins, such as Java, .NET, and Web.
- 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.
- You load add-ins using the Add-in Manager dialog box described in "How to Start QuickTest" on page 78. 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*.

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 27, "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).

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 29, "User-Defined Functions and Function Libraries."



Integration with Other HP Products

You can integrate QuickTest Professional with the following HP products:

- "Quality Center" on page 56
- "Business Process Testing" on page 57
- "Service Test" on page 58

Quality Center

Note: Unless otherwise specified, references to **Quality Center** in this guide apply to all currently supported versions of Quality Center and HP ALM. Note that some features and options may not be supported in the specific edition of Quality Center or HP ALM that you are using.

For a list of the supported versions of Quality Center or HP ALM, see the *HP QuickTest Professional Product Availability Matrix*, available from the Documentation Library Home page or the root folder of the QuickTest Professional DVD.

For details on Quality Center or HP ALM editions, see the *HP Quality Center User Guide* or the *HP Application Lifecycle Management User Guide*.

- 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 52, "Quality Center Integration." 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 Quality Center or HP ALM user guide.
- 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 59, "Business Process Testing."

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*.

Service Test

Service Test is HP's functional testing tool for GUI-less applications. You can use Service Test to test standard Web Services, non-SOAP Web Services, such as REST, and so on.

From QuickTest, you can insert a call to a Service Test test (and vice versa). This enables you to test both GUI and GUI-less applications using one comprehensive test.

You can integrate Service Test with QuickTest if:

- Service Test 11.00 is installed on the QuickTest computer, and
- QuickTest is using an HP Unified Functional Testing (also known as UnifiedFunctionalTesting) license.

You can also integrate QuickTest and Service Test with Quality Center if your tests are stored in a Quality Center project.

For details, see "Service Test Integration" on page 1735.

QuickTest Program Management

The following sections describe different operations you can perform to manage your QuickTest Professional program, set user permissions, and update license information.

This section includes:

- "Access Permissions" on page 59
- "Mercury Tours Sample Site" on page 60
- "License Information" on page 60
- "QuickTest Software Updates" on page 61

Access Permissions

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

Permissions Required When Working with 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

Mercury Tours 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 > HP QuickTest Professional > Sample Applications > Flight**.

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 (includes a 30-day demo license)
- 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*.

 **QuickTest Software Updates**

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 update took place more than seven days ago, and if so, it checks for updates.

You can also manually check for updates to all HP products installed on your computer at any time by choosing **Help > HP Update** from within QuickTest, or by choosing **Start > Programs > HP QuickTest Professional > HP Update**. (You must be logged on with Administrator privileges to use HP Update.)

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 includes:

Concepts

- QuickTest Main Window Overview on page 64
- QuickTest Panes on page 69

Tasks

- How to Start QuickTest on page 78

Reference

- QuickTest Main Window - User Interface on page 79
- QuickTest Commands on page 82
- QuickTest Professional Program Folder Structure on page 106
- Start Page on page 111
- Add-in Manager Dialog Box on page 113
- About QuickTest Professional Dialog Box on page 117
- Product Information Window on page 119

Troubleshooting and Limitations - QuickTest Program Management
on page 121

Concepts

QuickTest Main Window Overview

The following table introduces the key elements in the QuickTest window:

Element	Description
Keyword View	Contains each step, and displays the object hierarchy, in a modular, icon-based table. For details, see "Keyword View at a Glance" on page 66.
Expert View	Contains each step as a VBScript line. In object-based steps, the VBScript line defines the object hierarchy. For details, see "Expert View at a Glance" on page 67.
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 details, see "Start Page" on page 111.
QuickTest Panes	Provide information and functionality about the currently open test or function library. For details, see "QuickTest Panes" on page 69.

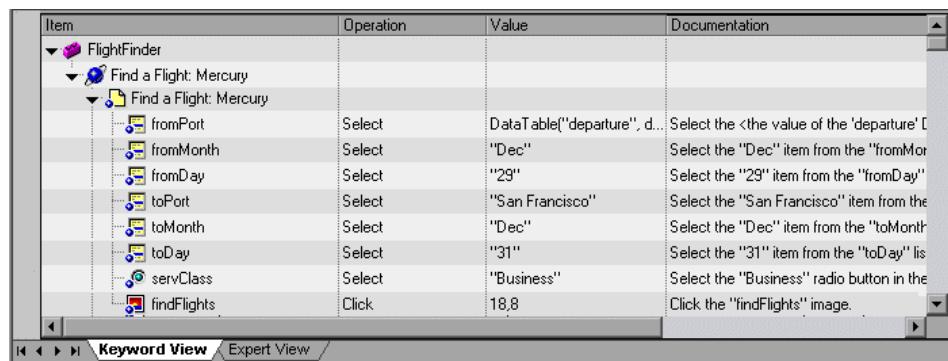
The following document types are available in QuickTest Professional:

Document	Description
Test	<p>Enables you to create, view and modify tests containing actions, steps, and resources.</p> <p>For details, see "Test Creation Overview" on page 383.</p> <p>Note: QuickTest also enables you to create, view, and modify business components, scripted components, and application areas. For details, see the <i>HP QuickTest Professional for Business Process Testing User Guide</i>.</p>
Function Library	<p>Enables you to create, view, and modify functions and subroutines for use with your test.</p> <p>For details, see "Function Libraries at a Glance" on page 68.</p>

Keyword View at a Glance

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.



The screenshot shows the Microsoft Test Manager interface with the 'Keyword View' tab selected. A tree view on the left shows a test case named 'FlightFinder' containing a single step 'Find a Flight: Mercury'. This step has several sub-steps under 'Find a Flight: Mercury', including 'fromPort', 'fromMonth', 'fromDay', 'toPort', 'toMonth', 'toDay', 'servClass', and 'findFlights'. Each sub-step is listed in a table with four columns: Item, Operation, Value, and Documentation. The 'Documentation' column provides a brief description of the operation, such as selecting specific values from dropdown menus or clicking radio buttons.

Item	Operation	Value	Documentation
FlightFinder			
Find a Flight: Mercury			
Find a Flight: Mercury	Select	DataTable("departure", d...	Select the <the value of the 'departure' [
fromPort	Select	"Dec"	Select the "Dec" item from the "fromMor
fromMonth	Select	"29"	Select the "29" item from the "fromDay"
fromDay	Select	"San Francisco"	Select the "San Francisco" item from the
toPort	Select	"Dec"	Select the "Dec" item from the "toMonth"
toMonth	Select	"31"	Select the "31" item from the "toDay" lis
toDay	Select	"Business"	Select the "Business" radio button in the
servClass	Select	18.8	Click the "findFlights" image.
findFlights	Click		

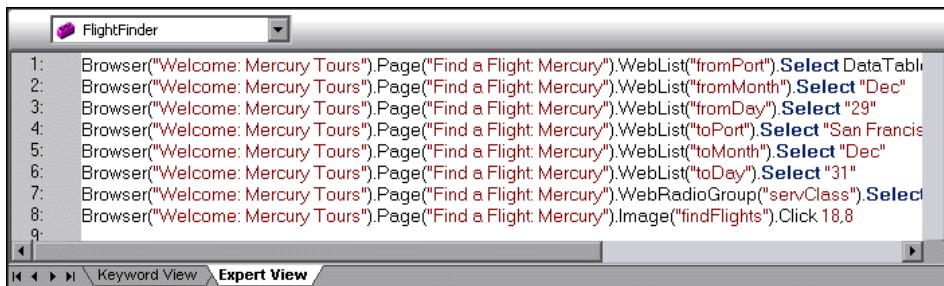
Each operation performed on your application during a recording session is recorded as a row in the Keyword View.

For each row in the Keyword View, QuickTest displays a corresponding line of script in the Expert View. If you focus on a specific step in the Keyword View and switch to the Expert View, the cursor is located in that corresponding line of the test.

For details on using the Keyword View, see Chapter 13, "Keyword View."

Expert View at a Glance

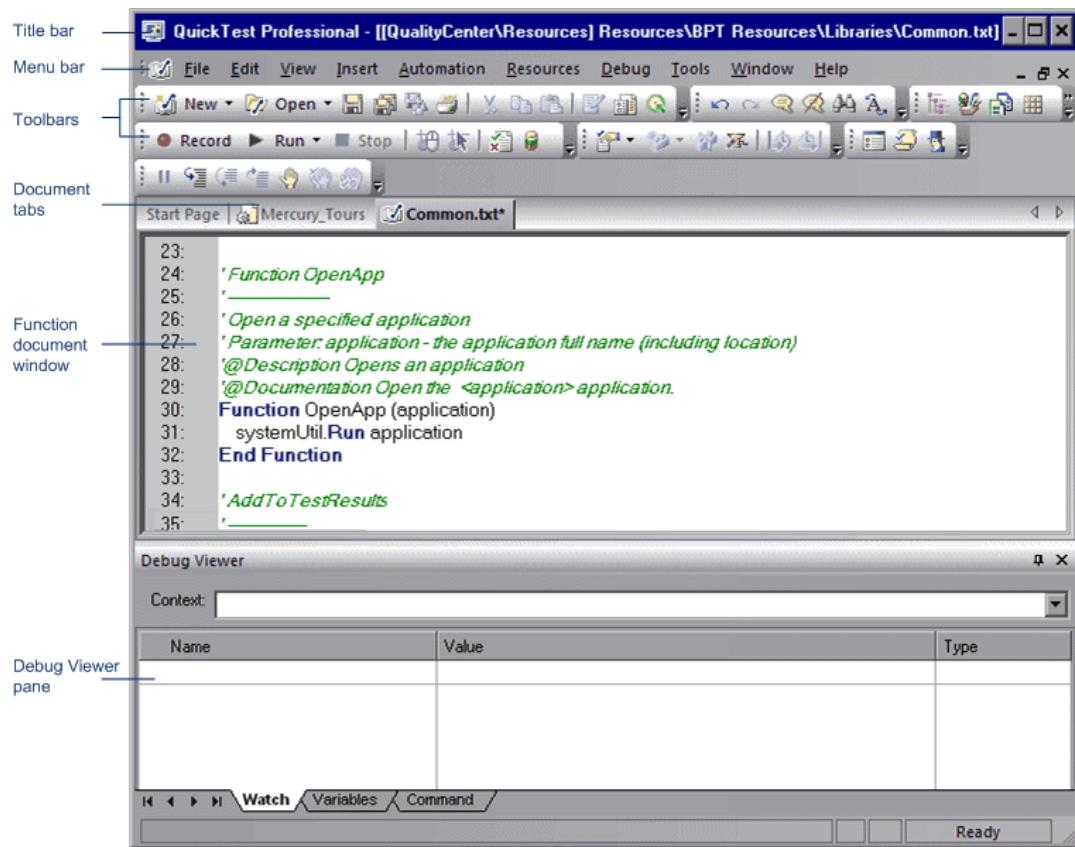
In the Expert View, QuickTest displays each operation performed on your application in the form of a script, comprised of VBScript statements. The Expert View is a script editor with many script editing capabilities. For each object and method in an Expert View statement, a corresponding row exists in the Keyword View. The **Action** list above the Expert View window lists the actions that are called from the test.



For details on using the Expert View, see Chapter 27, "Working in the Expert View and Function Library Windows."

Function Libraries at a Glance

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 details, see Chapter 29, "User-Defined Functions and Function Libraries."

 **QuickTest Panes**

The following sections describe the available QuickTest panes:

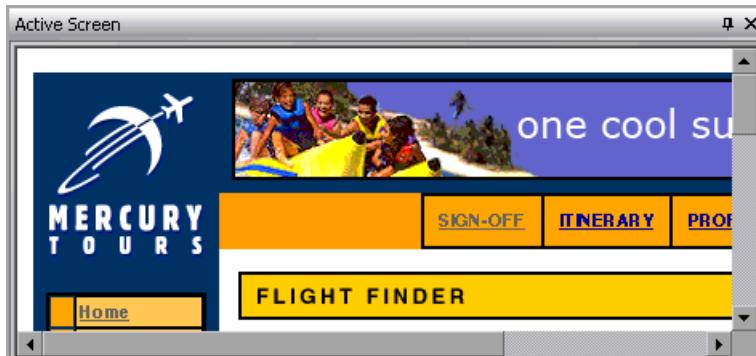
- ▶ "Active Screen Pane at a Glance" on page 70
- ▶ "Available Keywords Pane at a Glance" on page 71
- ▶ "Data Table Pane at a Glance" on page 72
- ▶ "Debug Viewer Pane at a Glance" on page 72
- ▶ "Information Pane at a Glance" on page 73
- ▶ "Missing Resources Pane at a Glance" on page 74
- ▶ "Process Guidance Panes at a Glance" on page 75
- ▶ "Resources Pane at a Glance" on page 76
- ▶ "Test Flow Pane at a Glance" on page 76
- ▶ "To Do Pane at a Glance" on page 77

Active Screen Pane at a Glance

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**.



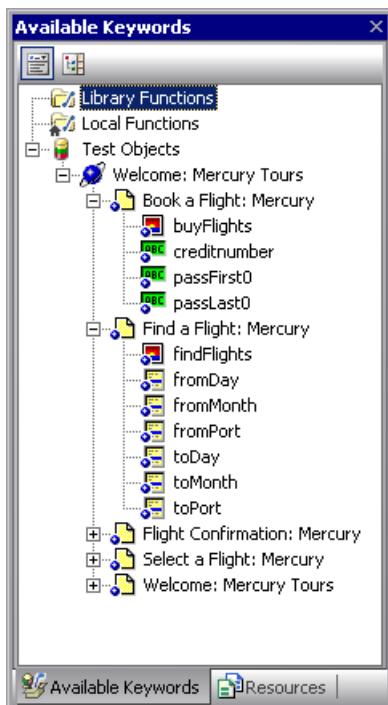
For details, see "Active Screen Pane" on page 1307.

Available Keywords Pane at a Glance

The Available Keywords pane displays all the keywords available to your test. It 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 details, see "Available Keywords Pane Overview" on page 1320.

Data Table Pane at a Glance

The data table is a spreadsheet-like table with columns and rows representing the data applicable to your test. The data table assists you in parameterizing your test, and contains one Global tab plus an additional tab for each action in your test.



To view the data table, click the **Data Table** toolbar button or select **View > Data Table**.

A1		Acapulco						
		departure	arrival	C	D	E	F	G
1		Acapulco	New York					
2		New York	Paris					
3		London	Frankfurt					
4								
5								

Global Action1

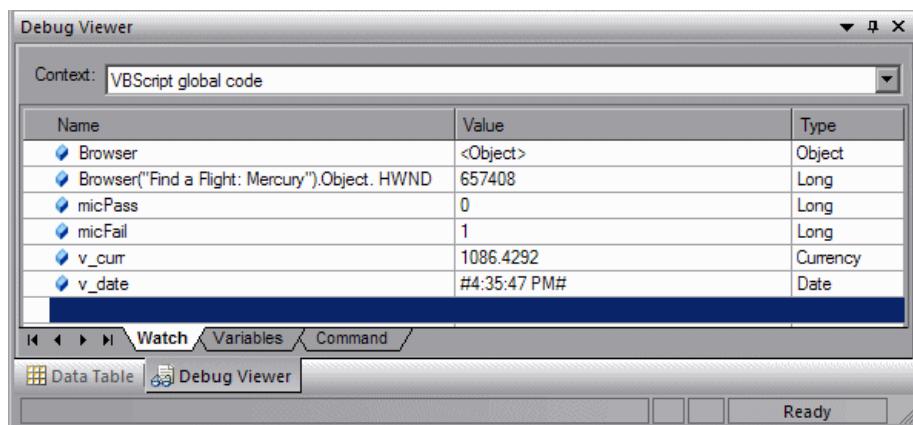
For details, see Chapter 38, "Data Table Pane."

Debug Viewer Pane at a Glance

The Debug Viewer pane assists you in debugging your tests or function libraries, and contains the following tabs:

Tab	Description
Watch	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	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	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.

To view the Debug Viewer pane, select **View > Debug Viewer**.

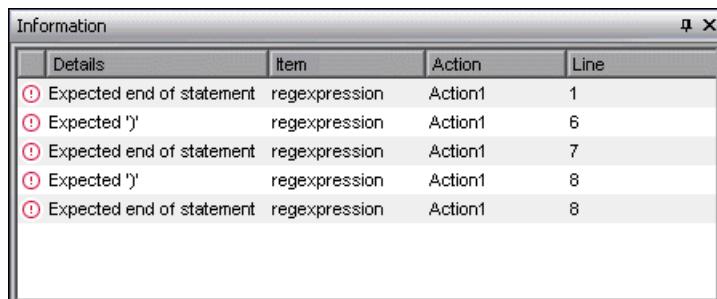


For details, see "Debug Viewer Pane" on page 1227.

Information Pane at a Glance

The Information pane provides a list of VBscript syntax errors in your test or function library scripts. 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 action or function library, and then correct it.

To view the Information pane, select **View > Information**.



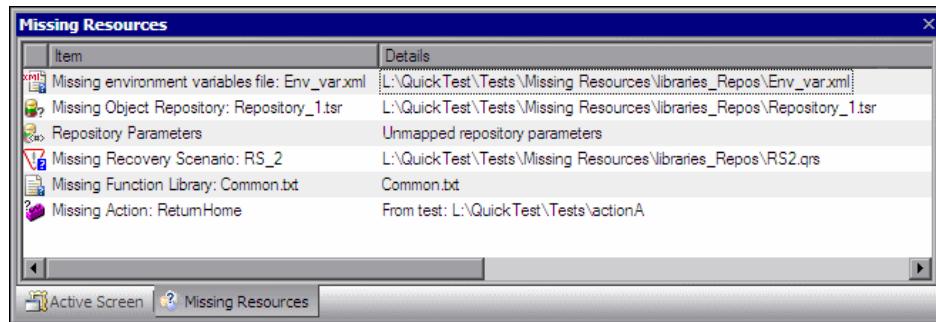
For details, see "Information Pane" on page 1357.

Missing Resources Pane at a Glance

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. The Missing Resources pane then enables you to locate or remove them from your test. If the Missing Resources pane is not currently displayed, QuickTest automatically opens it when a missing resource is detected.

To view the Missing Resources pane, select **View > Missing Resources** or click the **Missing Resource** button.



For details, see "Missing Resources Pane" on page 1363.

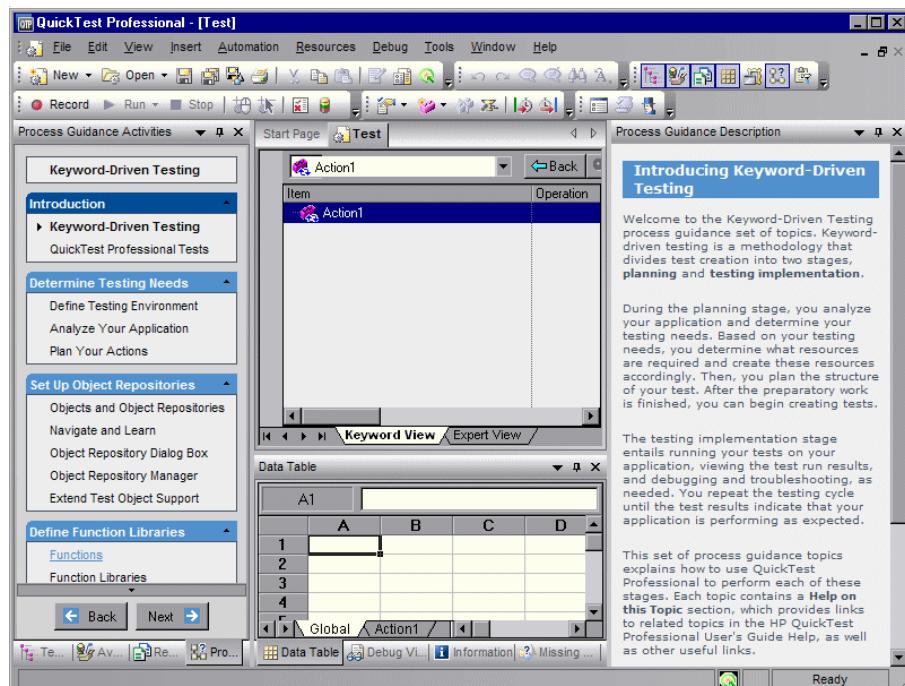
Process Guidance Panes at a Glance

Process guidance is a tool that provides procedures and descriptions on how to best perform specific processes, such as creating a test in QuickTest. You use process guidance to learn about new processes and to learn the preferred methodology for performing processes with which you are already familiar. Your organization may also provide you with process guidance that is accessible from these panes. Process guidance is displayed in two panes:

- **Process Guidance Activities pane (shown on the left).** Lists the activities that are part of the selected process, such as adding steps to a test.
- **Process Guidance Description pane (shown on the right).** Describes the tasks that you need to perform for a selected activity.



To view the Process Guidance panes, select **View > Process Guidance** or click the **Process Guidance panes** toggle button.



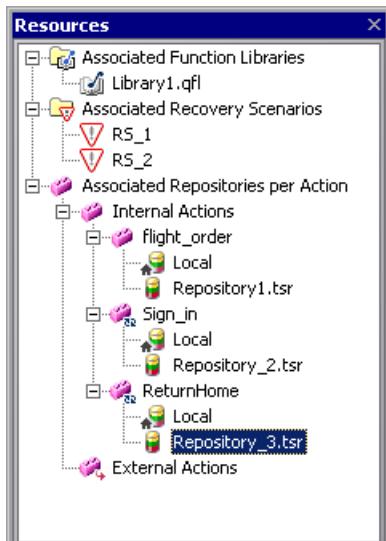
For details, see Chapter 41, "Process Guidance Panes."

Resources Pane at a Glance



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 details, see "Resources Pane" on page 1385.

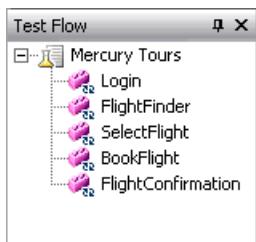
Test Flow Pane at a Glance

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 details, see "Test Flow Pane" on page 1395.



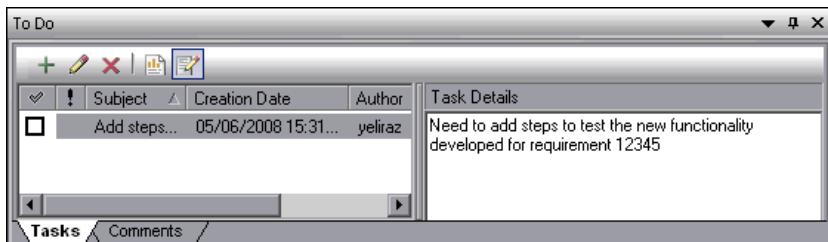
To Do Pane at a Glance

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 details, see "To Do Pane" on page 1405.

Tasks

How to Start QuickTest



- 1** From the **Start** menu, select **Programs > HP QuickTest Professional > HP QuickTest Professional**, or double-click the **QuickTest Professional** shortcut on your desktop.

Whenever you start QuickTest, the Add-in Manager dialog box opens, displaying the currently installed add-ins—unless you specify otherwise. For a user interface description, see "Add-in Manager Dialog Box" on page 113.

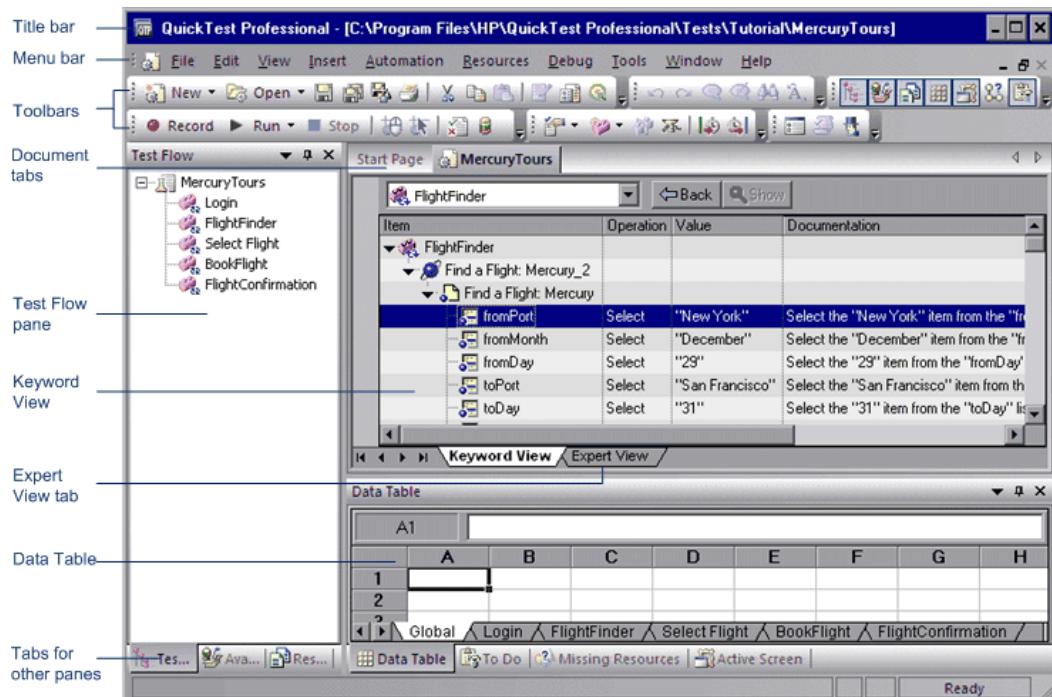
- 2** In the Add-in Manager dialog box, select the add-ins you want to load and click **OK**. The QuickTest Professional window opens, displaying the Start Page and a blank test. For details on the Start Page, see "Start Page" on page 111.

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

Reference

QuickTest Main Window - User Interface

The QuickTest window enables you to create, view and modify tests and function libraries. It also provides access to different resource management tools.



To access	Start QuickTest, as described in "How to Start QuickTest" on page 78.
Important information	You can work on one test and one or more function libraries simultaneously. For details, see "Windows Dialog Box" on page 1304.
See also	<ul style="list-style-type: none"> ➤ "QuickTest Main Window Overview" on page 64 ➤ "QuickTest Commands" on page 82 ➤ "QuickTest Panes" on page 69 ➤ "How to Customize the QuickTest Window" on page 1281

The QuickTest main window contains the following elements:

General User Interface Elements

UI Element	Description
Keyword View	Contains each step, and displays the object hierarchy, in a modular, icon-based table.
Expert View	Contains each step as a VBScript line. In object-based steps, the VBScript line defines the object hierarchy.
Start Page	Welcomes you to QuickTest and provides links to Process Guidance. You can use the shortcut buttons to open new and existing documents.
QuickTest Panes	Provide information and functionality about the currently open test or function library. For a list of available QuickTest panes, see "QuickTest Panes" on page 69.

Document Area

For your convenience, you can display one active document in the document area, or you can cascade or tile your open documents.

This area can display the following document types:

Document	Description
Test	<p>Enables you to create, view and modify tests containing actions, steps, and resources.</p> <p>Note: QuickTest also enables you to create, view, and modify business components, scripted components, and application areas. For details, see the <i>HP QuickTest Professional for Business Process Testing User Guide</i>.</p>
Function Library	<p>Enables you to create, view, and modify functions and subroutines for use with your test.</p>

Menus and Toolbars

- "File Menu" on page 84
- "Edit Menu" on page 87
- "View Menu" on page 91
- "Insert Menu" on page 93
- "Automation Menu" on page 96
- "Resources Menu" on page 98
- "Debug Menu" on page 99
- "Tools Menu" on page 100
- "Window Menu" on page 102
- "Help Menu" on page 103
- "Action Toolbar" on page 83
- "Other QuickTest Commands" on page 105

Additional Key Elements

- **Document tabs and scroll arrows.** Enable 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.
- **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.
- **Status bar.** Displays the status of the QuickTest application and other relevant information.

QuickTest Commands

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

The following menus are available:

- "Action Toolbar" on page 83
- "File Menu" on page 84
- "Edit Menu" on page 87
- "View Menu" on page 91
- "Insert Menu" on page 93
- "Automation Menu" on page 96
- "Resources Menu" on page 98
- "Debug Menu" on page 99
- "Tools Menu" on page 100
- "Window Menu" on page 102
- "Help Menu" on page 103
- "Other QuickTest Commands" on page 105

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.



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 details on actions, see Chapter 14, "Actions".

 **File Menu**

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
	ALM/QC Connection		<p>Opens the HP ALMConnection dialog box, enabling you to connect to an HP ALM or Quality Center project.</p> <p>Tip:</p> <ul style="list-style-type: none"> ▶ Double-click the ALM/QC icon on the status bar to manage your connection. ▶ Point to the ALM/QC icon on the status bar to view connection information. 
	ALM/QC Version Control		<p>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 a version-control enabled Quality Center project.</p> <p>For details, see "Version Management Commands" on page 1706.</p>
	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.

	Command	Shortcut Key	Function
	Export Test to Zip File	CTRL+ALT+S	Creates a zip file of the active test.
	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 by default from the **Standard** toolbar, shown below:



 **Edit Menu**

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.

Command	Shortcut Key	Function
Advanced > Apply "With" to Script	CTRL+W	Generates With statements for the action displayed in the Expert View, and enables IntelliSense within With statements.
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 by default from the **Edit** toolbar, shown below:





View Menu

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 pane.
	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. Note: This command is disabled if the Keyword View is hidden. You can instruct QuickTest to show or hide the Keyword View using the Display the Keyword View for tests and scripted components command (Tools > Options > General pane).

	Command	Function
	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. 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.

Some of the **View** menu commands are also available by default from the **View** toolbar, shown below:



 **Insert Menu**

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 business component. (Relevant only for business components)
	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 Run Results.

	Command	Shortcut Key	Function
	Conditional Statement		Inserts an If...Then , ElseIf...Then , or Else statement according to your selection.
	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 Service Test Test		Inserts a call to a Service Test test. (Applicable only if Service Test 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 by default from the **Insert** toolbar, shown below:



Automation Menu

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 "Run Pane (Options Dialog Box)" on page 1447.)	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).

	Command	Shortcut Key	Function
	Analog Recording	SHIFT+ALT+F3	Starts recording in Analog Recording mode.
	Low Level Recording	CTRL+SHIFT+F3	Starts recording in Low Level Recording mode.
	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 Run Results viewer, enabling you to view results for a test run session.

Some of the **Automation** menu commands are also available by default from the **Automation** toolbar, shown below:



 **Resources Menu**

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. Also available from the Automation toolbar (described on page 96).
	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.

 **Debug Menu**

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 by default from the **Debug** toolbar, shown below:



Tools Menu

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.
	Extensibility Accelerator		<p>Opens Extensibility Accelerator for HP Functional Testing, enabling you to extend the Web Add-in and customize how QuickTest recognizes and interacts with different types of controls, using a Visual Studio-like IDE.</p> <p>Note: Extensibility Accelerator is an external tool, which you install from the QuickTest Professional Installation DVD. If Extensibility Accelerator is not installed, a message box informs you that the tool is not installed and provides links to more information.</p>

	Command	Shortcut Key	Function
	Regular Expression Evaluator		Opens the Regular Expression Evaluator dialog box, enabling you to create and test a regular expression to determine if it suits your needs.
	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. You can also add an object to the local object repository and highlight the object in the application.
	Web Event Recording Configuration		Opens the Web Event Recording Configuration dialog box, enabling you to specify a recording configuration level.
	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.

	Command	Shortcut Key	Function
	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 by default from the **Tools** toolbar, shown below:



Window Menu

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**

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

Command	Shortcut Key	Function
QuickTest Professional Help	F1	Opens the QuickTest Professional Help.
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.
Product Feature Movies		Enables you to view movies illustrating various QuickTest features.
Troubleshooting & Knowledge Base		Opens the Troubleshooting area of the HP Software Support Web site, enabling you to select from several self-help troubleshooting options, including a product-specific knowledge base articles. (Requires login.) The URL is: http://h20230.www2.hp.com/troubleshooting.jsp
HP Software Support		Opens the HP Software Support Web site. This site enables you to browse the HP Support Knowledge Base and add your own articles. You can also post to and search user discussion forums, submit support requests, download patches and updated documentation, and more. The URL is: www.hp.com/go/hpsoftwaresupport

Command	Shortcut Key	Function
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>
UP Update		<p>Checks online for any available updates to all HP products installed on your computer. You can choose which updates you want to download and (optionally) install.</p>
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: http://www.hp.com/go/qtp</p>
About QuickTest Professional		<p>Displays information about the installed version of QuickTest Professional.</p>

 **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 () <i>[Microsoft Natural Keyboard only]</i>	Opens the context menu for the selected step data cell in the data table.
Expand all branches	*	Expands all branches in the Keyword View. <i>[on the numeric keypad]</i>
Expand branch	+	Expands the selected item branch and all branches below it in the Keyword View. <i>[on the numeric keypad]</i>
Collapse branch	-	Collapses the selected item branch and all branches below it in the Keyword View. <i>[on the numeric keypad]</i>
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.

QuickTest Professional Program Folder Structure

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

- **Documentation.** Provides the following links to commonly used documentation files:

Option	Description
HP 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.
HP QuickTest Professional Help	Opens a comprehensive help file containing the following: <ul style="list-style-type: none">➤ <i>HP QuickTest Professional User Guide</i>➤ <i>HP QuickTest Professional for Business Process Testing User Guide</i>➤ <i>HP QuickTest Professional Add-ins Guide</i>➤ <i>HP QuickTest Professional Object Model Reference</i> (including the relevant sections for any installed add-ins)➤ <i>QuickTest Advanced References</i> (Automation API and XML Schema references)➤ <i>Microsoft VBScript Reference.</i>

Option	Description
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.
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. 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:

Option	Description
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 details, see the <i>HP QuickTest Professional Tutorial</i> .

- **Tools.** Contains the following utilities and tools that assist you with the testing process:

Option	Description
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 details, click the Help button in the HP Micro Player window.
Java Add-in JRE Support Tool	<p>Opens the Java Add-in JRE Support Tool, which adjusts the JVM Runtime Parameters of your JRE to enable the Java Add-in to recognize Java applets and the Java objects within them.</p> <p>This tool is necessary only for certain operating systems, browsers, and JRE versions. For details, see the Java section of the <i>HP QuickTest Professional Add-ins Guide</i>.</p>
License Validation Utility	Opens the License Validation utility, which enables you to retrieve and validate license information. For details, click the Help button in the License Validation Utility window.
Password Encoder	Opens the Password Encoder tool, which enables you to encode passwords. You can use the resulting strings as method arguments or Data Table parameter values (tests only). For details, see "Password Encoder Tool" on page 519.
QuickTest Script Editor	(Relevant only for tests and function libraries) Opens the QuickTest Script Editor, which enables you to open and modify the scripts of multiple tests and function libraries, simultaneously. For details, see "QuickTest Script Editor" on page 1571.

Option	Description
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 objects when recording or running tests. For details, see the section on registering browser controls in the <i>HP QuickTest Professional Add-ins Guide</i> .
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 details, see "Remote Agent Settings Dialog Box" on page 1642.
Run Results Deletion Tool	(Relevant only for tests) Opens the Run 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 details, see "Run Results Deletion Tool" on page 1160.
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 details, see "Save and Restore Settings" on page 1811.
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 details, see "Silent Test Runner" on page 1768.
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 details, see "How to Run Your Test" on page 1067.

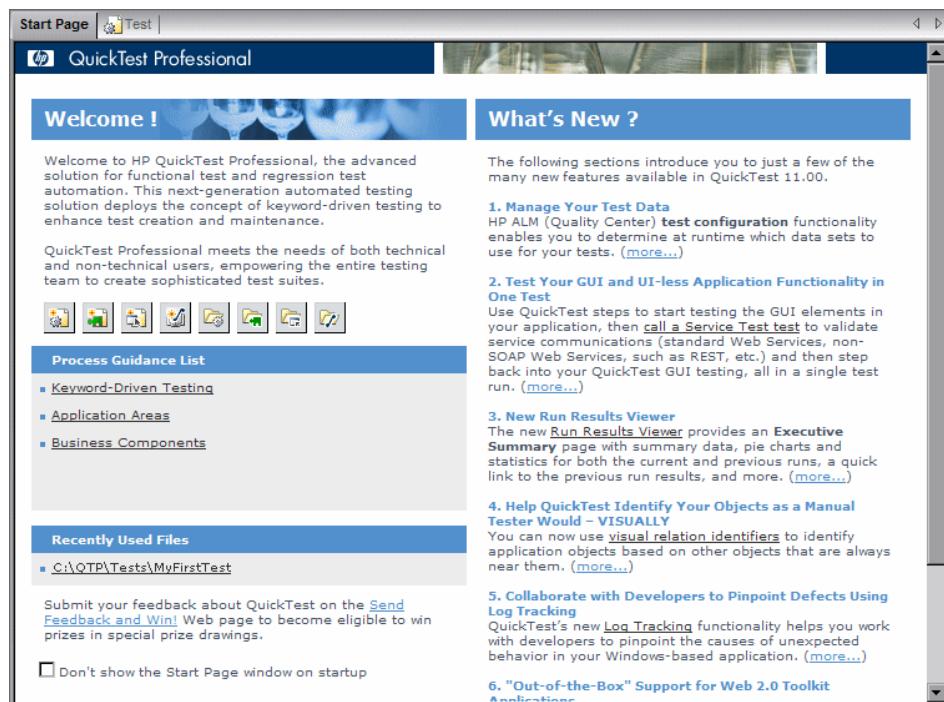
Note: There may be additional tools depending on the installed QuickTest add-ins.

- **HP QuickTest Professional.** Opens the QuickTest Professional application.
 - **HP QuickTest Professional Product Availability Matrix.** Provides a complete list of all environments, programs, and versions that are supported for QuickTest Professional and its add-ins.
 - **HP Run Results Viewer.** Opens the Run Results Viewer, which enables you to select a test and view information about the steps performed during the run session. For details, see "Run Results Viewer User Interface" on page 1107.
 - **HP Update.** Checks online for any available updates to all HP products installed on your computer. You can choose which updates you want to download and (optionally) install.HP
 - **Readme.** Opens the *HP QuickTest Professional Readme*, which provides the latest news and information on QuickTest Professional and the QuickTest Professional add-ins.
-

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.

Start Page

This 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.



To access	<p>1 Start QuickTest, as described in "How to Start QuickTest" on page 78.</p> <p>2 If the Start Page window is not displayed, select View > Start Page.</p>
------------------	--

General User Interface Elements

UI Elements	Description
Welcome!	Enables you to open new or existing documents by using the quick access buttons described below.
Process Guidance List	The list of available QuickTest processes. If your organization has descriptions for its own custom processes, these processes may also be available from the Process Guidance List . For details, see "Process Guidance Panes" on page 1375.
Recently Used Files	The list of recently opened documents.
Don't show the Start Page window on startup	When selected, instructs QuickTest to open a blank document directly after starting.
What's New area	The list of the newest features, enhancements, and supported environments in the latest version of QuickTest.

Quick Access Buttons

Button	Description	Button	Description
	Opens a new test.		Opens an existing test.
	Opens a new business component.		Opens an existing business component.
	Opens a new application area.		Opens an existing application area.
	Opens a new function library.		Opens an existing function library.

Add-in Manager Dialog Box

This dialog box enables you to select the add-ins that you want QuickTest to load by selecting the check boxes adjacent to required add-ins.



To access	By default, this dialog box opens when you start QuickTest. To display the Add-in Manager if it does not open when you start QuickTest, select Tools > Options > General node and then select Display Add-in Manager on startup .
Important information	<ul style="list-style-type: none"> ► If you select the check box of an add-in that contains a child add-in, the parent add-in is selected automatically. ► If you clear the check box for a parent add-in, the check boxes for its children are also cleared. ► QuickTest remembers which add-ins you selected so that the next time you open QuickTest, the same add-ins are selected in the Add-in Manager dialog box.
Relevant tasks	"How to Start QuickTest" on page 78
See also	"Product Information Window" on page 119

User interface elements are described below:

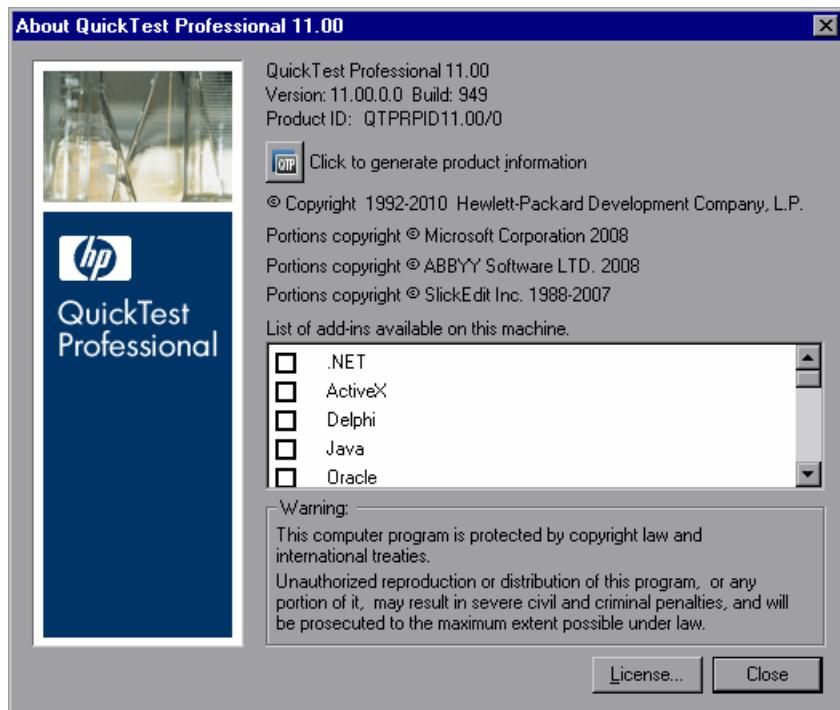
UI Element	Description
Add-in	<p>The names of the installed add-ins.</p> <p>The list of add-ins might also include child nodes representing add-ins that you or a third party developed to support additional environments or controls using add-in extensibility. For more information, see the relevant Add-in Extensibility Developer Guide, available from the QuickTest Professional Extensibility Documentation program group (Start > Programs > HP QuickTest Professional > Extensibility > Documentation).</p>
License	<p>The license used by the add-in, if any, and the time remaining until a time-limited license expires:</p> <ul style="list-style-type: none"> ▶ Licensed. Applies to the add-ins that are provided with QuickTest Professional. Add-ins use the same license as QuickTest Professional. Therefore, if QuickTest uses a Permanent license, the add-ins use the same Permanent license; if QuickTest uses a Time-Limited license, the add-ins use the same Time-Limited license. ▶ Not Licensed. Applies to an add-in that does not have an installed seat license or access to a concurrent license (for example, if all concurrent licenses are currently in use, or if the required add-in license is not installed on the concurrent license server on your subnet). To load the add-in, you first need to install or access a license. ▶ Time Remaining. Specifies the number of days and hours remaining until a time-limited add-in license expires. (Displayed only when using a QuickTest seat license—not a concurrent license.) <p>For more details, see the <i>HP QuickTest Professional Installation Guide</i>.</p>
Add-in Description	The description of the environment that the selected add-in supports.

UI Element	Description
Load Unified Functional Testing license	<p>Instructs QuickTest to use a UnifiedFunctionalTesting license from the concurrent license server.</p> <p>This check box must be selected if you want to work with tests that contain calls to Service Test tests. (Not relevant for components)</p> <p>This option may be enabled, disabled, or hidden, as follows:</p> <ul style="list-style-type: none"> ➤ Enabled. A UnifiedFunctionalTesting license is one of the licenses currently available on the concurrent license server. ➤ Enabled and selected. You selected this check box the last time you opened QuickTest, and a Unified Functional Testing license is one of the licenses currently available on the concurrent license server. ➤ Disabled and selected. This occurs in any of the following cases: <ul style="list-style-type: none"> ➤ The only available concurrent license is a UnifiedFunctionalTesting license. ➤ Service Test is currently open on your computer, and Service Test is using a UnifiedFunctionalTesting license. ➤ A UnifiedFunctionalTesting license is the minimum required license for one or more of the installed add-ins. ➤ Disabled and cleared. No UnifiedFunctionalTesting license is currently available. ➤ Hidden. QuickTest is currently using a seat license. <p>Note: QuickTest remembers your last selection, so if you cleared the Show on startup check box in a previous session, QuickTest tries to load the same license type that it used in that session, if available. To change the license type, display this dialog box (as described below).</p>

UI Element	Description
Show on startup	<p>Instructs QuickTest to display the Add-in Manager dialog box each time you open QuickTest.</p> <p>When this check box is cleared, QuickTest opens and loads the same add-ins it loaded in the previous session, without displaying the Add-in Manager.</p> <p>Note for concurrent license users: If this check box was cleared in the previous session, and the license type selected from the concurrent license server in that session is not currently available, QuickTest tries to load an available license that matches the selected add-ins.</p> <p>To display the Add-in Manager again: Select Tools > Options > General node and select Display Add-in Manager on startup.</p>

About QuickTest Professional Dialog Box

This dialog box enables you to 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 access	Select Help > About QuickTest Professional .
-----------	--

User interface elements are described below (unlabeled elements are shown in angle brackets):

UI Element	Description
<version information area>	The version of QuickTest that is installed on your computer, its build number, and Product ID number.
	Opens the "Product Information Window" on page 119, enabling you to view more detailed information on the QuickTest Professional products installed on your computer.
List of add-ins available on this machine	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 details on QuickTest add-ins, see the <i>HP QuickTest Professional Add-ins Guide</i> .
License	Enables you to view details for, or modify, the QuickTest Professional licenses installed on your computer. For details, see the <i>HP QuickTest Professional Installation Guide</i> .

Product Information Window

This window displays detailed information on the QuickTest Professional products installed on your computer.

Product Information

Product name:	QuickTest Professional
Product version:	11.00
Product ID:	QTPRPID11.00/01
Product build:	882
Operating system:	Microsoft Windows XP Service Pack 2 (Build 2600)
Internet Explorer version:	7.0.5730.13
ALM/QC connectivity:	11.0.0.4146

Add-in Information:

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

Hotfix and Patch Information:

Name
QTP_00557 for HP QuickTest Professional 11.00 QFE
QTP_00558 for HP QuickTest Professional 11.00 QFE
QTP_00559 for HP QuickTest Professional 11.00 QFE

© Copyright 1992–2010 Hewlett-Packard Development Company, L.P.



To access	In the About QuickTest Professional dialog box, click the Product Information button  .
See also	"About QuickTest Professional Dialog Box" on page 117

User interface elements are described below (unlabeled elements are shown in angle brackets):

UI Elements	Description
<general product 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.
ALM/QC connectivity	The version of the Quality Center or HP ALM 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.

Troubleshooting and Limitations - QuickTest Program Management

This section describes troubleshooting and limitations for QuickTest tools and program management.

Tools in the HP QuickTest Professional program group

You cannot use some of the tools from the **HP QuickTest Professional > Tools** program group when the UAC (User Account Control) option in is set to ON. (Relevant for Microsoft Windows Vista, Windows 7, Windows Server 2008, and Windows Server 2008 R2.)

Workaround: Temporarily turn off the UAC option while using these tools, by doing the following:

For Microsoft Windows Vista and Windows Server 2008:

- 1 Log in as an administrator.
- 2 From the Control Panel, select **User Accounts > Change Security Settings**, and clear the **Use User Account Control (UAC) to help protect your computer** check box.
- 3 After working with the desired tool, return to the **Change Security Settings** screen, and select the **Use User Account Control (UAC) to help protect your computer** check box to turn on UAC again.

For Microsoft Windows 7 and Windows Server 2008 R2:

- 1 Log in as an administrator.
- 2 From the Control Panel, select **User Accounts > User Accounts > Change User Account Settings**.
- 3 In the User Account Control Settings window, move the slider to **Never notify**.
- 4 After working with the desired tool, return to the User Account Control Settings window, and restore the slider to its previous position to turn the UAC option on again.

Characters do not display correctly

QuickTest does not fully support UTF-16 surrogate pairs and combining characters. The Run Results Viewer and some user interface elements in QuickTest do not display these characters correctly.

QuickTest and DEP (Data Execution Prevention)

On Windows 7 64-bit and Windows Server 2008 R2 operating systems, QuickTest crashes when the DEP (Data Execution Prevention) flag is set to **Always On**.

Workaround: Switch off the DEP, or modify its settings to be ON only for important operating system processes.

Part II

Working with Test Objects

3

The Test Object Model

This chapter includes:

Concepts

- ▶ [Test Object Model - Overview on page 126](#)
- ▶ [How QuickTest Applies the Test Object Model Concept on page 130](#)
- ▶ [Object Repository Types - Overview on page 137](#)
- ▶ [Deciding Whether to Use Local or Shared Object Repositories on page 140](#)

Tasks

- ▶ [How to Use the Object Spy to View Object Properties and Operations or Add an Object to a Repository on page 144](#)

Reference

- ▶ [Comparison of Object Repository Window and Object Repository Manager on page 147](#)
- ▶ [Object Identification Process Workflow on page 149](#)
- ▶ [Object Spy Dialog Box on page 151](#)
- ▶ [Object Selection Dialog Box on page 155](#)
- ▶ [Tips for Using the Pointing Hand on page 157](#)

[Troubleshooting and Limitations - Object Spy on page 158](#)

Concepts

Test Object Model - Overview

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.

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.



How QuickTest Identifies Objects During a Run Session

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

Suppose as a continuation to the experiment (described in "How QuickTest Applies the Test Object Model Concept" on page 130), 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.

For example, Alex is told he will have to recognize a particular tree out of multiple trees in the photo, and he knows he is going to have to be able to identify it again in a photo taken from a different angle. If there isn't enough clearly identifying information about the tree itself, then he might take note of where the tree is located relative to some other permanent item, such as a nearby lamp post or picnic table. Then he will be able to identify the tree again, even if the next picture he sees is from a different angle (as long as all the required items are still visible in the picture).

This is similar to the **visual relation identifier** property, which enables QuickTest to identify test objects according to their neighboring objects in the application. You use this property to link less stable test objects to more unique test objects, and as long as those objects in the application maintain their relative location to your object, QuickTest should still be able to identify the test object even after predictable user interface changes in the application.

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 7, "Configuring Object Identification."

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

For a flowchart illustrating the general object identification process, see "Object Identification Process Workflow" on page 149.

How QuickTest Applies 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 4, "Managing Test Objects in Object Repositories", Chapter 6, "Shared Object Repositories" and Chapter 11, "Test Creation - 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.

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.

Test Object Descriptions

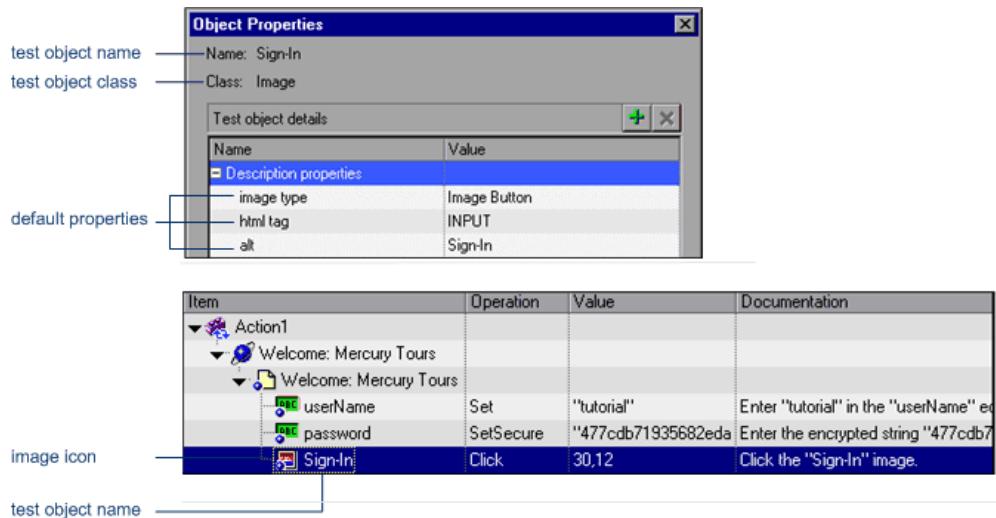
For each test object class, QuickTest learns a set of identification properties when it learns an object, and selects a sub-set of these properties to serve as a unique object description. QuickTest then uses this description to identify the object when it runs the test.

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 7, "Configuring Object Identification."

Example

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



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

QuickTest Test 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*.



Properties and Operations for Test Objects and Run-Time Objects

QuickTest uses unique terms to differentiate between the properties and operations for test objects and run-time objects. The table below introduces some of these terms.

QuickTest Test Objects	Run-time Objects From Your Application
<p>Identification properties are QuickTest-specific properties that QuickTest uses to identify objects in applications, to retrieve and store information about those objects, or to compare stored values with the current values of an object in an application.</p> <p>The identification properties available for a test object are determined by its test object class (and not by the actual properties available for the object in the application).</p>	<p>Native properties are the properties created by the object creator for each run-time object. (Examples of object creators include Microsoft for Microsoft Internet Explorer objects, Netscape for Netscape Browser objects, and the product developer for ActiveX objects.)</p>
<p>A test object operation is a method or property that QuickTest can perform on an object from a particular test object class.</p> <p>Example: QuickTest can perform the Click method on a WebButton test object.</p>	<p>Native operations are the methods of the object in your application as defined by the object creator.</p>

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.

The sections below describe some of the ways in which you can view and modify properties and operations for test objects and run-time objects.

For test objects

- You can retrieve or modify identification property values manually while designing your test, or you can use **SetTOProperty** statements during a run session.
For more information, see "Managing Test Objects in Object Repositories" on page 159, "Retrieving and Setting Identification Property Values" on page 961, and the *HP QuickTest Professional Object Model Reference*.
- You can use regular expressions to identify property values based on conditions or patterns you define.
For more information, see "Regular Expressions Overview" on page 863.
- You can parameterize identification property values with data table parameters so that a different value is used during each iteration of the test.
For more information, see "Parameterizing Values" on page 723.
- 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 168.
- You can view the current identification property values of any visible object using the Properties tab of the Object Spy.
For more information, see "How to Use the Object Spy to View Object Properties and Operations or Add an Object to a Repository" on page 144.
- You can view the syntax of the test object operations of any visible object using the Operations tab of the Object Spy. For more information, see "How to Use the Object Spy to View Object Properties and Operations or Add an Object to a Repository" on page 144.

For run-time objects

- You can view the syntax of the native operations of any visible object using the Operations tab of the Object Spy. For more information, see "How to Use the Object Spy to View Object Properties and Operations or Add an Object to a Repository" on page 144.

- You can retrieve native 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 961.
- If the available test object operations and identification properties do not provide the functionality you need, you can access the internal operations and native 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 "Native Properties and Operations" on page 962 and the *HP QuickTest Professional Object Model Reference*.

Object Repository Types - Overview

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). You can use the same shared object repository for multiple actions. You can also use multiple object repositories for each action.

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.

Tests always use 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.

You perform many object repository-related tasks either in the Object Repository window or in the Object Repository Manager. To view a list of the tasks that you can perform in each, see "Comparison of Object Repository Window and Object Repository Manager" on page 147.

Planning where to store 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.
- 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. Use a combination of objects from your local and shared object repositories, according to your needs.
- 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 140.

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 (as long as the Object Repository Manager is connected to your Quality Center project).

How QuickTest determines the object to use when an object with the same name exists in multiple object repositories

- 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 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 "Associate Repositories Dialog Box" on page 229.

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.

The following table describes when it is preferable to use each type of object repository:

Use this object repository type...	In these situations...
local object repository	<ul style="list-style-type: none">➤ You are creating single-action tests.➤ You are creating simple tests, especially under the following conditions:<ul style="list-style-type: none">➤ 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 are new to using QuickTest. 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. <p>See also: "Local Object Repository - Overview" on page 141</p>

Use this object repository type...	In these situations...
shared object repository	<ul style="list-style-type: none"> ➤ 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 often work with multi-action tests and regularly use the Insert Copy of Action and Insert Call to Action options. ➤ 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. ➤ 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. <p>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.</p> <p>See also: "Shared Object Repository - Overview" on page 143</p>



Local Object Repository - Overview

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 "Shared Object Repositories Overview" on page 250.) 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).

 **Shared Object Repository - Overview**

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 "Local Object Repository - Overview" on page 141.)

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 and Deleting Test Objects in a Local or Shared Object Repository" on page 160.

You modify a shared object repository using the Object Repository Manager. For more information, see Chapter 6, "Shared 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 221.

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 "When to Update a Shared Object Repository from Local Object Repositories" on page 339.

Tasks

How to Use the Object Spy to View Object Properties and Operations or Add an Object to a Repository

This task describes how to use the Object Spy to view identification properties, native properties, test object operations, or native operations, as well as add an object to an object repository.

This task includes the following steps:

- "Prerequisites" on page 145
- "Open the Object Spy dialog box" on page 145
- "Select the details you want to view for the object" on page 145
- "Resize the Object Spy dialog box - Optional" on page 145
- "Use the pointing hand to select the application object on which you want to Spy" on page 145
- "Add selected objects to the object repository - Optional" on page 146

1 Prerequisites

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

2 Open the Object Spy dialog box

For more information, see the **To Access** section of the "Object Spy Dialog Box" on page 151.

3 Select the details you want to view for the object

In the Object Spy dialog box, select the **Properties** or **Operations** tab, and then select the radio button to view the **Native** or **Identification** properties.

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 961 and "Native Properties and Operations" on page 962.

4 Resize the Object Spy dialog box - Optional

This is useful if the objects on which you want to spy have a deep hierarchy or long property names and values, as it enables you to view all of the information without scrolling. For more information, see "Object Spy Dialog Box" on page 151.

5 Use the pointing hand to select the application object on which you want to Spy

In the Object Spy, click the pointing hand button and then mouse over or click an object in your application. In most environments, as you mouse over objects in your application, the Object Spy highlights the object it identifies and displays the relevant information in the Object Spy, according to the options you selected in step 3.

When you click an object in your application, the Object Spy captures its information, and you can:

- ▶ Change the selected radio button or tab in the Object Spy to view additional details.
- ▶ View properties, values, or operations of other test objects currently displayed in the **Object hierarchy** tree, by selecting that test object in the tree.
- ▶  Highlight the object in the application, by clicking the **Highlight in Application** button.

For details on using the pointing hand, see "Tips for Using the Pointing Hand" on page 157.

6 Add selected objects to the object repository - Optional



After clicking an object you can add it (or any other object in the **Object hierarchy** tree) to the object repository using the **Add to Repository button**. For details, see "Object Spy Dialog Box" on page 151.

If an object in the **Object hierarchy** tree already exists in a repository associated with the active action, a repository icon  is displayed in the lower-right corner of the object's icon.

Reference



Comparison of Object Repository Window and Object Repository Manager

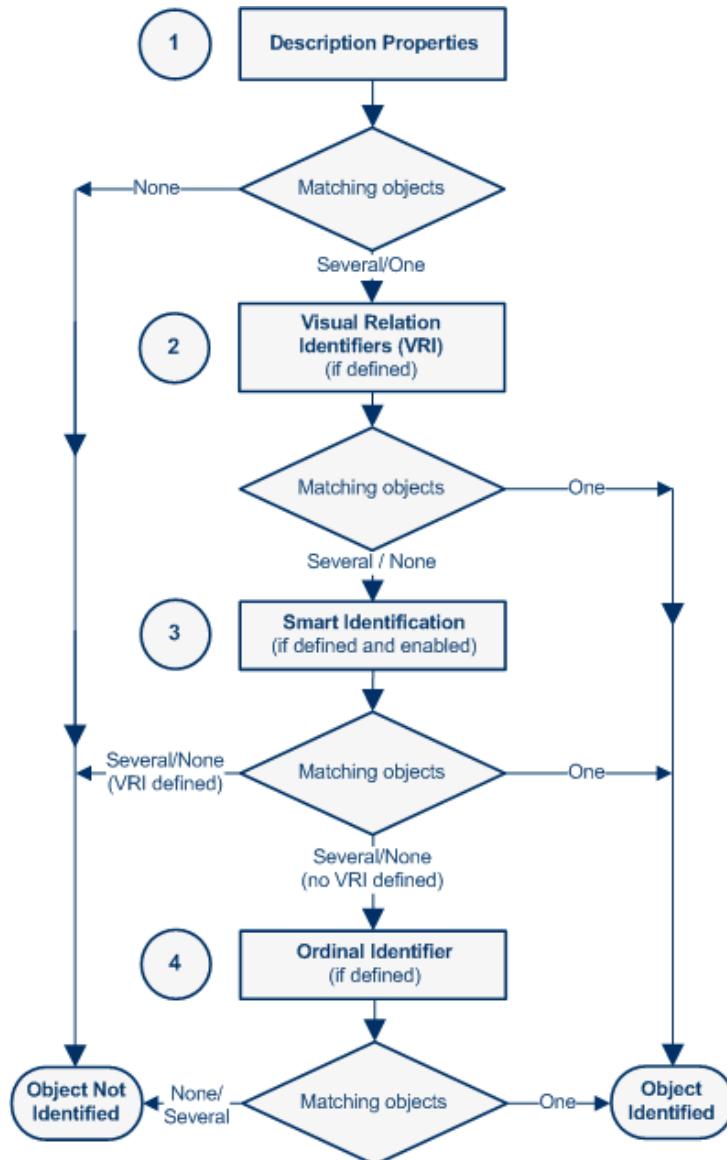
You perform many object repository-related tasks either in the Object Repository window or in the Object Repository Manager. Some object repository-related tasks can also be performed in both. The following table lists features and functionality, indicating if they are available in the Object Repository window or the Object Repository Manager:

Functionality	Object Repository window	Object Repository Manager
"Adding and Deleting Test Objects in a Local or Shared Object Repository" on page 160	✓	✓
"Copying, Pasting, Moving, or Deleting Objects in the Object Repository" on page 179	✓	✓
"Highlighting an Object in Your Application" on page 166	✓	✓
"Locating a Test Object in the Object Repository" on page 166	✓	✓
"Specifying or Modifying Property Values" on page 168	✓	✓
"Updating Identification Properties from an Object in Your Application" on page 168	✓	✓
"Restoring Default Mandatory Properties for a Test Object" on page 169	✓	✓
"Renaming Test Objects" on page 170	✓	✓
"Adding Properties to a Test Object Description" on page 171	✓	✓

Functionality	Object Repository window	Object Repository Manager
"Defining New Identification Properties" on page 173	✓	✓
"Removing Properties from a Test Object Description" on page 188	✓	✓
"Exporting Local Objects to a Shared Object Repository" on page 221	✓	✗
"Copying Objects to the Local Object Repository" on page 222	✓	✗
"Managing Shared Object Repositories" on page 259 (includes creating, opening, saving, closing, and enabling editing functionality)	✗	✓
"Adding a Test Object to an Object Repository" on page 176	✓	✓
"Adding Test Objects Using the Navigate and Learn Option" on page 264	✗	✓
"Working with Repository Parameters" on page 252	✗	✓
"Merging Two Shared Object Repositories" on page 344	✗	✓
"Importing and Exporting Shared Object Repositories Using XML" on page 251	✗	✓

Object Identification Process Workflow

The following flowchart provides a general overview of the main process of QuickTest object identification.



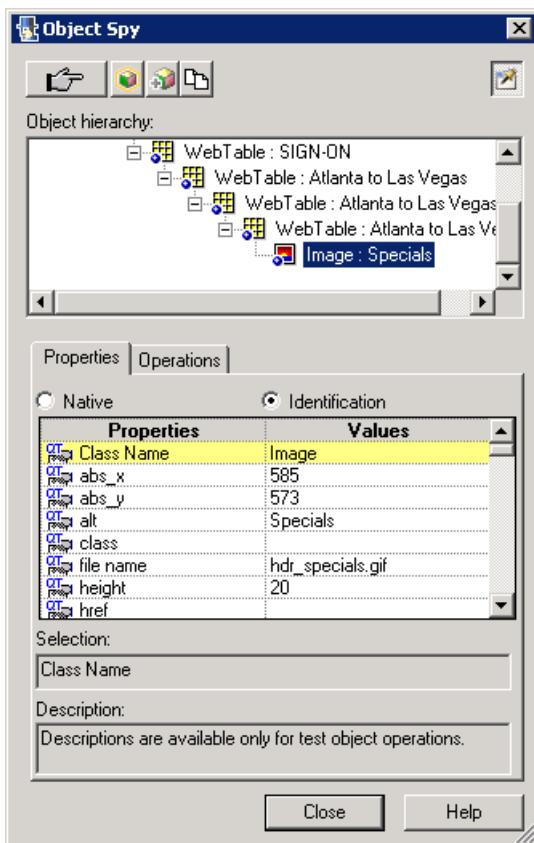
Note for Web-based objects:

- ▶ If you defined Web object identifiers (such as XPath/CSS properties) for these test objects, they are used before the description properties. If one or more objects are found, QuickTest continues to identify the object using the description properties. For details, see the section on Web Object Identifiers (described in the *HP QuickTest Professional Add-ins Guide*).
 - ▶ Additional QuickTest-generated properties, such as source index or automatic XPath, may also affect the object identification process. You enable these properties in the Advanced Web Options tab of the Options dialog box (described in the *HP QuickTest Professional Add-ins Guide*).
-

Object Spy Dialog Box

This dialog box 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.

You can also check if an object is in a repository associated with your action, add the selected object to the local object repository (or a shared object repository if you are using the Object Spy from the Object Repository Manager), and highlight an object in the application.



To access	<p>Use one of the following:</p> <ul style="list-style-type: none"> ➤ Select Tools > Object Spy. ➤ Click the Object Spy toolbar button . <p>Available from:</p> <ul style="list-style-type: none"> ➤ QuickTest Main Window - User Interface (described on page 79) ➤ Object Repository Window (described on page 237) ➤ Object Repository Manager Main Window (described on page 266)
Relevant tasks	<p>"How to Use the Object Spy to View Object Properties and Operations or Add an Object to a Repository" on page 144</p>
See also	<p>"Properties and Operations for Test Objects and Run-Time Objects" on page 135</p>

User interface elements are described below:

Option	Description
	<p>Pointing Hand. Turns the mouse pointer into a pointing hand. Use the pointing hand to highlight or click the object whose properties and/or operations you want to view.</p> <ul style="list-style-type: none"> ➤ As you move the pointing hand over the objects in the application, the objects are highlighted in the application (in some environments), and their details are displayed in the Object Spy dialog box. ➤ To capture information about a particular object and its parent objects in the Object Spy, click on the object in the application. <p>See also: "Tips for Using the Pointing Hand" on page 157.</p>
	<p>Keep on Top. Keeps the Object Spy dialog box in view while spying on an object in your application.</p> <p>Note: When the Keep on Top button is not pressed, the Object Spy dialog box may 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>

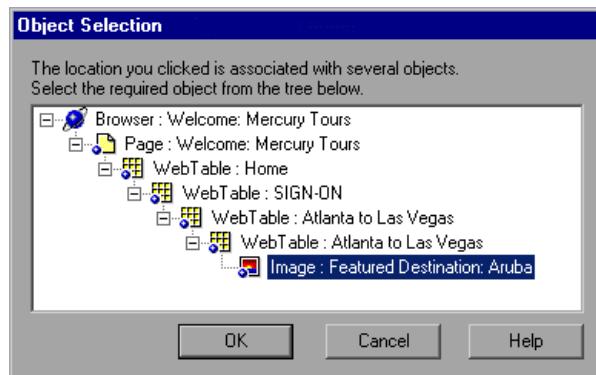
Option	Description
	<p>Highlight in Application. Highlights the object in the application that corresponds to the test object currently selected in the Object Spy.</p>
	<p>Add Object to Repository. Adds the test Object selected in the Spy hierarchy tree to the object repository.</p> <ul style="list-style-type: none"> ➤ If the object is successfully added to the object repository, the repository icon  is added to the bottom of the test object icon. ➤ When the Object Spy dialog box is opened from QuickTest or the Object Repository Window, the object is added to the local object repository of the active action . ➤ When the Object Spy dialog box is opened from the Object Repository Manager, the object is added to the active object repository. This option is disabled when there is no open repository.
	<p>Copy Identification Properties to Clipboard. Copies all of the properties and values for the object currently selected in the Object hierarchy tree. You can paste the copied data from the Clipboard into any document.</p> <p>Enabled only when the Identification radio button is selected in the Properties tab.</p> <p>The copied properties and values are formatted in standard programmatic description syntax with line breaks between each property-value pair. For example:</p> <pre>"Class Name:=Image", "abs_x:=585", "abs_y:=573", "alt:=Specials",</pre> <p>This option is useful if you want to:</p> <ul style="list-style-type: none"> ➤ Compare the properties and values of two objects in your application. ➤ Copy the relevant strings when creating programmatic descriptions. <p>For more information on programmatic descriptions, see "Programmatic Descriptions" on page 946.</p>

Option	Description
Object hierarchy	<p>Displays the hierarchy of test objects that are related to the object you selected in your application.</p> <ul style="list-style-type: none"> ➤ If an object in the hierarchy already exists in a repository associated with the active action, a repository icon  is displayed in the lower-right corner of the object's icon. ➤ To view properties, values, or operations for another test object in the Object hierarchy tree, select that test object in the tree. ➤ While an object is highlighted in the application, test object classes are displayed in the Object hierarchy tree, but test object names are not. Test object names (such as Atlanta to Las Vegas and Specials in the image shown above) are displayed only after clicking the object to capture the information in the Object Spy.
Properties tab	<p>Select the Native or Identification radio button to display the native properties and their values, or the test object identification properties and values of the run-time object associated with the selected test object in the Object hierarchy tree.</p> <ul style="list-style-type: none"> ➤ Properties. Displays the identification or native property names for the test object that is currently selected in the Object hierarchy tree. ➤ Values. Displays the identification or native property values of the relevant object in the application.
Operations tab	<p>Select the Native or Test Object radio button to display 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.</p>

Option	Description
Selection	<p>The content in this box differs depending on which tab is displayed above it.</p> <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 operation, when available.

Object Selection Dialog Box

This dialog box enables you to specify which object you want QuickTest to use when you select a location in your application that is associated with more than one object.



To access	<ul style="list-style-type: none">➤ After performing an operation that changes the pointer to a pointing hand, if you point to a location that is associated with more than one object, the Object Selection dialog box opens.➤ When you right-click a location in the Active Screen to perform an operation that requires selecting an object, if the location is associated with more than one object, the Object Selection dialog box opens.
See also	"Tips for Using the Pointing Hand" on page 157

User interface elements are described below (unlabeled elements are shown in angle brackets):

UI Elements	Description
<object tree>	A tree containing the parent hierarchy of the object you selected. Select the object that you want QuickTest to use from the tree and click OK .



Tips for Using the Pointing Hand

Clicking the pointing hand button converts the mouse (device) pointer into a pointing hand. You use the pointing hand to highlight or click an object in your application.

This section provides 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.

Examples of pointing hand usage

From the Object Spy, you use the pointing hand to view the object's properties and/or operations.

From the Select Object for Step dialog box, you use the pointing hand to add an object to your test. When you click the pointing hand, QuickTest is hidden, enabling you to view the entire desktop.

Troubleshooting and Limitations - Object Spy

When working in Windows XP, using **CTRL+Left mouse button** to select Taskbar items is not supported.

Workaround: Use **CTRL+Right mouse button** to open the shortcut menu, and then select **Restore** to display the selected item.

4

Managing Test Objects in Object Repositories

This chapter includes:

Concepts

- Adding and Deleting Test Objects in a Local or Shared Object Repository on page 160
- Guidelines for Copying, Pasting, and Moving Objects on page 163
- Locating Objects on page 165
- Maintaining Identification Properties - Overview on page 167
- Visual Relation Identifiers on page 174

Tasks

- How to Add a Test Object to an Object Repository on page 176
- How to Copy, Paste, Move, or Delete Objects in the Object Repository on page 180
- How to Locate an Object in an Object Repository on page 183
- How to Maintain Test Objects in Object Repositories on page 184
- How to Define a Visual Relation Identifier for a Specific Test Object - Use-Case Scenario on page 190

Reference

- Add Properties Dialog Box on page 197
- Define New Test Object Dialog Box on page 200
- Define Object Filter Dialog Box on page 202

- ▶ Find and Replace Dialog Box on page 206
- ▶ Ordinal Identifier Dialog Box on page 208
- ▶ Visual Relation Identifier Dialog Box on page 210

[Troubleshooting and Limitations - Managing Test Objects](#) on page 217

Concepts

Adding and Deleting Test Objects in 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 an 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.

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.

If needed, you can merge test objects from the local object repository into a shared object repository. For more information, see Chapter 9, "Object Repository Merge Tool."

You can also add test objects to a shared object repository while navigating through your application. For more information, see "Navigate and Learn Toolbar" on page 274.

This section also includes:

- "Adding Test Objects to the Local Object Repository from the Active Screen" on page 161
- "Defining New Test Objects" on page 162
- "Deleting Objects in the Object Repository" on page 162

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 "Active Screen Pane (Options Dialog Box)" on page 1434.

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 be used only 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).

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 be used only 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 6, "Shared 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 168.

Deleting Objects in the Object Repository

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

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

If you are working with a shared object repository, confirm that the object does not appear in any other action using the same shared object repository before you choose to delete the object from the object repository. Also, confirm that the object you are deleting is not used in a visual relation identifier for another test object.

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.



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 a 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.
- If you delete a test object that are used in a visual relation identifier for another test object, object identification for that test object will fail.

Locating Objects

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.

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`.

Note: When locating test objects, QuickTest also takes into account the visual relation identifier property, if defined for that test object. For details, see "Visual Relation Identifiers" on page 174.

This section also includes:

- "Finding Objects in an Object Repository" on page 165
- "Highlighting an Object in Your Application" on page 166
- "Locating a Test Object in the Object Repository" on page 166

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.

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. Both the frame and the flashing behavior are temporary.

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.



Maintaining Identification Properties - Overview

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.

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 (assuming that the **text** property is included in the test object's description).

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.

This section also includes:

- "Specifying or Modifying Property Values" on page 168
- "Updating Identification Properties from an Object in Your Application" on page 168
- "Restoring Default Mandatory Properties for a Test Object" on page 169
- "Renaming Test Objects" on page 170
- "Adding Properties to a Test Object Description" on page 171
- "Defining New Identification Properties" on page 173
- "Ordinal Identifiers" on page 173

Specifying or Modifying Property Values

You can specify or modify values for properties in the test object description. You can specify a value using a constant value (either a simple value or a constant value that includes regular expressions) or you can parameterize it. You can 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 168.

You can also find and replace specific identification property values. For more information, see "Find and Replace Dialog Box" on page 206.

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 7, "Configuring Object Identification."

Updating Identification Properties from an Object in Your Application

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 "Define New Test Object Dialog Box" on page 200.

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 169.

Restoring Default Mandatory Properties for a Test Object

You can restore the default properties for a selected test object. When you restore the default properties, it restores the mandatory property set defined for the selected object class, 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, those 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.

You can use the **Restore mandatory property set** option to restore 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 are currently different for this test object class than they were 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 168.

Renaming Test Objects

When an object changes in your application, or if you are not satisfied with the current name of a test object for any reason, you can change the name that QuickTest assigns to the stored object. You can also provide 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 for you to identify in the test, for example, Color_Picker, Eraser, Airbrush, and so forth.

Renaming a test object does not affect the way QuickTest recognizes the object in your application, as the test object name is not included in the test object description.

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.

Adding Properties to a Test Object Description

You can add to the list of properties that QuickTest uses to identify an object. For each object class, QuickTest has a default property set that it uses for the object description for a particular 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 "New Property Dialog Box" on page 199.

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.

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 283. 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 "Update Options Tab (Update Run Dialog Box)" on page 1271.

Defining New Identification Properties

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.

Ordinal Identifiers

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

Note: When visual relation identifiers are used, the **Ordinal identifier** option is disabled in the Object Repository Manager or window. For details on visual relation identifiers, see "Visual Relation Identifiers" on page 174.

For more information on ordinal identifiers, see "Ordinal Identifiers" on page 287.

Visual Relation Identifiers

When testing applications with multiple identical objects, QuickTest assigns an ordinal identifier to each test object. This may lead to unreliable object identification. However, it may not (immediately) result in a failed step. For details on ordinal identifiers, see "Ordinal Identifiers" on page 287.

A **visual relation identifier** is a set of definitions that enable you to identify the object in the application according its neighboring objects in the application. You can select neighboring objects that will maintain the same relative location to your object, even if the user interface design changes. This enables you to help QuickTest identify similar objects much as a human tester would, and helps in creating more stable object repositories that can withstand predictable changes to the application's user interface.

You define visual relations in the Visual Relation Identifier dialog box, which is accessible from the local or shared object repository, and from the Object Properties dialog box. For user interface details, see "Visual Relation Identifier Dialog Box" on page 210. For details on maintaining test object in object repositories, see "How to Maintain Test Objects in Object Repositories" on page 184.

How Visual Relation Identifiers Work

Suppose that you are shown a photograph of a classroom, and are then asked to note identical twins sitting at different desks, and to be able to identify each twin successfully when shown different photographs of the same classroom at a later time.

You are told that one differentiating characteristic is that one twin always carries a blue school bag, and that the other twin always carries a red school bag. You are then also told that each twin has an assigned desk partners, which means that even if the twins sit at a different desk in other photographs, they always sit next to their assigned lab partners.

Therefore, the solution is to identify the twin with the blue school bag in this manner: <twin> is nearest to <twin's desk partner>, and carries <blue school bag>.

QuickTest uses visual relation identifiers in a similar manner, by examining related objects in the application that you are testing. However, because QuickTest compares the related objects based on their relation to the test object to identify as a set of definitions, you would state that <twin's desk partner> is nearest to <twin>, and not vice versa.

How QuickTest Uses Visual Relation Identifiers

- During a run session, QuickTest first attempts to identify the test object using the object's description properties. For details, see "How QuickTest Identifies Objects During a Run Session" on page 128.
- If QuickTest finds one or more objects matching the test object's description, it attempts to identify the object using the visual relation identifier. For details, see "How to Define a Visual Relation Identifier for a Specific Test Object - Use-Case Scenario" on page 190.
- After the visual relation identifier is applied, if no objects or more than one object is found, the visual relation identifier fails, and QuickTest continues to Smart Identification (when defined for that test object class). For details, see "Smart Identification" on page 292.
- After Smart Identification is applied, if no objects or more than one object is found, no ordinal identifiers are used for that test object.
- For general considerations on working with visual relation identifiers, see "Considerations for Working with Visual Relation Identifiers" on page 215.
- For a workflow of the general object identification process, see "Object Identification Process Workflow" on page 149.

Tasks

How to Add a Test Object to an Object Repository

This task describes how to add test objects to local or shared object repositories. This functionality is available in the Object Repository window for the local object repository, and the Object Repository Manager for shared object repositories.

This section also includes:

- "Add test objects to the object repository using the Add Objects to Local or Add Objects option" on page 177
- "Add a test object to the local object repository using the Select Object for Step dialog box or the Step Generator" on page 178
- "Define a new test object" on page 178
- "Add a test object to the object repository using the Object Spy dialog box" on page 179
- "Add a test object to the local object repository using the View/Add Object option from the Active Screen" on page 179
- "Add a test object to the local object repository by inserting a step from the Active Screen" on page 179

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 "Local Copies of Objects from Shared Object Repositories" on page 222.
 - 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.
-

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 be used only 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. In some environments, as you move the pointing hand over your application, the test objects are highlighted. For information on using the pointing hand, see "Tips for Using the Pointing Hand" on page 157.

- 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**. For more information, see "Object Selection Dialog Box" on page 155.

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. For details on the Define Object Filter dialog box, see "Define Object Filter Dialog Box" on page 202.

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 be edited only in the Object Repository Manager.

Add a test object to the local object repository using the Select Object for Step dialog box or the Step Generator

You can 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).

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. For details see, "Define New Test Object Dialog Box" on page 200.

Add a test object to the object repository using the Object Spy dialog box

- 1** Click the **Object Spy** button from QuickTest or the Object Repository Manager.



- 2** Click the **Add Object** button. Depending on where you opened the Object Spy dialog box, the object is added to the local or shared object repository. For more details, see "Object Spy Dialog Box" on page 151.

Add a test object to the local 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. For more information, see "Object Selection Dialog Box" on page 155.
- 5** The Object Properties dialog box opens and displays the default identification properties for the object. For details, see "Object Properties Dialog Box" on page 234.

Add a test object to the local 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**. For more information, see "Object Selection Dialog Box" on page 155.

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).

How to Copy, Paste, Move, or Delete Objects in the Object Repository

The following steps describe how to copy, paste, move, and delete objects in an object repository. This functionality 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:

- "Move an object to a different location within an object repository" on page 181
- "Copy an object to a different location within an object repository" on page 181
- "Move or copy an object without its child objects" on page 181
- "Cut, copy, and paste objects within an object repository" on page 181
- "Cut, copy, and paste objects between shared object repositories" on page 181
- "Copy objects from one shared object repository to another" on page 181
- "Move objects from one shared object repository to another" on page 182
- "Delete an object from the object repository" on page 182
- "Add test objects from a local or shared object repository to your test" on page 182

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.

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.

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.

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.

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.

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.

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 "Local Copies of Objects from Shared Object Repositories" on page 222.

Delete an object from the object repository



In the repository tree, select the object you want to delete and then click the **Delete** button.

Add test objects from a local or shared object repository to your test

You can drag and drop test objects from a shared or local object repository 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.

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).

Note: You cannot drag and drop checkpoint or output value objects from the Object Repository Manager.

For details on adding objects to your test, see:

- "Available Keywords Pane Overview" on page 1320
- "Object Repository Window" on page 237

How to Locate an Object in an Object Repository

The following steps describe how to locate a specific object in your object repository.

- "Find an Object in an Object Repository" on page 183
- "Highlight an Object in Your Application" on page 183
- "Locate an Object from Your Application in the Object Repository" on page 183

Find an Object in an 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. The Find & Replace dialog box opens. For user interface details, see "Find and Replace Dialog Box" on page 206.



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.



Locate an Object from Your Application in the Object Repository

- 1 Make sure your application is open to the correct window or page.
 - 2 Click the **Locate in Repository** button.
- QuickTest is hidden, and the pointer changes into a pointing hand. In some environments, as you move the pointing hand over your application, the test objects are highlighted.
- 3 Use the pointing hand to click the required object in your application. For information on using the pointing hand, see "Tips for Using the Pointing Hand" on page 157.



- 4 If the location you clicked is associated with more than one object, the Object Selection dialog box opens. For details, see "Object Selection Dialog Box" on page 155. The selected object is highlighted in the object repository.

Note: 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.

How to Maintain Test Objects in Object Repositories

The following steps describe different options for maintaining and modifying the test object details of objects in your repositories.

- "Specify an identification property value" on page 185
- "Update identification properties from an object in your application" on page 186
- "Restore the mandatory property set" on page 186
- "Rename test objects" on page 187
- "Add properties to a test object description" on page 187
- "Define a new identification property" on page 188
- "Remove properties from a test object description" on page 188
- "Specify an ordinal identifier" on page 189
- "Define related objects for a specific test object" on page 189

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 "Value Configuration Options Dialog Box" on page 872.

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. For more details, see "Object Repository Window" on page 237.

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. In some environments, as you move the pointing hand over your application, the test objects are highlighted. For more information on using the pointing hand, see "Tips for Using the Pointing Hand" on page 157.
- 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.

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.

Rename test objects

- 1 In the object repository tree of the Object Repository window or Manager, select the test object that you want to rename.
- 2 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. For a list of naming conventions, see the **Test object name** section in "Naming Conventions" on page 1779. Test object names are not case-sensitive.

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.
- 3 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). For details, see "Add Properties Dialog Box" on page 197.



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.

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.



- 1 In the **Test object details** area, click the **Add description properties** button. The Add Properties dialog box opens.



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.



- 2 Click the **Define new property** button. The New Property dialog box opens. For details, see "New Property Dialog Box" on page 199.

Remove properties 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.

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. For details, see "Ordinal Identifier Dialog Box" on page 208.

Define related objects for a specific test object

- 1 In the **Visual Relation Identifier Settings** row of the Object Repository window or Object Properties dialog box, click in the **Value** cell.
- 2 Click the **Browse** button in the cell. The Visual Relation Identifier dialog box opens.
- 3 Set the options for the visual relation identifier. For details, see "Visual Relation Identifier Dialog Box" on page 210.

Results:

- The visual relation identifier is added to the selected test object, and the text in the **Value** cell indicates that a visual relation identifier is defined.
- Any related objects you specified are linked to the test object for which you are using a visual relation identifier. You cannot define visual relations for those objects.
- The **Ordinal identifier** property is disabled in the **Object Details** area of the local or shared object repository, and is not used during the object identification process. However, QuickTest still uses this property during the learn process, when comparing existing objects with the objects to be learned, and therefore the ordinal identifier value should not be manually changed or removed.

For considerations on working with visual relation identifiers, see "Considerations for Working with Visual Relation Identifiers" on page 215.

For a use-case scenario related to this task, see "How to Define a Visual Relation Identifier for a Specific Test Object - Use-Case Scenario" on page 190.

How to Define a Visual Relation Identifier for a Specific Test Object - Use-Case Scenario

This scenario describes the process you would follow to define a visual relation identifier for a specific test object that would otherwise require the use of ordinal identifiers.

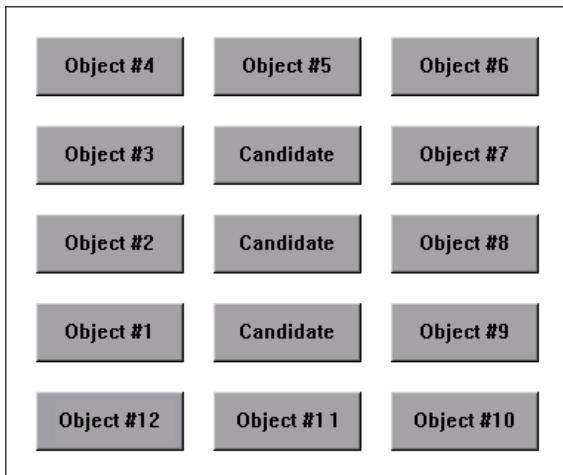
Note: For a task related to this scenario, see "How to Maintain Test Objects in Object Repositories" on page 184.

This scenario includes the following steps:

- "Background" on page 191
- "Access the Visual Relation Identifier dialog box" on page 192
- "Highlight the objects in the application that match the test object's description" on page 193
- "Define the first related test object using horizontal visual relations" on page 193
- "Define the second related object using vertical visual relations" on page 195
- "Define the third related object using distance visual relations" on page 196
- "Results" on page 197

1 Background

- The application you are testing contains three identical instances of the Candidate object.



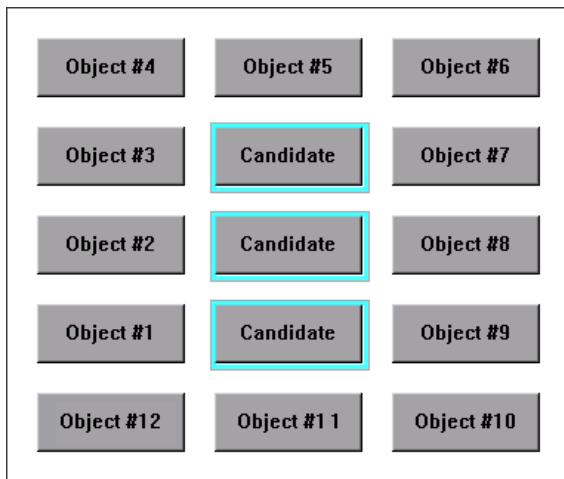
- When QuickTest learned the objects in the application, it assigned an ordinal identifier to each Candidate test object.
- For the purpose of this exercise, Object #1 and Object #9 make up an object pair, which is always to the left and right of the Candidate object to identify. You want to instruct QuickTest to identify the instance of the Candidate object that is located between the Object #1 and Object #9 object pair during every run session, even if the sorting order of the object pairs changes between run sessions.

2 Access the Visual Relation Identifier dialog box

- a** In QuickTest, open the relevant object repository and select the Candidate test object to identify.
- b** Verify that you have selected the correct test object by selecting Object > Highlight in Application, and making sure that the correct object is highlighted in the application.
- c** Open the Visual Relation Identifier dialog box, as described in "Visual Relation Identifier Dialog Box" on page 210.

3 Highlight the objects in the application that match the test object's description

In the Visual Relation Identifier dialog box, click the **Preview** button. This instructs QuickTest to highlight all objects that match the test object description (ignoring the ordinal identifiers). The main QuickTest window is hidden, and each instance of the **Candidate** object in the application is highlighted, including the instance of the test object you want to identify.



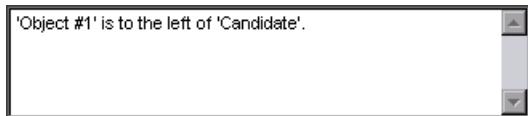
You then click the **Preview** button again to restore the QuickTest window.

4 Define the first related test object using horizontal visual relations

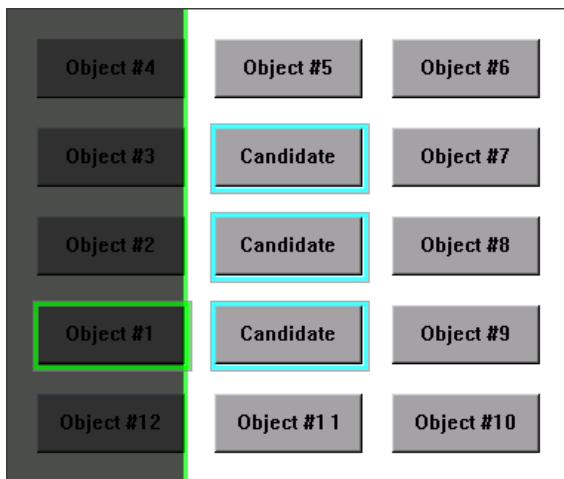
- a In the **Related Objects** area, click the **Add** button. The Select Test Object dialog box opens, enabling you to either select a test object from the object repository, or add an object from the application. For details, see "Select Test Object Dialog Box" on page 513.

For the purpose of this scenario, the first related test object is Object #1, which is located to the left of the **Candidate** object in the application.

- b** In the **Relation Details** area, select the first checkbox and then select **Left** from the drop-down list. The description area displays a summary of the visual relation identifier.



- c** To verify that the visual relation is defined correctly, click the **Preview** button again. The main QuickTest window is hidden, and the visual relation identifier displays the objects that match the test object's description, including the currently defined visual relation. It also highlights the selected related object, and a visual representation of the defined relation details. Since Object #1 is to the left of all three Candidate buttons, all three buttons are still highlighted when you use the **Preview** button.{

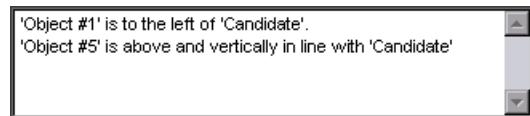


5 Define the second related object using vertical visual relations

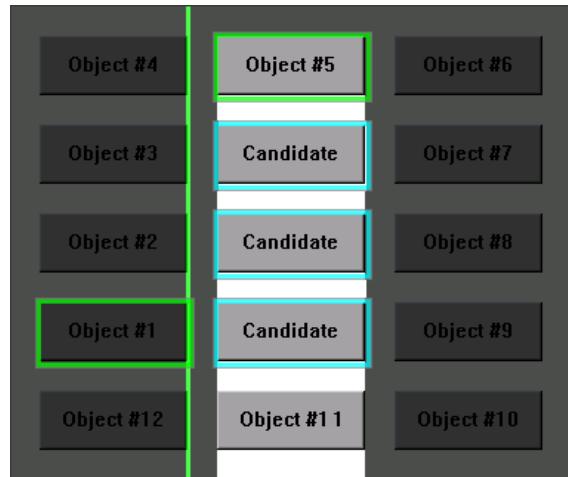
- a In the **Related Objects** area, click the **Add** button. The Select Test Object dialog box opens, enabling you to select or add another object.

For the purpose of this scenario, the second related test object is Object #5, which is located above and vertically in line with the Candidate object.

- b In the **Relation Details** area, select the second check box. From the drop-down list select **Above**, and then select the **In line (vertically)** checkbox. The description area displays a tooltip of all the selected visual relations.



- c To verify that the visual relations are defined correctly, click the **Preview** button again. Since Object #5 is above all three Candidate objects, all three are still highlighted. That means you still need to select another related object to create a visual relation identifier that uniquely identifies your object.

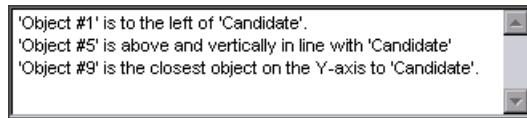


6 Define the third related object using distance visual relations

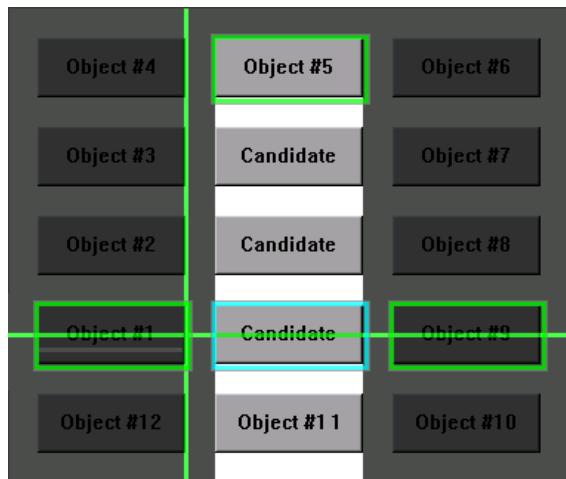
- a In the **Related Objects** area, click the **Add** button. The Select Test Object dialog box opens, enabling you to select another test object.

For the purpose of this scenario, the third related test object is Object #9, which is the closest object to the right of the Candidate object.

- b In the **Relation Details** area, select the third checkbox and then select **Closest on the Y-axis** from the drop-down list. The description area displays an updated summary of the visual relation identifier.



- c To verify that the visual relations are defined correctly, click the **Preview** button again. You can now see that this third related object enables QuickTest to uniquely identify the correct object.



7 Results

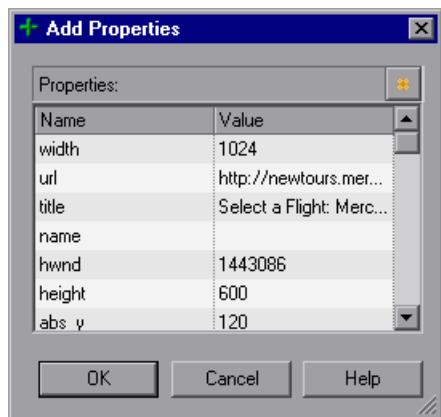
After you finish defining all of the necessary visual relations:

- The desired Candidate object is the only object in the application that is identified when you use **Preview**.
- QuickTest can now correctly identify the desired Candidate object during every run session, even if the user interface changes, as long as the Candidate object maintains its relative location to the three related objects you defined.
- The **Ordinal Identifier** property is disabled in the Object Repository Manager or window.

Reference

Add Properties Dialog Box

This dialog box enables you to add properties to the test object description of test objects.



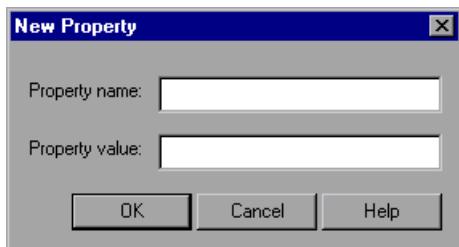
To access	From the Object Repository Manager or window, in the Test object details area, click the Add description properties button  .
Important information	<ul style="list-style-type: none"> ➤ Values for all properties are displayed only if the application that contains the object is currently open. 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. ➤ 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 168.
Relevant tasks	"How to Maintain Test Objects in Object Repositories" on page 184
See also	"Maintaining Identification Properties - Overview" on page 167

User interface elements are described below:

UI Elements	Description
	Define new property. Opens the New Property dialog box. For details see "New Property Dialog Box" on page 199.
<Properties list>	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.

 **New Property Dialog Box**

This dialog box enables you to define a new identification property for a test object.



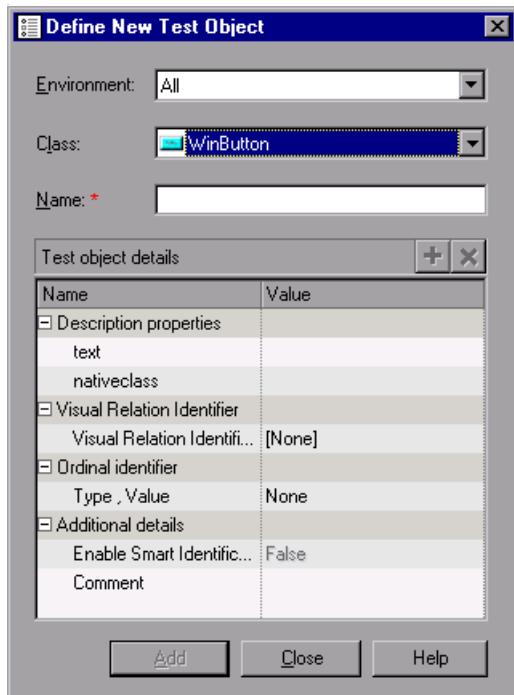
To access	Click the Define new property button  in the Add Properties Dialog Box.
Important information	You must enter a valid identification property in the Property value edit box. If you enter an invalid property and then select it to be part of the object description, your run session will fail.
Relevant tasks	"How to Maintain Test Objects in Object Repositories" on page 184

User interface elements are described below:

UI Elements	Description
Property name	The property name.
Property value	The value for the property.

Define New Test Object Dialog Box

This dialog box enables you to define test objects in your object repository that do not yet exist in your application.



To access	From the Object Repository Manager or Object Repository window, select Object > Define New Test Object .
Relevant tasks	"Adding and Deleting Test Objects in a Local or Shared Object Repository" on page 160
See also	"Defining New Test Objects" on page 162

User interface elements are described below:

UI Elements	Description
Environment	<p>The list of available environments. The test object classes associated with the selected environment are displayed in the Class box.</p> <p>Note:</p> <ul style="list-style-type: none"> ▶ 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 <i>HP QuickTest Professional Add-ins Guide</i>. ▶ The Environment list might also include additional environments for which you or a third party developed support using QuickTest extensibility.
Class	Select the class of the test object you want to define.
Name	Enter a name for the new test object. After you enter a name, the Test object details area is enabled.
Test object details	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 - Overview" on page 167.

Define Object Filter Dialog Box

This dialog box enables you to define which objects should be learned (while using the **Navigate and Learn** option or the **Add Objects** option). 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.



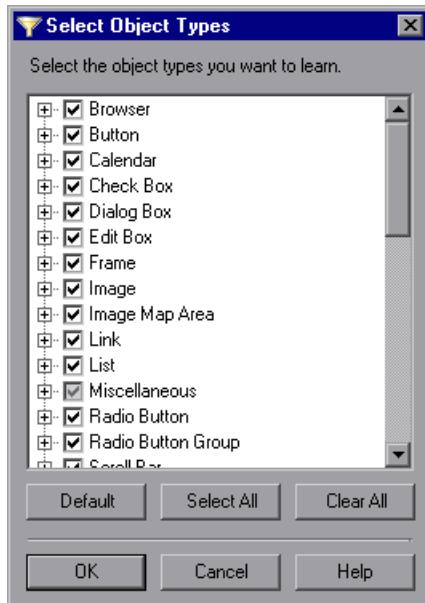
To access	<p>Do one of the following:</p> <ul style="list-style-type: none"> ➤ Add a test object to the object repository using the Add Objects to Local from the Object Repository window. ➤ Add a test object to the object repository using the Add Objects option from the Object Repository Manager. <p>For details, see "Adding and Deleting Test Objects in a Local or Shared Object Repository" on page 160.</p>
Relevant tasks	<p>"Adding and Deleting Test Objects in a Local or Shared Object Repository" on page 160</p>

User interface elements are described below:

UI Elements	Description
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 "Select Object Types Dialog Box" on page 204.

Select Object Types Dialog Box

This 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).



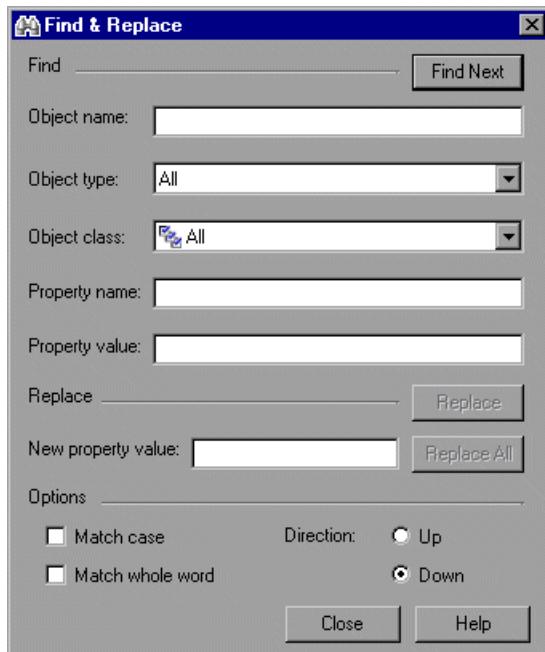
To access	Click the Select button in the Define Object Filter Dialog Box.
Important information	<ul style="list-style-type: none"> ▶ When you define an object filter, it is automatically saved for future add object operations (performed from both the Navigate and Learn option and the Add Objects option). ▶ 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 <i>HP QuickTest Professional Add-ins Guide</i>.
Relevant tasks	"Adding and Deleting Test Objects in a Local or Shared Object Repository" on page 160
See also	"Adding Test Objects to the Local Object Repository from the Active Screen" on page 161

User interface elements are described below:

UI Elements	Description
Default	Restores 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.
Select All	Selects all the check boxes.
Clear All	Clears all the check boxes.

Find and Replace Dialog Box

This dialog box enables you to find an object, property, or property value in the object repository.



To access	Click the Find & Replace button  in the Object Repository Manager or Object Repository window.
Important information	<ul style="list-style-type: none"> ➤ The functionality in this dialog box is available in the Object Repository window for the local object repository, and the Object Repository Manager for shared object repositories. ➤ You can search using any or all of the criteria in the Find and Replace dialog box. ➤ The Find and Replace dialog box can find checkpoint and output values only 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.

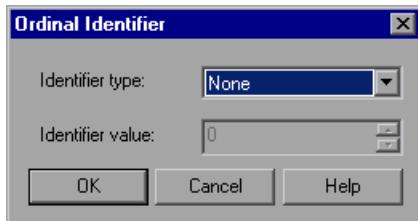
Relevant tasks	"How to Locate an Object in an Object Repository" on page 183
See also	"Locating Objects" on page 165

User interface elements are described below:

UI Elements	Description
Object name	The name or partial name of the object you want to find.
Object type	The type of object you want to find, for example, Button. 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	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	The name or partial name of the property you want to find.
Property value	The property value or partial property value you want to find.
New property value	The new property value if you specified a property value and want to replace it with a different value.
Options	<ul style="list-style-type: none"> ▶ Match case. Select this option if you want the search to distinguish between upper and lower case letters. ▶ Match whole word. Select this option if you want the search to find only complete words that exactly match the single word you entered. ▶ Direction. The direction you want to search.

 **Ordinal Identifier Dialog Box**

This dialog box enables you to define an ordinal identifier for a test object.



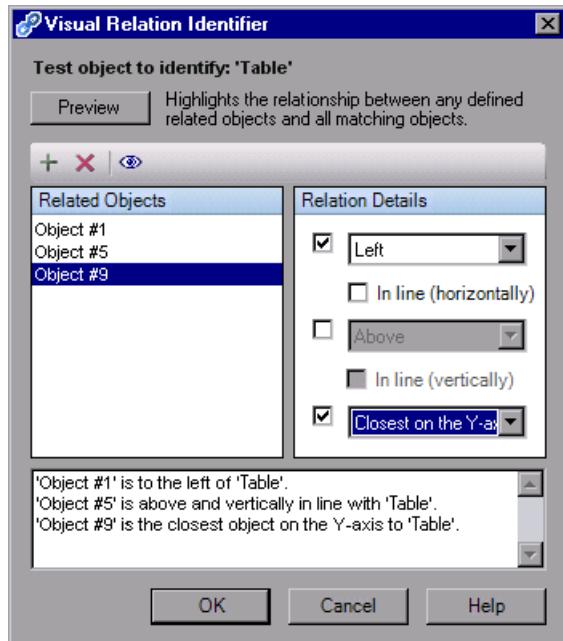
To access	1 From the Object Repository Manager or window, in the Test object details area, click in the Value column of the Type, Value row in the Ordinal identifier section. 2 Click the Browse button in that cell.
Relevant tasks	"How to Maintain Test Objects in Object Repositories" on page 184
See also	"Maintaining Identification Properties - Overview" on page 167

User interface elements are described below:

UI Elements	Description
Identifier type	<ul style="list-style-type: none">▶ Index. The order in which the object appears within the parent window, frame, or dialog box relative to other objects with an otherwise identical description.▶ Creation Time. (Browser objects only). 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.
Identifier value	The numeric value of the ordinal identifier.

Visual Relation Identifier Dialog Box

This dialog box enables you to define related objects for the visual relation identifier of a specific test object. QuickTest uses this identifier in addition to the test object description during the run session to identify the test object.



To access	<ol style="list-style-type: none"> 1 In the Object Repository window, or the Object Properties dialog box, select the test object you want to identify. 2 In the Visual Relation Identifier Settings row of the Object Repository window or Object Properties dialog box, click in the Value cell. 3 Click the Browse button in the text box.
Important information	<p>"Considerations for Working with Visual Relation Identifiers" on page 215</p>

Relevant tasks	<ul style="list-style-type: none"> ▶ "How to Maintain Test Objects in Object Repositories" on page 184 ▶ "How to Define a Visual Relation Identifier for a Specific Test Object - Use-Case Scenario" on page 190
See also	<ul style="list-style-type: none"> ▶ "Visual Relation Identifiers" on page 174 ▶ "How QuickTest Identifies Objects During a Run Session" on page 128 ▶ "How QuickTest Interprets Horizontal and Vertical Visual Relations" on page 214

User Interface Elements

User interface elements are described below (unlabeled elements are shown in angle brackets):

UI Elements	Description
Test object to identify	The name of the test object as it appears in the object repository (read-only).
	Add. Opens the Select Test Object dialog box, which enables you to add a test object to the Related Objects list, either from the object repository or directly from the application. Any object you add from the application is automatically added to the object repository. For details, see "Select Test Object Dialog Box" on page 513.
	Remove. Removes the selected related object from the Related Objects list. Note: Removing the related object from the list does not remove the test object remains from the object repository.

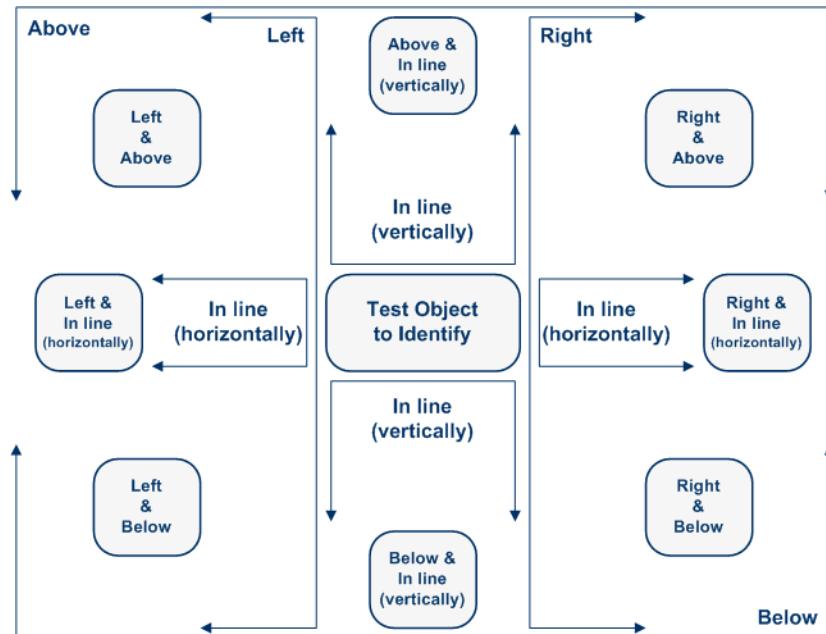
UI Elements	Description
Preview	<p>Highlights the visual relation between all related objects and the objects matching the test object to identify (main QuickTest window is hidden). While in Preview mode, the text to the right of the button displays the number of matching objects. This enables you to test the relation details you are defining without closing the dialog box or running the steps.</p> <p>Note:</p> <ul style="list-style-type: none"> ▶ During the Preview process, the main QuickTest window is hidden. Click the Preview button again to restore the QuickTest window. ▶ While the Preview button is pressed, you can change the relation details and preview the changes without returning to the QuickTest window. ▶ If QuickTest is unable to perform the Preview operation, a message box opens. This could be the result of one of the following scenarios: <ul style="list-style-type: none"> ▶ One or more related objects cannot be found in the object repository. ▶ One or more related objects are already used in a visual relation identifier for another test object. ▶ One or more related objects cannot be uniquely identified in the object repository.
Related Objects	<p>The list of related objects.</p> <p>Tooltip. The tooltip for each related object displays the full name.</p> <p>Note: If the related object cannot be found in the object repository, an indicating icon is displayed next to the name of that related object, and a tooltip is displayed when you hover the cursor over the icon.</p>
Relation Details	<p>The details of the visual relation for the selected related object. You can select a value from one or more of the relation categories.</p> <p>For details on the available visual relation categories, see "Visual Relation Categories (Relation Details Area)" on page 213.</p>
<Relation description>	Textual description of the currently defined visual relations.

Visual Relation Categories (Relation Details Area)

Category	Description
Horizontal	<p>Enables you to define related objects according to their horizontal location relative to the object to identify. The following options are available:</p> <ul style="list-style-type: none"> ➤ Left ➤ Right ➤ In line (horizontally) <p>For details on this option, see "How QuickTest Interprets Horizontal and Vertical Visual Relations" on page 214.</p>
Vertical	<p>Enables you to define related objects according to their vertical location relative to the object to identify. The following options are available:</p> <ul style="list-style-type: none"> ➤ Above ➤ Below ➤ In line (vertically) <p>For details on this option, see "How QuickTest Interprets Horizontal and Vertical Visual Relations" on page 214.</p>
Distance and hierarchy	<p>Enables you to define related objects according to their distance or hierarchical location relative to the object to identify. The following options are available:</p> <ul style="list-style-type: none"> ➤ Closest on the X-axis ➤ Closest on the Y-axis ➤ Closest on both axes ➤ Contains

How QuickTest Interprets Horizontal and Vertical Visual Relations

The following diagram illustrates the way QuickTest interprets horizontal and vertical visual relations. It also shows the boundaries that are used for determining **in line** related objects.

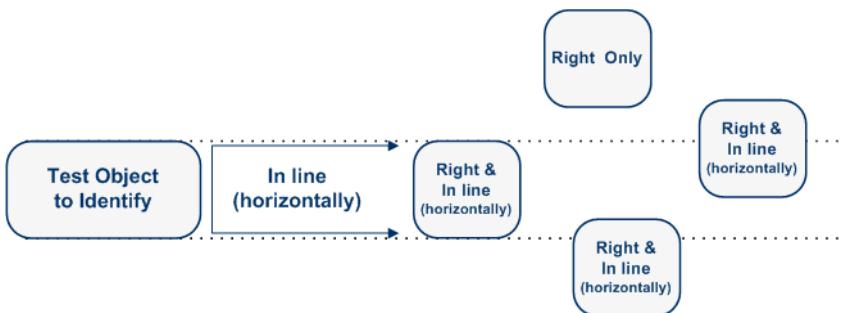


How QuickTest Identifies In Line Related Objects

When you select a related object as a horizontal and/or vertical relation in the Visual Relation Identifier dialog box, you can also fine-tune that definition by indicating that it is **in line** with the test object to identify.

QuickTest identifies the related object as **in line** even if the area of the related object surface is only partially in line with the test object.

The following example illustrates how QuickTest identifies related objects that are in line with the test object to identify.



Considerations for Working with Visual Relation Identifiers

Consider the following when working with visual relation identifiers:

- If you define a visual relation identifier for a test object, then that test object's ordinal identifier (if it exists) is not used during the test object identification process (when running steps, highlighting objects in the application, etc.). To indicate this, the **Ordinal identifier** option for that test object is disabled in the Object Repository window.
- If you add a related object that was not previously in the object repository, then the test object for the related object is added to that object repository, even if you click **Cancel** in the Visual Relation Identifier dialog box.

Workaround: If you do not need that test object, manually delete it from the object repository.

- Visual relation identifiers are used only during a run session, or when identifying objects in the application (for example, from the Object Repository window or the Object Spy). Therefore, even if you define related objects for a specific test object, if QuickTest re-learns the object, it uses only the identification properties that are defined in the Object Identification dialog box for that test object class, as well as an ordinal identifier (if needed). This may cause QuickTest to learn the same object more than once.

Example: If you learned the Object1 test object with an ordinal identifier property value of 1, and then you manually remove the ordinal identifier or otherwise significantly modify the description properties (because you are depending on the visual relation identifiers to fine tune the description), then the next time a learn session includes this object, QuickTest will not recognize Object1 as the same object and will learn a new test object with the name Object1_1.

- QuickTest uses visual relation identifiers only when one or more objects match the test object's description properties during the identification process. If no objects in the application match the test object's description properties, then the visual relation identifier you defined is ignored, and QuickTest continues to Smart Identification (if defined for that test object class).

For details on the complete flow QuickTest uses to identify objects, see "Object Identification Process Workflow" on page 149.

- A test object cannot be used as a related object to itself.
- The following test objects cannot act as related objects for another test object's visual relation identifier:
 - A test object that has a visual relation identifier.
 - A child test object for one of its parent objects.
- If you delete a test object (A) that is used in the visual relation identifier for another test object (B), you must make sure to remove the deleted test object A from the **Related Object** list for test object B.
- Visual relation identifiers are not supported for WebService test objects.
- You can retrieve or replace the visual relation identifier settings of a specific test object during a run session using the **VisualRelationsCollection** object. For details, see the **VisualRelationsCollection** object in the *HP QuickTest Professional Object Model Reference*.

- After performing a **Preview** operation, if you close your application while the Visual Relation Identifier dialog box is open and then open your application again, no objects are highlighted when you press **Preview** again.

Workaround: After opening your application, close the Visual Relation Identifier dialog box, select a different test object in the object repository, return to the test object you want to identify, and open the Visual Relation Identifier dialog box again.

Troubleshooting and Limitations - Managing Test Objects

For troubleshooting and limitations for learning objects, see "Learning objects, and recording and running steps" on page 1083.

5

Working with Your Test's Object Repositories

This chapter includes:

Concepts

- Object Repository Window - Overview on page 220
- Exporting Local Objects to a Shared Object Repository on page 221
- Local Copies of Objects from Shared Object Repositories on page 222
- Repository Parameter Value Mappings on page 223
- Working with Test Objects During a Run Session on page 225

Tasks

- How to Export Local Objects to a Shared Object Repository on page 226
- How to Copy an Object to the Local Object Repository on page 227
- How to Modify Identification Properties During a Run Session on page 228

Reference

- Associate Repositories Dialog Box on page 229
- Map Repository Parameters Dialog Box on page 232
- Object Properties Dialog Box on page 234
- Object Repository Window on page 237

Troubleshooting and Limitations - Object Repositories on page 247

Concepts

Object Repository Window - Overview

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 read-only (gray).

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

Note: You modify objects in a shared object repository using the Object Repository Manager. For details, see Chapter 6, "Shared 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 details, see "Local Copies of Objects from Shared Object Repositories" on page 222.

Exporting Local Objects to a Shared Object Repository

You can export all of the test objects, checkpoint objects, and output value objects contained in an action's local object repository to a 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 "Repository Parameter Value Mappings" on page 223). For details on repository parameters, see "Working with Repository Parameters" on page 252.

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 details, see Chapter 9, "Object Repository Merge Tool."

Local Copies of Objects from Shared Object Repositories

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

You can create a local copy of any object stored in a shared object repository that is associated with the action currently displayed in the in the object repository tree.

Copying an object to the local repository is useful, for example, if you want to modify an object in the current action without affecting other actions that use the shared object repository.

When you create a local copy of an object and modify it in the Object Repository window, the changes you make affect only the action in which you make the change. Conversely, if you modify the object in the shared object repository using the Object Repository Manager, the changes you make are reflected in all actions that use the shared object repository. However, if you modify an object in a shared object repository, and a copy of the object (with the same name) exists in the local repository, your changes do not affect the local copy of the object in your action.

During a run session, QuickTest uses the object in the local object repository to identify the object in your application. This is because the action's local object repository has higher priority than any shared object repository associated with the action.

Considerations for Copying an Object to the Local Object Repository

- 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.

Repository Parameter Value Mappings

You use **repository parameters** to specify that certain property values of an object in a shared object repository should be parameterized, but that the actual values should be defined in each action that is associated with the shared object repository. For details on repository parameters, see "Working with Repository Parameters" on page 252.

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.

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. The Object Repository window is read-only during a run session.

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 details on programmatic descriptions, see "Programmatic Descriptions" on page 946.

Tasks

How to Export Local Objects to a Shared Object Repository

This task describes how to export local objects to a shared object repository.

This task includes the following steps:

- "Prerequisites" on page 226
- "Select an action" on page 226
- "Export the local objects" on page 226
- "Results" on page 227

1 Prerequisites

-  **a** Open the test that has the local objects you want to export.
- b** Open the Object Repository window by selecting **Resources > Object Repository** or clicking the **Object Repository** button.

2 Select an action

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

3 Export the local objects

Select **File > Export Local Objects**, or **File > Export and Replace Local Objects**. The Save Shared Object Repository dialog box opens. For details, see "Save <Resource> Dialog Box" on page 417.

4 Results

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, 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.

How to Copy an Object to the Local Object Repository

This task describes how to copy an object from a shared object repository to the local object repository.

- 1 Open the test containing the action to whose local object repository you want to copy the object.
- 2 In the object repository tree of the Object Repository window, select the action associated with the shared object repository containing the object you want to copy.
- 3 Select the object that you want to copy to the local object repository. (Objects in a shared object repository are read-only.) You can select multiple objects as long as the selected objects have the same parent object.
- 4 Select **Object > Copy to Local** or right-click the objects and select **Copy to Local**. The objects (and parent objects, if any) are copied to the local object repository and are made editable.

How to Modify 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:

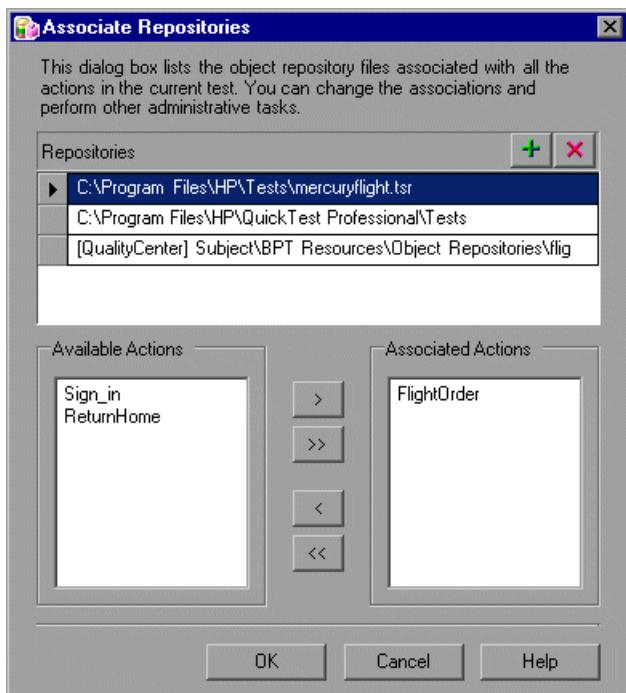
Object(*description*).SetTOProperty *Property, Value*

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

Reference

Associate Repositories Dialog Box

This dialog box enables you to associate one or more shared object repositories with one or more actions in a test. You can view the shared object repositories associated to each of the actions in the current test, and remove object repository associations from selected actions, or from all actions in the test.



To access	<p>Do one of the following:</p> <ul style="list-style-type: none"> ➤ In the QuickTest main window, select Resources > Associate Repositories. ➤ In the Object Repository window, select Tools > Associate Repositories. ➤ In the Object Repository window, click the Associate Repositories button .
Important information	<ul style="list-style-type: none"> ➤ You prioritize the object repositories using the Associated Repositories tab of the Action Properties dialog box (and not the Associate Repositories dialog box). For details, see "Associated Repositories Tab (Action Properties Dialog Box)" on page 567. ➤ You can associate, remove, prioritize, and view the properties of shared object repositories in the Resources pane. For details, see "Resources Pane User Interface" on page 1391.

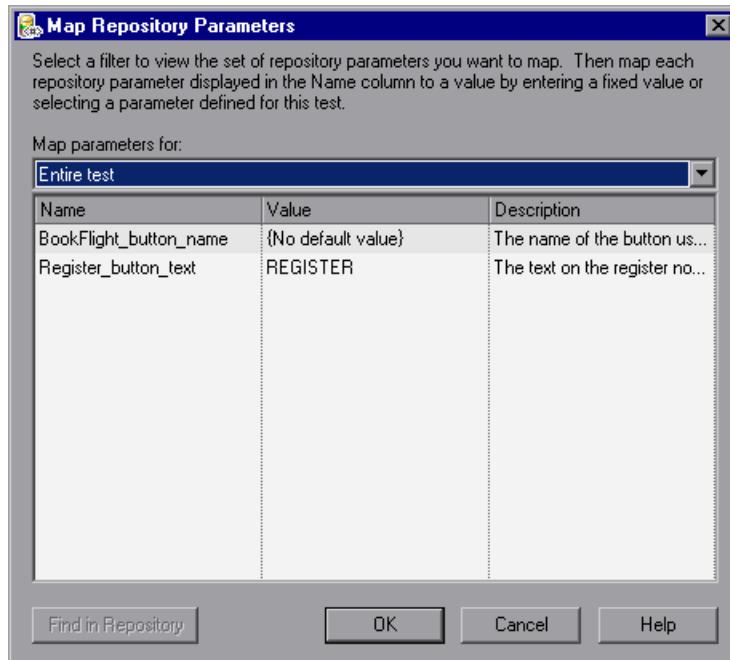
User interface elements are described below (unlabeled elements are shown in angle brackets):

UI Elements	Description
	Add Repository. Enables you to browse and add shared repositories to be associated with actions. The new object repository is displayed at the bottom of the Repositories list.
	Remove Repository. Enables you to remove shared repositories and their associations from your test.
	< Add Action >. Moves the selected action in the Available Actions list to the Associated Actions list. You can also double-click the action name to move it to the Associated Actions list.
	< Add All Actions >. Moves all the actions in the Available Actions list to the Associated Actions list. You can also use the Shift and/or Control keys to select multiple actions.

UI Elements	Description
	< Remove Action >. Moves the selected action in the Associated Actions list to the Available Actions list. You can also double-click the action name to move it to the Available Actions list.
	< Remove All Actions >. Moves all the actions in the Associated Actions list to the Available Actions list. You can also use the Shift and/or Control keys to select multiple actions.
Repositories	<p>The list of available shared repositories.</p> <p>Tip: You can modify the name or path of an associated shared object repository without changing the current associations, as follows:</p> <ul style="list-style-type: none"> ▶ Click a shared object repository and then browse to select a different shared object repository. ▶ Click the shared object repository and modify the name or path directly.
Available Actions	The list of actions to associate to the selected shared repository.
Associated Actions	The list of actions currently associated with the selected shared repository.

Map Repository Parameters Dialog Box

This dialog box enables you to map values for the repository parameters that are used in shared object repositories that are associated with your action. This enables you to specify the property values used to identify the test object during a run session.



To access	Use one of the following: <ul style="list-style-type: none"> ➤ Select Resources > Map Repository Parameters ➤ In the Missing Resources pane, double-click the Repository Parameters row (Relevant only if your test contains unmapped repository parameters (repository parameters without a default value)).
See also	"Repository Parameter Value Mappings" on page 223

User interface elements are described below:

UI Elements	Description
Map parameters for	<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 that are not mapped to 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	<p>The name of the repository parameter.</p>
Value	<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:</p> <ul style="list-style-type: none"> ▶ Enter a new constant value or modify an existing constant value by typing directly in the Value cell. ▶ Tip: You can also enter a constant value in the Value Configuration Options dialog box by clicking the parameterization button . For details, see "Value Configuration Options Dialog Box" on page 872. ▶ Parameterize the value by clicking in the Value cell of the relevant parameter and then clicking the parameterization button . You can parameterize the value using a Data Table (Global sheet only), random number, environment, or test parameter. For details, see "Value Configuration Options Dialog Box" on page 872. ▶ 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 . The default value, if any, that was defined in the Add Repository Parameter dialog box is displayed in the cell. For details, see "Add Repository Parameter Dialog Box" on page 280.

UI Elements	Description
Description	A textual description of the parameter, if any.
Find in Repository	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.

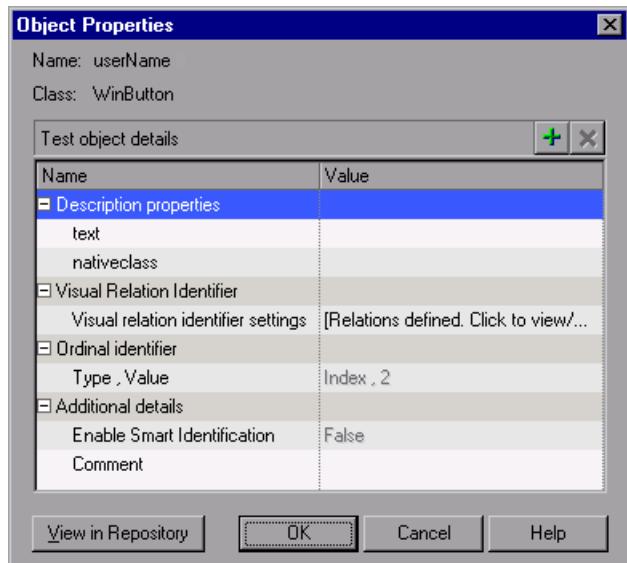
Object Properties Dialog Box

This dialog box enables you to:

- ▶ View identification properties and values for objects in your test steps or in the Active Screen, as well as other object details.
- ▶ 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 details, see "Maintaining Identification Properties - Overview" on page 167.

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.



To access	<ul style="list-style-type: none"> ➤ 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.
------------------	--

User interface elements are described below:

UI Elements	Description
	Add description properties button.
	Remove selected description properties button.

UI Elements	Description
Name	The name that QuickTest assigns to the object. You can change the name of a object in the local object repository. For details, see "Renaming Test Objects" on page 170.
Class	The class of the object.
Description properties	<p>The properties and property values used to identify the object during a run session.</p> <p>Tips:</p> <ul style="list-style-type: none"> ▶ For details on adding properties to or removing properties from the test object description, see "Add Properties Dialog Box" on page 197. ▶ For details on specifying a property value as a constant or parameterizing a value, see "Specifying or Modifying Property Values" on page 168.
Visual relation identifier	<p>A set of definitions that enable you to identify the object in the application according to its neighboring objects in the application. When this option is defined and enabled, the Ordinal identifier option is disabled. For details, see "Visual Relation Identifier Dialog Box" on page 210.</p> <p>Note: If one or more related objects cannot be found in the object repository, indicating text is displayed in the cell.</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 details, see "Ordinal Identifier Dialog Box" on page 208.</p> <p>Note: If a visual relation identifier is defined for a specific test object, this option is disabled. For details, see "Considerations for Working with Visual Relation Identifiers" on page 215.</p>

UI Elements	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. 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 details, see "Smart Identification" on page 292. ➤ Comment. Enables you to add textual information about the test object.
View in Repository	<p>Opens the Object Repository window and displays the identification properties and values for the selected object.</p>
Add to Repository	<p>Displayed only for an object that is not in the object repository. (Available only when you open this dialog box from the Active Screen.)</p> <p>Adds this object to the action's local object repository. After this object is added, the Add to Repository button changes to View in Repository.</p>

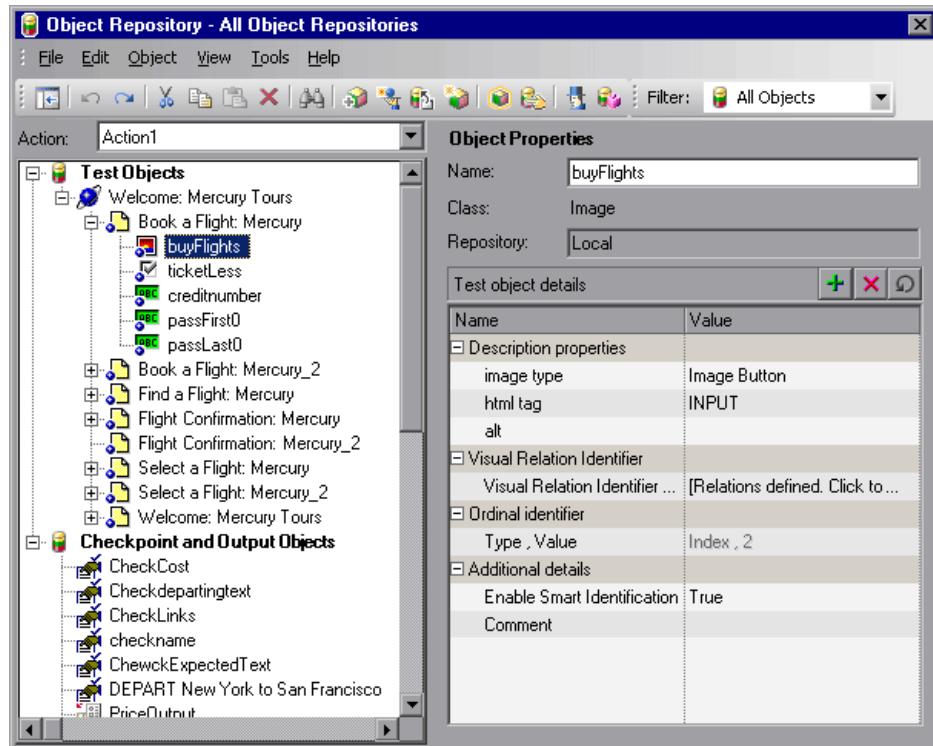
Object Repository Window

This window enables you to manage identification properties and object repository associations for your action.

You can use the Object Repository window to:

- View the object description of any object in the repository (in local and shared object repositories).
- Modify local objects and their properties.
- Add test objects to your local object repository.
- Drag and drop test objects 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.

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).



To access	<p>Do one of the following:</p> <ul style="list-style-type: none"> ▶ QuickTest main window: Click the Object Repository button , or choose Resources > Object Repository ▶ Resources pane: Double-click an object repository, or right-click an object repository and choose Open Repository ▶ Test Flow pane: Right-click an action and choose Object Repository ▶ Available Keywords pane: Right-click an object in the repository and choose Open Resource
Important information	<ul style="list-style-type: none"> ▶ Local objects are editable (black). Objects from a shared object repository are read-only format (gray). ▶ You can modify checkpoint and output value details for objects saved in the local object repository. ▶ You cannot drag and drop checkpoint and output objects from the Object Repository window to your testing document. ▶ You can copy an object from a shared object repository to the local object repository, and then modify it. ▶ Test objects for environments that are not installed/loaded with QuickTest are displayed with a question mark icon. ▶ The Object Repository window is read-only during a record or run session.

Relevant tasks	<p>Primary tasks:</p> <ul style="list-style-type: none"> ➤ "If the Active Screen is not displayed, select View > Active Screen or click the Active Screen toolbar button to display it." on page 179 ➤ "How to Define a Visual Relation Identifier for a Specific Test Object - Use-Case Scenario" on page 190 <p>Related tasks:</p> <ul style="list-style-type: none"> ➤ "How to Export Local Objects to a Shared Object Repository" on page 226 ➤ "How to Copy an Object to the Local Object Repository" on page 227
See also	<ul style="list-style-type: none"> ➤ For an overview of this window, see "Object Repository Window - Overview" on page 220. ➤ For details on dragging and dropping test objects from other locations, see "Available Keywords Pane Overview" on page 1320 and "How to Manage Objects in Shared Object Repositories" on page 263. ➤ For details on modifying the properties of a test object during a run session, see "Working with Test Objects During a Run Session" on page 225. ➤ For details on viewing and modify object properties from other locations, see "Maintaining Identification Properties - Overview" on page 167.

User interface elements are described in the following sections:

- "Object Repository Window - Edit Toolbar" on page 241
- "Object Repository Window - Filter Toolbar" on page 243
- "Object Repository Window Options" on page 244
- "Object Repository Window - Object Details Area" on page 245

Object Repository Window - Edit Toolbar

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 details, see "If the Active Screen is not displayed, select View > Active Screen or click the Active Screen toolbar button to display it." on page 179.
	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 details, see "If the Active Screen is not displayed, select View > Active Screen or click the Active Screen toolbar button to display it." on page 179.
	Copy	Copies the selected object from the object repository tree into the clipboard. For details, see "If the Active Screen is not displayed, select View > Active Screen or click the Active Screen toolbar button to display it." on page 179.
	Delete	Deletes the selected object from the object repository tree.
	Find & Replace	Finds and replaces an object in the object repository. For details, see "Find and Replace Dialog Box" on page 206.

Button	Name	Description
	Add Objects to Local	Adds an object to the local object repository. For details, see "Adding and Deleting Test Objects in a Local or Shared Object Repository" on page 160.
	Update from Application	Updates the identification properties from an object in the application. For details, see "Updating Identification Properties from an Object in Your Application" on page 168.
	Define New Test Objects	Defines a new test object. For details, see "Define New Test Object Dialog Box" on page 200.
	Highlight in Application	Highlights the selected object in the object repository tree, in the application. For details, see "Highlighting an Object in Your Application" on page 166.
	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 details, see "Locating a Test Object in the Object Repository" on page 166.
	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. You can also add an object to the local object repository and highlight an object in the application. For details, see "Object Spy Dialog Box" on page 151.
	Associate Repositories	Enables you to manage the shared object repository associations of your action. For details, see "Associate Repositories Dialog Box" on page 229.

Object Repository Window - Filter Toolbar

UI Element	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>

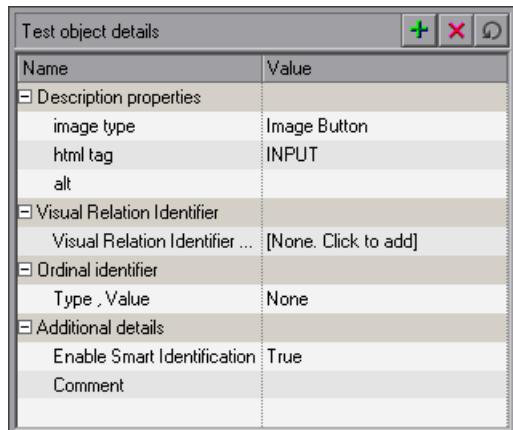
Object Repository Window Options

User interface elements are described below (unlabeled elements are shown in angle brackets):

UI Elements	Description
Action	Enables you to select the action whose objects you want to view.
<Object repository tree>	<p>Displays all of the test objects, checkpoint objects, and output objects in the local and shared object repositories associated with in the selected action.</p> <p>You can filter the objects shown in the object repository tree. For details, see "Object Repository Window - Filter Toolbar" on page 243.</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 details, see "Associated Repositories Tab (Action Properties Dialog Box)" on page 567.</p>
Name	The name that QuickTest assigns to the object. You can change the name of a object in the local object repository. For details, see "Renaming Test Objects" on page 170.
Class	The class of the object.
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 area>	<p>Displays one of the following:</p> <ul style="list-style-type: none"> ► The properties and property values used to identify a test object during a run session ► The properties of a checkpoint or output object. <p>For details, see the "Object Repository Window - Object Details Area" below.</p>

Object Repository Window - 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.



For test objects:

UI Elements	Description
Description properties Tips: <ul style="list-style-type: none"> ➤ For details on adding properties to or removing properties from the test object description, see "Add Properties Dialog Box" on page 197. ➤ For details on specifying a property value as a constant or parameterizing a value, see "Specifying or Modifying Property Values" on page 168. 	The properties and property values used to identify the object during a run session.

UI Elements	Description
Visual relation identifier	<p>A set of definitions that enable you to identify the object in the application according to its neighboring objects in the application. When this option is defined and enabled, the Ordinal identifier option is disabled. For details, see "Visual Relation Identifier Dialog Box" on page 210.</p> <p>Note: If one or more related objects cannot be found in the object repository, indicating text is displayed in the cell.</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 details, see "Ordinal Identifier Dialog Box" on page 208.</p> <p>Note: If a visual relation identifier is defined for a specific test object, this option is disabled. For details, see "Considerations for Working with Visual Relation Identifiers" on page 215.</p>
Additional details	<p>Contains the following options:</p> <ul style="list-style-type: none"> ➤ Enable Smart Identification. Enables you to select True or False to specify whether QuickTest should use Smart Identification to identify the test object during the run session if it is not able to identify the object using the test object description. 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 details on Smart Identification, see "Smart Identification" on page 292. ➤ Comment. Enables you to add textual information about the test object.

For checkpoints: The object details area contains the same information as the Checkpoint Properties dialog box. For details, see "Checkpoint Properties Dialog Box" on page 609.

For output objects: The object details area contains the same information as the Output Value Properties dialog box. For details, see "Output Value Properties Dialog Box" on page 803.

Troubleshooting and Limitations - Object Repositories

This section describes troubleshooting and limitations for working with object repositories.

- If you modify the name of a test object in the Object Repository while your test script contains a syntax error, the new name is not updated correctly within your test steps.

Workaround: Clear the **Automatically update test and component steps when you rename test objects** check box (**Tools > Options > General** node) and perform the renames in the steps manually (recommended) or solve the syntax error, and then close and reopen the document in QuickTest to display the renamed objects in your steps.

- If you use the Export and Replace Local Objects option for an object repository that contains action parameters, the created repository parameters are mapped to test parameters instead of action parameters.

Workaround: Manually adjust the mapping in the exported object repository.

6

Shared Object Repositories

This chapter includes:

Concepts

- Shared Object Repositories Overview on page 250
- Considerations for Working with Shared Object Repositories on page 255

Tasks

- How to Manage Shared Object Repositories on page 258
- How to Manage Objects in Shared Object Repositories on page 263

Reference

- Object Repository Manager Main Window on page 266
- Manage Repository Parameters Dialog Box on page 278

Concepts

Shared Object Repositories Overview

A shared object repository contains 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 shared object repositories.

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

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.

Advantages of Shared Object Repositories

- You can use the same shared object repository with multiple actions, instead of saving objects directly with an action in a local object repository. This enables them to be accessed from multiple actions.
- You can use multiple shared object repositories with each action.
- If your shared object repositories are stored in Quality Center, you can apply version control to them. For details, see "Version Control in Quality Center 10.00 or HP ALM" on page 1695.

For details on associating shared object repositories, see "How to Modify Associations Between Resources and Your Test or Action" on page 1386.

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, using the Object Repository window. For details on local object repositories, see Chapter 4, "Managing Test Objects in Object Repositories."

This section also includes:

- "Importing and Exporting Shared Object Repositories Using XML" on page 251
- "Working with Repository Parameters" on page 252
- "Managing Shared Object Repositories Using Automation" on page 253

Importing and Exporting Shared Object Repositories Using XML

You can import and export shared object repositories from and to XML files. XML provides a structured, accessible format that enables you to make changes to shared object repositories using the XML editor of your choice and then import them back into QuickTest. You can view the required format for the shared 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 shared object repository.

Note: QuickTest uses a defined XML schema for shared object repositories. You must follow this schema when creating or modifying shared object repository files in XML format. 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 import and export files either from and to the file system or a Quality Center project (if QuickTest is connected to Quality Center).

For details, see:

- "Import a shared object repository from XML" on page 261
- "Export a shared object repository to XML" on page 262



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 shared 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.

Example

Suppose you 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 shared object repository you can specify the location from which the property value should be taken. For example, in one test that uses this shared object repository you can specify that the property value comes from an environment variable. In another test it can come from the data table. In a third test you can specify it as a constant value.

You define all the repository parameters for a specific shared object repository using the Manage Repository Parameters dialog box. For details, see "Manage Repository Parameters Dialog Box" on page 278.

Considerations for Working with Repository Parameters

- 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.
- When you open a test that uses a shared 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 details on mapping repository parameters, see "Unmapped Shared Object Repository Parameter Values" on page 1366.



Managing Shared 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 retrieve a test object, you can manipulate it using the methods and properties available for that test object class. For example, you can use the `GetTOProperty` and `SetTOProperty` methods to retrieve and modify its properties. For details 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 shared 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 "QuickTest Automation Scripts" on page 1589.

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 shared 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.



Considerations for Working with Shared Object Repositories

Consider the following when working with shared object repositories:

Opening, Modifying, and Saving Shared Object Repositories

- **Enabling editing.** If you opened the shared object repository in read-only mode, you must enable editing for the shared object repository before you can modify it. This locks the shared object repository and prevents it from being modified simultaneously by multiple users. For details on enabling editing, see "How to Manage Shared Object Repositories" on page 258.
- **Unlocking.** When you enable editing for a shared object repository, the shared object repository is locked so that it cannot be modified by other users. To enable other users to modify the shared object repository, you must first unlock it (by disabling edit mode, or by closing it). If a shared 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.
- **Applying changes.** All changes you make to a shared object repository are automatically updated in all tests open on the same computer that use the shared object repository as soon as you make the change—even if you have not yet saved the shared object repository with your changes.

If you close the shared object repository without saving your changes, the changes are rolled back in any open tests that were open at the time.

- **Updating changes.** When you open a test on the same computer on which you modified the shared object repository, the test is automatically updated with all saved changes made in the associated shared object repository. To see saved changes in a test or repository open on a different computer, you must open the test or shared object repository file or lock it for editing on your computer to load the changes.

- **Merging.** You can 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 shared 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 details, see Chapter 9, "Object Repository Merge Tool."

Managing Objects in Shared 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.
- The following table describes QuickTest behavior in cases of duplicate objects in shared object repositories:

If...	QuickTest Uses...
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 local object definition.
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.	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.

General Tips and Guidelines

- If your test contains test objects from environments that are not installed/loaded with QuickTest, the test objects are displayed with a question mark in the shared object repository.
- When you modify a shared object repository, an asterisk (*) is displayed in the title bar until the object repository is saved.
- ➤ 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 shared object repository, you cannot undo and redo operations that were performed on that file before the save operation.
- You perform search operations in the shared object repository the same way you perform them in the local object repository. For details, see "Locating Objects" on page 165.

Tasks

How to Manage Shared Object Repositories

This task describes the different operations you can perform to manage shared object repositories using the Object Repository Manager.

This task includes the following steps:

- "Prerequisites and considerations" on page 258
- "Create a new shared object repository" on page 259
- "Open a shared object repository" on page 259
- "Enable editing for a shared object repository" on page 259
- "Save a shared object repository" on page 259
- "Close shared object repositories" on page 260
- "Associate a shared object repository with actions" on page 261
- "Merge shared object repositories" on page 261
- "Import a shared object repository from XML" on page 261
- "Export a shared object repository to XML" on page 262

Prerequisites and considerations

- If you want to edit a shared object repository stored in the file system, and the shared 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.
- If you are working with shared object repositories that are stored in a Quality Center project, you must connect to Quality Center either from QuickTest or from the Object Repository Manager by choosing **File > ALM/QC Connection** or clicking the **ALM/QC Connection** button. For details, see "HP ALM Connection Dialog Box" on page 1635.



Create a new shared object repository

Open the Object Repository Manager and create a new shared object repository, as described in "Object Repository Manager Main Window" on page 266.

Open a shared object repository

- Use the Open Shared Object Repository dialog box to open shared object repositories from the file system or a Quality Center project. For a user interface description, see "Open <Resource> Dialog Box" on page 406.

Make sure you clear the **Open in read-only mode** check box if you want to modify the shared object repository.

- You can also open a shared object repository from the **Recent Files** list in the **File** menu.

Enable editing for a shared object repository



Select **File > Enable Editing** or click the **Enable Editing** button. The shared object repository becomes editable. For details, see "Object Repository Manager Toolbar Buttons" on page 271.

For considerations on enabling editing, see "Considerations for Working with Shared Object Repositories" on page 255.

Save a shared object repository

Save the shared object repository, as described in "Object Repository Manager Toolbar Buttons" on page 271. 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. For details, see "Save <Resource> Dialog Box" on page 417.

For considerations on saving shared object repositories, see "Considerations for Working with Shared Object Repositories" on page 255.

Note about relative and absolute paths: When you specify a path to a resource in the file system or in Quality Center 9.2, 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 details, see "Relative Paths in QuickTest" on page 391.

If you are working with the Resources and Dependencies model with Quality Center 10.00 or HP ALM, you should specify an absolute Quality Center path. For details, see "Relative Paths and Quality Center" on page 1656.

Close shared object repositories

Do one of the following:

- To close a single shared object repository, select **File > Close** or click the **Close** button in the shared object repository window's title bar. The shared object repository closes and is automatically unlocked. If you made changes that are not yet saved, you are prompted to do so before the file closes.
- To close all open shared object repositories, select **File > Close All Windows**, or **Window > Close All Windows**. All open shared object repositories close and are automatically unlocked. If you made changes that are not yet saved, you are prompted to do so before the files close.

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.

Associate a shared object repository with actions

You can associate the shared object repository with one or more actions from within QuickTest. For details on associating shared object repositories, see "How to Modify Associations Between Resources and Your Test or Action" on page 1386.

Merge shared object repositories

Using the Object Repository Merge tool, you can merge 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 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 shared object repository.

You can also use the Object Repository Merge Tool to merge two shared object repositories into a single shared object repository.

For details, see Chapter 9, "Object Repository Merge Tool."

Import a shared object repository from XML

You can import an XML file (created using the required format) as a shared object repository. The XML file can either be a shared 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.

To import from XML:

- 1 Select **File > Import from XML**. The Open XML File dialog box opens. For a user interface description, see "Open <Resource> Dialog Box" on page 406.

The XML file is imported and a summary message box opens showing information regarding the number of test objects, checkpoint and output objects, parameters, and metadata that were successfully imported from the specified file.

- 2 Click **OK** to close the message box. The imported XML file is opened as a new shared object repository. You can now modify it as required and save it as a shared object repository.

Export a shared object repository to XML

You can export the objects in a shared 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 shared object repository whose objects you want to export is the active window.
- 2 Make sure that the shared object repository is saved.
- 3 Select **File > Export to XML**. The Save XML File dialog box opens. For a user interface description, see "Save <Resource> Dialog Box" on page 417.

QuickTest exports the objects in the shared object repository to the specified XML file, and a summary message box opens showing information regarding the number of test objects, checkpoint and output objects, parameters, and metadata that were successfully exported to the specified file.

- 4 Click **OK** to close the message box. You can now open the XML file and view or modify it with any XML editor.

How to Manage Objects in Shared Object Repositories

This task describes various operations you can perform to manage object in shared object repositories using the Object Repository Manager.

Note: 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 described in Chapter 4, "Managing Test Objects in Object Repositories", and are only referenced here. This task contains operations that you can only perform in the Object Repository Manager.

Although most of the procedures apply equally to the Object Repository Manager and the Object Repository window, the windows and options may differ slightly.

This task includes the following steps:

- "Prerequisites" on page 264
- "Manage objects in a shared object repository" on page 264
- "Add test objects using the Navigate and Learn option" on page 264
- "Manage repository parameters" on page 265

Prerequisites

Make sure that the shared object repository you want to edit is the active window.

Manage objects in a shared object repository

You can perform the following operations to manage objects in a shared object repository, as described in Chapter 4, "Managing Test Objects in Object Repositories":

- "How to Add a Test Object to an Object Repository" on page 176
- "If the Active Screen is not displayed, select View > Active Screen or click the Active Screen toolbar button to display it." on page 179
- "How to Locate an Object in an Object Repository" on page 183
- "How to Maintain Test Objects in Object Repositories" on page 184
- "How to Copy an Object to the Local Object Repository" on page 227

Add test objects using the Navigate and Learn option

- 1 Select **Object > Navigate and Learn** or press F6. The **Navigate and Learn** toolbar opens. For a user interface description, see "Navigate and Learn Toolbar" on page 274.
- 2 Click the parent object (for example, Browser, Dialog, Window) you want to add to the shared object repository to focus it. The **Learn** button in the toolbar is enabled.
- 3 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 shared object repository according to the defined filter.
- 4 When you finish adding the required objects to the shared 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.

Manage repository parameters

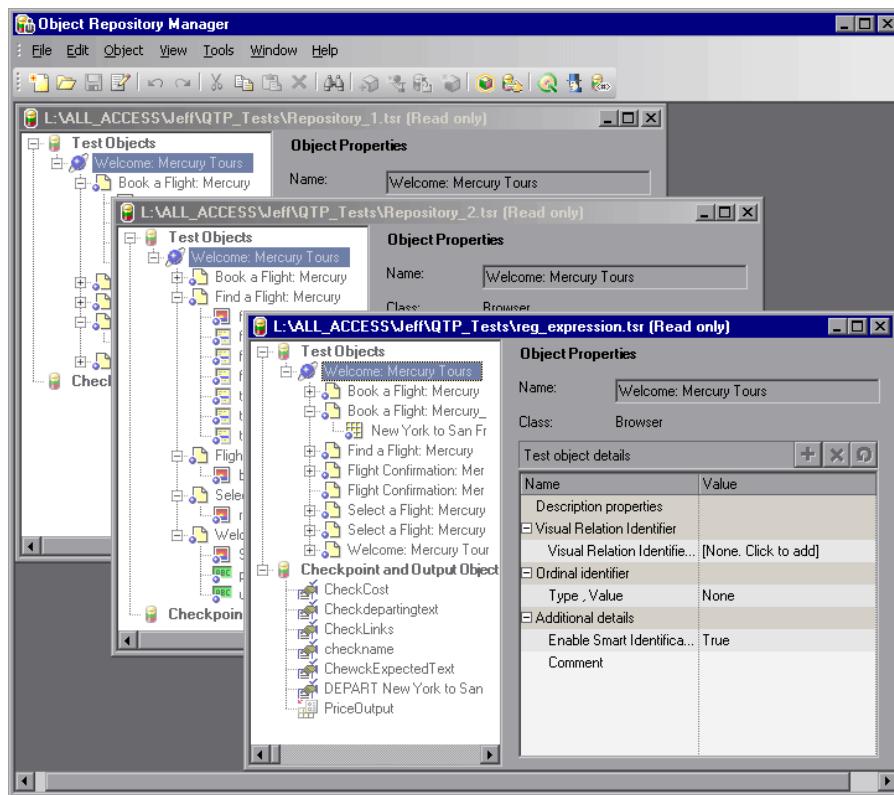
Use the Manage Repository Parameters dialog box, as described in "Manage Repository Parameters Dialog Box" on page 278.

Reference

Object Repository Manager Main Window

This window enables you to open multiple shared object repositories and modify them as needed.

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.



To access	Select Resources > Object Repository Manager.
Important information	See "Considerations for Working with the Object Repository Manager" on page 268

Relevant tasks	<ul style="list-style-type: none"> ➤ "How to Manage Shared Object Repositories" on page 258 ➤ "How to Manage Objects in Shared Object Repositories" on page 263
See also	<ul style="list-style-type: none"> ➤ "Shared Object Repositories Overview" on page 250 ➤ "Considerations for Working with Shared Object Repositories" on page 255 ➤ "New Property Dialog Box" on page 199 ➤ "Ordinal Identifier Dialog Box" on page 208

Considerations for Working with the Object Repository Manager

- 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.
- While the Object Repository Manager is open, you can continue working with other QuickTest windows.
- You cannot add checkpoint or output value objects to a shared object repository via the Object Repository Manager. They are added to the local object repository as the test object they are performed on. You can then upload them to the shared object repository, if needed. For details, see "How to Update a Shared Object Repository From a Local Object Repository" on page 347.
- 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).
- 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 details, see "Version Control in Quality Center 10.00 or HP ALM" on page 1695 and "Viewing and Comparing Versions of QuickTest Assets" on page 1669.

- Even when steps containing an object are deleted from your action, the objects remain in the shared object repository.

User interface elements are described in the sections below:

- "Object Repository Document Window" on page 269
- "Object Details Area (Test Objects)" on page 270
- "Object Details Area (Checkpoint Objects)" on page 270
- "Object Details Area (Output Value Objects)" on page 271
- "Object Repository Manager Toolbar Buttons" on page 271

Object Repository Document Window

Each open object repository contains the following user interface elements (unlabeled elements are shown in angle brackets):

UI Element	Description
Title bar	Displays the name and file path of the shared object repository currently active (in focus).
Object Repository tree	Located on the left side of the Object Repository window, and contains all objects in the shared object repository. Test objects of environments that are not installed with QuickTest are displayed with a question mark icon in the test object tree.
Name	Specifies the name that QuickTest assigns to the selected object. You can change the object name. For details, see "Renaming Test Objects" on page 170.
Class	Specifies the class of the selected object.

Object Details Area (Test Objects)

Enables you to view the properties and property values used to identify a test object during a run session.

For details on key functionality of this area, see:

- "Updating Identification Properties from an Object in Your Application" on page 168
- "Restoring Default Mandatory Properties for a Test Object" on page 169
- "How to Maintain Test Objects in Object Repositories" on page 184
- "Renaming Test Objects" on page 170
- "How to Define a Visual Relation Identifier for a Specific Test Object - Use-Case Scenario" on page 190

Object Details Area (Checkpoint Objects)

Enables you to view the properties of a checkpoint object, the same way as you do in the relevant checkpoint properties dialog box.

For details on specifying and modifying values for properties of a checkpoint object, see:

- "Checkpoint Properties Dialog Box" on page 609
- "Image Checkpoint Properties Dialog Box" on page 615
- "Bitmap Checkpoint Properties Dialog Box" on page 626
- "Table Checkpoint Properties Dialog Box (Table Content Tab)" on page 639
- "Text / Text Area Checkpoint Properties Dialog Box" on page 660
- "Database Checkpoint Properties Dialog Box" on page 679
- "XML Checkpoint Properties Dialog Box" on page 703
- The Web section of the *HP QuickTest Professional Add-ins Guide* (for Page and Accessibility checkpoints)

Object Details Area (Output Value Objects)

Enables you to view the properties of an output value object, the same way as you do in the relevant checkpoint properties dialog box.

For details on specifying and modifying values for properties of an output object, see:

- ▶ "Output Value Properties Dialog Box" on page 803
- ▶ "Text / Text Area Output Value Properties Dialog Box" on page 822
- ▶ "Table Output Value Properties Dialog Box (Table Content Tab)" on page 812
- ▶ "Database Output Value Properties Dialog Box" on page 830
- ▶ "XML Output Properties Dialog Box" on page 835

Object Repository Manager Toolbar Buttons

The Object Repository Manager toolbar contains the following buttons:

Button	Description
	New. Enables you to create a new shared object repository. For details, see "Create a new shared object repository" on page 259.
	Open. Enables you to open a shared object repository from the file system or from Quality Center. For details, see "Open a shared object repository" on page 259.
	Save. Enables you to save the active shared object repository to the file system or to Quality Center. For details, see "Save a shared object repository" on page 259.
	Enable Editing. Enables you to edit the active shared object repository, by making the shared object repository editable. For details, see "Enable editing for a shared object repository" on page 259.
	Undo. 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 details, see "If the Active Screen is not displayed, select View > Active Screen or click the Active Screen toolbar button to display it." on page 179.

Button	Description
	Redo. 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 details, see "If the Active Screen is not displayed, select View > Active Screen or click the Active Screen toolbar button to display it." on page 179.
	Cut. 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 details, see "If the Active Screen is not displayed, select View > Active Screen or click the Active Screen toolbar button to display it." on page 179.
	Copy. 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 details, see "If the Active Screen is not displayed, select View > Active Screen or click the Active Screen toolbar button to display it." on page 179.
	Paste. 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 details, see "If the Active Screen is not displayed, select View > Active Screen or click the Active Screen toolbar button to display it." on page 179.
	Delete. 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 details, see "If the Active Screen is not displayed, select View > Active Screen or click the Active Screen toolbar button to display it." on page 179.
	Find and Replace. 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 details, see "Find and Replace Dialog Box" on page 206.
	Add Object to Repository. 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 details, see "Adding and Deleting Test Objects in a Local or Shared Object Repository" on page 160.

Button	Description
	Update from Application. 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 details, see "Updating Identification Properties from an Object in Your Application" on page 168.
	Define new Test Object. 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 details, see "Define New Test Object Dialog Box" on page 200.
	Highlight in Application. 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 details, see "How to Locate an Object in an Object Repository" on page 183.
	Locate in Repository. 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 details, see "Locating a Test Object in the Object Repository" on page 166.
	Quality Center Connection. 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 details, see "HP ALM Connection Dialog Box" on page 1635.
	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 operations of the test object that QuickTest uses to represent that object. For details, see "How to Use the Object Spy to View Object Properties and Operations or Add an Object to a Repository" on page 144.
	Manage Repository Parameters. Enables you to add, edit, and delete repository parameters in the active shared object repository. For details, see "Manage Repository Parameters Dialog Box" on page 278.

 **Navigate and Learn Toolbar**

This toolbar enables you to add multiple test objects to a shared object repository while navigating through your application.



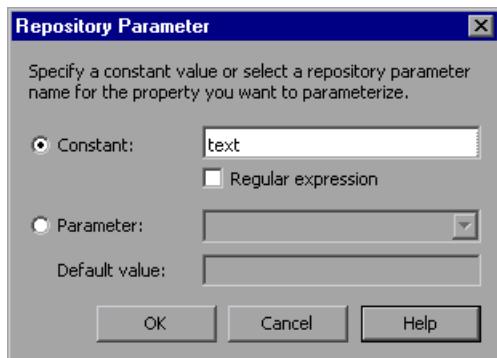
To access	In the Object Repository Manager, do one of the following: ► Select Object > Navigate and Learn . ► Press F6.
Important information	<ul style="list-style-type: none"> ► Each time you select a window to learn, the selected window and its descendant objects are added to the active shared object 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 details on modifying the filter definitions, see "Define Object Filter Dialog Box" on page 202. ► The Navigate and Learn option is not supported for environments with mixed hierarchies (object hierarchies that include objects from different environments), for example, <code>Browser("Homepage").Page("Welcome").AcxButton("Save")</code> or <code>Dialog("Edit").AcxEdit("MyEdit")</code>. To add objects within mixed hierarchies, use other options, as described in "Adding and Deleting Test Objects in a Local or Shared Object Repository" on page 160. ► Minimized windows are not learned when using the Navigate and Learn option.
Relevant tasks	"How to Manage Shared Object Repositories" on page 258
See also	"Object Repository Manager Main Window" on page 266

User interface elements are described below (unlabeled elements are shown in angle brackets):

UI Elements	Description
Learn	<p>Adds the active (in focus) parent object and its descendants to the shared object repository, according to the defined filter.</p> <p>Note: This button is disabled if there is no recognized active parent object (for example, Browser, Dialog, Window).</p> <p>Keyboard shortcut: ENTER</p>
	<p>Define Object Filter. Opens the Define Object Filter dialog box, enabling you to set filter definitions for objects learned. For details, see "Define Object Filter Dialog Box" on page 202. The current filter definitions are displayed in the button tooltip (in parentheses after the button name).</p> <p>Note: If this is the first time you are adding objects to the shared object repository, you may want to change the filter definitions before you continue.</p> <p>Keyboard shortcut: CTRL+F</p>
<keyboard shortcuts>	<ul style="list-style-type: none"> ► Help. F1 ► Return to Object Repository Manager. ESC

Repository Parameter Dialog Box

This dialog box enables you to specify or modify property values for test objects in the shared object repository. 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 a value using a repository parameter.



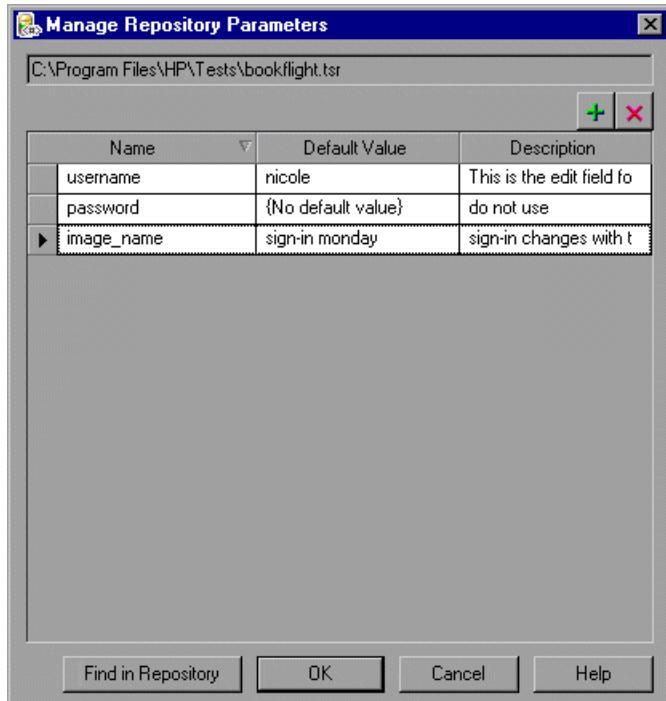
To access	In the Object Repository Manager, do the following: 1 Select the test object whose property value you want to specify. 2 In the Test object details area, click in the Value cell for the required property.
Important information	You can also specify or modify values for properties of a checkpoint or output object, directly in the object details area of the Object Repository Manager. For details, see "Object Repository Manager Main Window" on page 266.
Relevant tasks	"How to Manage Objects in Shared Object Repositories" on page 263
See also	<ul style="list-style-type: none"> ► "Working with Repository Parameters" on page 252 ► "Manage Repository Parameters Dialog Box" on page 278

User interface elements are described below:

UI Elements	Description
Constant	<p>The constant value of the object property.</p> <p>Note: You can also enter a constant value directly in the Value cell of the Test object details area.</p>
Regular expression	<p>Enables you to specify a regular expression for the constant value.</p> <p>For details on regular expressions, see "Regular Expressions Overview" on page 863 and "Regular Expression Characters and Usage Options" on page 875.</p>
Parameter	<p>Enables you to select a repository parameter from the list of defined parameters.</p>
Default value	<p>The default value for the parameter, where applicable.</p>

Manage Repository Parameters Dialog Box

This dialog box enables you to add, edit, and delete repository parameters for a single shared object repository.



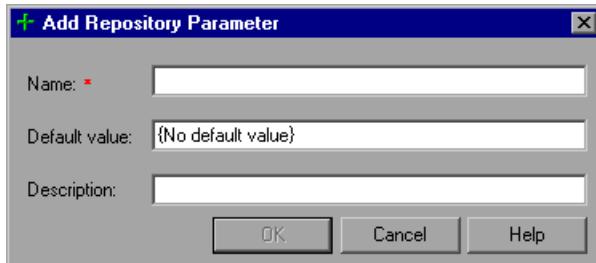
To access	In the Object Repository Manager, do one of the following: ► Select Tools > Manage Repository Parameters . ► Click the Manage Repository Parameters button  .
Relevant tasks	"How to Manage Objects in Shared Object Repositories" on page 263
See also	► "Working with Repository Parameters" on page 252. ► "Add Repository Parameter Dialog Box" on page 280 ► "Repository Parameter Dialog Box" on page 276

User interface elements are described below:

Option	Description
<Repository>	Displays the name and path of the shared object repository whose repository parameters you are managing.
	Add Repository Parameter. Enables you to add a new repository parameter. For details, see "Add Repository Parameter Dialog Box" on page 280.
	Delete Repository Parameter. Enables you to delete the currently selected repository parameters.
Parameter list (Name, Default Value, and Description)	Displays the list of repository parameters currently defined in this shared object repository. You can modify a parameter's default value and description directly in the parameter list. For considerations, see "Working with Repository Parameters" on page 252.
	<p>Clear Default Value. Enables you to remove the default value of the parameter. If you remove the default value, the text {No Default Value} is displayed in the cell.</p> <p>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.</p>
Find in Repository	Searches for and highlights the first test object in the shared 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 on.

 **Add Repository Parameter Dialog Box**

This dialog box enables you to define a new repository parameter. You can specify a default value for the parameter and a meaningful description to help identify it when it is used in a test step.



To access	In the Manage Repository Parameters dialog box (Tools > Manage Repository Parameters), click the Add Repository Parameter button  .
Relevant tasks	"How to Manage Objects in Shared Object Repositories" on page 263
See also	<ul style="list-style-type: none">▶ "Working with Repository Parameters" on page 252▶ "Manage Repository Parameters Dialog Box" on page 278▶ "Repository Parameter Dialog Box" on page 276

User interface elements are described below:

UI Elements	Description
Name	A meaningful name for the parameter. For a list of naming conventions, see "Naming Conventions" on page 1779.
Default value	<p>The 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 shared 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.</p> <p>Note: 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.</p>
Description	A description of the repository parameter. The description will help you identify the parameter when mapping repository parameters in a test.

Configuring Object Identification

This chapter includes:

Concepts

- [Object Identification Configuration - Overview on page 284](#)
- [Smart Identification on page 292](#)
- [Test Object Mapping for Unidentified or Custom Classes on page 299](#)

Tasks

- [How to Configure Object Identification for a Test Object Class on page 300](#)
- [How to Manage Identification Properties of a Test Object Class on page 301](#)
- [How to Map an Unidentified or Custom Class to a Standard Windows Class on page 303](#)

Reference

- [Object Identification Dialog Box on page 304](#)
- [Object Mapping Dialog Box on page 310](#)
- [Smart Identification Properties Dialog Box on page 312](#)

Concepts

Object Identification Configuration - Overview

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.

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 "Ordinal Identifiers" on page 287.
 - If a specific test object relies mainly on ordinal identifiers, you can also define visual relation identifiers for that test object, to help improve identification reliability for that object. For details, see "Visual Relation Identifiers" on page 174.
-

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.

This section also includes:

- "Mandatory and Assistive Properties" on page 286
- "Ordinal Identifiers" on page 287



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.

 **Ordinal Identifiers**

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.
- A visual relation identifier is not defined for the test object. For details, see "Visual Relation Identifiers" on page 174.

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 "The Index Ordinal Identifier" on page 289.
- **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 "The Location Ordinal Identifier" on page 289.

- **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 "The CreationTime Ordinal Identifier" on page 291.

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: When learning an object, 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 4, "Managing Test Objects in Object Repositories."

This section also includes:

- "The Index Ordinal Identifier" on page 289
- "The Location Ordinal Identifier" on page 289
- "The CreationTime Ordinal Identifier" on page 291

 **The Index Ordinal Identifier**

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")
```

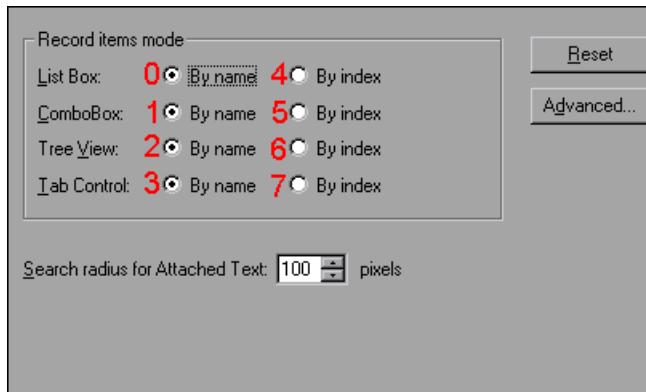
In contrast, the following statement refers to the second item in the list, as that is the first object of any type (`WebElement`) with the name `UserName`:

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

 **The Location Ordinal Identifier**

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")
```



The CreationTime Ordinal Identifier

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.

Smart Identification

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 (if defined and enabled) to try to identify the object.

The Smart Identification dialog box enables you to create and modify the Smart Identification definition that QuickTest uses for a selected test object class. 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.

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.

This section also includes:

- "When to Use Smart Identification" on page 294
- "The Smart Identification Process" on page 294
- "Smart Identification Information in the Run Results" on page 295
- "How QuickTest Uses Smart Identification - Use-Case Scenario" on page 296



When to Use Smart Identification

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.

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 pane of the Test Settings dialog box. For more information, see Chapter 4, "Managing Test Objects in Object Repositories," and "Run Pane (Test Settings Dialog Box)" on page 1471.

However, if you do not learn Smart Identification properties, you cannot enable the Smart Identification mechanism for an object later. For more information on learning Smart Identification properties, see "Smart Identification Properties Dialog Box" on page 312.



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.



Smart Identification Information in the Run 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 Run 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 "Smart Identification in the Run Results" on page 1170.

How QuickTest Uses Smart Identification - Use-Case Scenario

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:

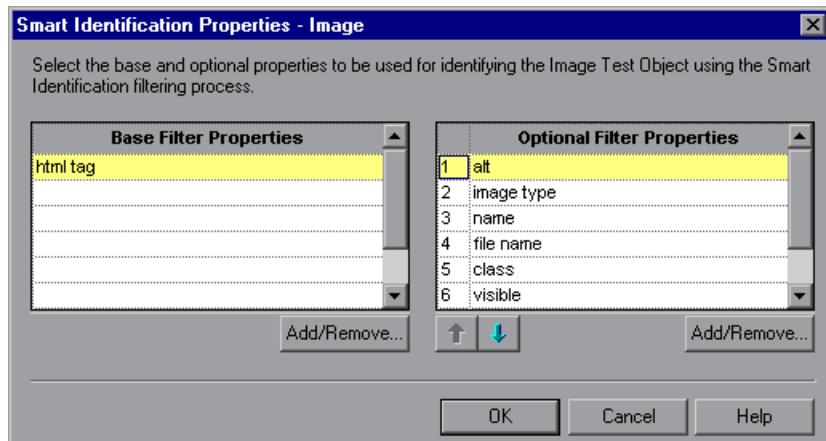
Name	Value
≡ Description properties	
image type	Image Button
html tag	INPUT
alt	Login

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 when 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.

Test Object Mapping for Unidentified or Custom 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.

Tasks

How to Configure Object Identification for a Test Object Class

This task describes steps you can perform to configure the properties that QuickTest uses to learn and identify objects. Perform one or more of these steps to configure object identification for a specific test object class.

Repeat the task for all relevant test object classes.

This task includes the following steps:

- "Prerequisites" on page 300
- "Set the properties that QuickTest learns for identifying an object in this test object class" on page 300
- "Set the properties that QuickTest uses for Smart Identification" on page 301

Prerequisites

Determine what you would like to change about how QuickTest identifies objects of various test object classes within your application. For more information, see "Object Identification Configuration - Overview" on page 284.

Set the properties that QuickTest learns for identifying an object in this test object class

In the Object Identification Dialog Box (described on page 304), set the properties that are learned for the test object description:

- 1** Select the environment.
- 2** Select the test object class.
- 3** Set mandatory and assistive properties.
- 4** Select an ordinal identifier.

Set the properties that QuickTest uses for Smart Identification

- 1 In the Object Identification Dialog Box, select the **Enable Smart ID** check box. The **Configure** button is enabled.
- 2 Click the **Configure** button to open the Smart Identification Properties Dialog Box (described on page 312).
- 3 Set the base and optional properties.
- 4 Set the order in which the optional properties are used.
- 5 If you do not want QuickTest to learn the Smart Identification properties, clear the **Enable Smart Identification** check box. The configuration is saved, but not used.

For more information, see "Smart Identification" on page 292.

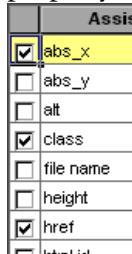
How to Manage Identification Properties of a Test Object Class

The following task describes how to manage the identification properties of a specified test object class in the Add/Remove Properties Dialog Box (described on page 308) and includes the following:

- "Add and remove properties to the object identification property lists" on page 302
- "Add identification properties to the test object class" on page 302

Add and remove properties to the object identification property lists

To include a property in the **Mandatory**, **Assistive**, **Base Filter**, or **Optional Filter Properties** lists, open the Add/Remove Properties Dialog Box, select the check box next to the property name you want to add. To remove a property from the list, clear the corresponding check box.



Add identification properties to the test object class

To add a new property to the test object class, open the **Add/Remove** Properties dialog box and click **New**. The **New Property** dialog box opens. Enter a valid property in the format `attribute/<PropertyName>` (for Web objects) or `<PropertyName>` (for other environment objects) and click **OK**. The new property is added to the available properties list.

How to Map an Unidentified or Custom Class to a Standard Windows Class

This task describes how to map an unidentified or custom class to a standard Windows class. This instructs QuickTest to identify objects of the specified class in the same way as it identifies objects of the Windows class to which it is mapped.

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

- 1 Open the Object Mapping dialog box by selecting **Standard Windows** in the **Environment** box in the Object Identification dialog box and then clicking the **User-Defined** button. For more information, see "Object Mapping Dialog Box" on page 310.
- 2  Click the pointing hand and then click the object whose class you want to add as a user-defined test object class. The name of the user-defined object is displayed in the **Class name** box.

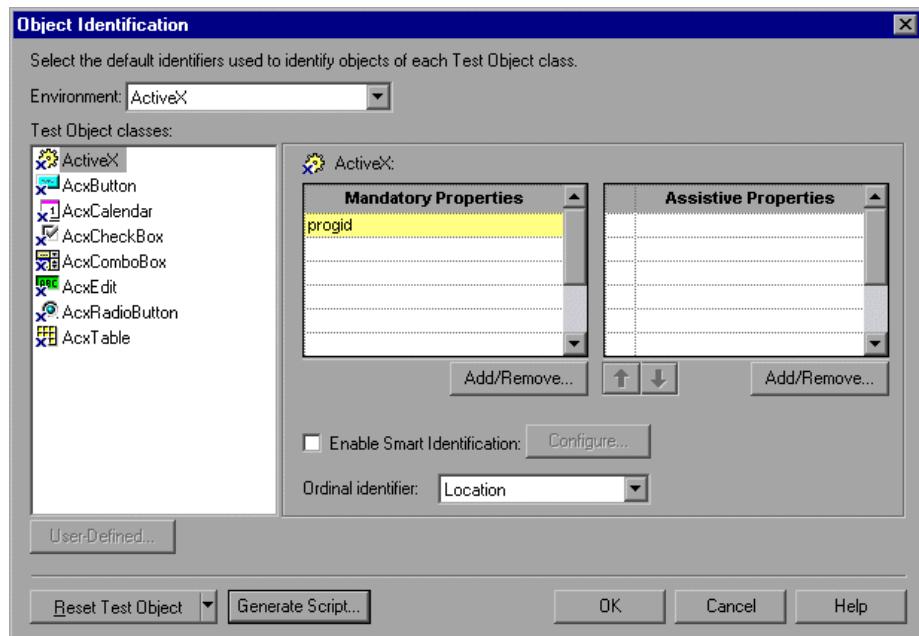
For information on using the pointing hand, see "Tips for Using the Pointing Hand" on page 157.
- 3 In the **Map to** box, select the standard object class to which you want to map your user-defined test object class and click **Add**. The class name and mapping is added to the object mapping list.
- 4 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 class. Note that your object has an icon with a red U in the lower-right corner, identifying it as a user-defined class.
- 5 Configure the object identification settings for your user defined test object class just as you would any other test object class. For more information, see "How to Configure Object Identification for a Test Object Class" on page 300.

Caution: 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.

Reference

Object Identification Dialog Box

This dialog box enables you 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 class.



To access	Select Tools > Object Identification .
Important information	<ul style="list-style-type: none"> ➤ Only currently loaded environments are listed in the Environments box. ➤ You cannot include the same property in both the mandatory and assistive property lists. ➤ 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 classes cannot be configured, for example, the WinMenu, VbLabel, and VbToolbar objects. These classes are therefore not included in the Test Object classes list for the selected environment.
Relevant tasks	"How to Configure Object Identification for a Test Object Class" on page 300
See also	"Object Identification Configuration - Overview" on page 284

User interface elements are described below:

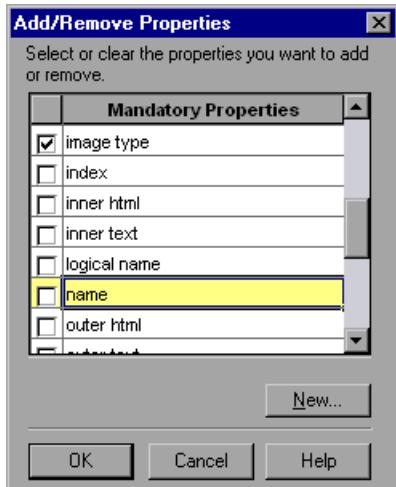
UI Elements	Description
Environment	<p>The list of environments, according to the loaded add-ins. This list may also include additional environments for which you or a third party developed support using QuickTest Add-in Extensibility.</p> <p>For more information on loading add-ins, see the section on loading QuickTest add-ins in the <i>HP QuickTest Professional Add-ins Guide</i>.</p>
Test Object classes	<p>The alphabetized list of the test object classes associated with the selected environment.</p> <p>Note: In Standard Windows, the user-defined classes are displayed at the bottom of the list.</p>
Mandatory Properties	The list of properties that QuickTest always learns as part of the description for test objects of the selected class.

UI Elements	Description
Assistive Properties	The list of additional properties that QuickTest can learn for a test object of the selected class to create a unique test object description.
	<p>Up and Down arrows. Enable you to set the preferred order for the Assistive Properties list.</p> <p>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.</p>
Add/Remove	Enables you to add or remove properties for the test object class using the Add/Remove Properties Dialog Box. For more information, see "Add/Remove Properties Dialog Box" on page 308.
Enable Smart Identification	Enables or disables Smart Identification for the selected test object class.
Configure	<p>Opens the Smart Identification Properties Dialog Box (described on page 312), which enables you to specify the properties for QuickTest to learn as Smart Identification Properties.</p> <p>Enabled only when the Enable Smart Identification check box is selected.</p>
Ordinal identifier	The type of ordinal identifier to use when QuickTest needs to identify an object with an otherwise identical description. The ordinal identifier enables QuickTest to assign a serial order value to each of the otherwise identical objects. For more information, see "Ordinal Identifiers" on page 287.
User-Defined	<p>Enables you to map an unidentified or custom class to a standard Windows class by opening the Object Mapping Dialog Box. For more information, see "Object Mapping Dialog Box" on page 310.</p> <p>Note: To enable this option, you must select the Standard Windows environment from the Environment list.</p>

UI Elements	Description
Reset Test Object	<p>The down arrow enables you to select one of the following options:</p> <ul style="list-style-type: none"> ▶ Reset Test Object. Resets the selected test object to the system default. (Default selection) ▶ 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. <p>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, using the Reset Test Object option in the Object Identification Dialog Box. For more information, see "Object Identification Dialog Box" on page 304.</p> <p>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 "Test Object Mapping for Unidentified or Custom Classes" on page 299.</p>
Generate Script	<p>Enables you to generate an automation script that sets the current object identification settings.</p> <p>When you click the Generate Script button, a Save As dialog box opens, enabling you to specify the name and file system location to store the generated file.</p> <p>You can use some or all of the script lines from this generated script in an automation script. This can be useful, for example, if you want to set the same object identification settings on multiple QuickTest computers.</p> <p>For more information, see "QuickTest Automation Scripts" on page 1589 and the <i>QuickTest Professional Automation Object Model Reference</i></p>

Add/Remove Properties Dialog Box

This dialog box enables you to set the properties you want to include in the **Mandatory**, **Assistive**, **Base Filter**, or **Optional Filter Properties** lists that are used to define the object identification preferences for a selected test object class. You can also add new property names to the set of usable properties.



To access	<p>You can access this dialog box by clicking Add/Remove in the following dialog boxes:</p> <ul style="list-style-type: none"> ► Object Identification Dialog Box (described on page 304) ► Smart Identification Properties Dialog Box (described on page 312)
Important information	<ul style="list-style-type: none"> ► You cannot add the same property to both the mandatory and assistive property lists. ► You cannot add the same property to both the base and optional filter property lists.

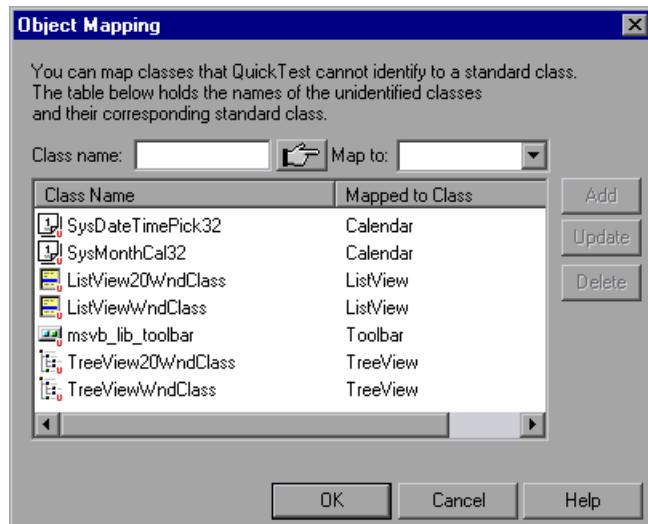
Relevant tasks	"How to Configure Object Identification for a Test Object Class" on page 300
See also	<ul style="list-style-type: none"> ➤ "Object Identification Configuration - Overview" on page 284 ➤ "Smart Identification" on page 292

User interface elements are described below (unlabeled elements are shown in angle brackets):

UI Elements	Description
<Properties> list	The list of available identification properties for the selected test object class. For more information, see "Add and remove properties to the object identification property lists" on page 302.
New	<p>Enables you to add new identification properties to the available properties list.</p> <p>QuickTest retrieves the property values from the object's native properties. Therefore, the name of a new identification property must be identical to the name of a native property. (In most environments, you can use the Object Spy to view the list of available native properties.)</p> <p>Note: New properties are displayed in the Object Spy but are not available for checkpoints on objects of the selected class.</p> <p>For more information, see "Add identification properties to the test object class" on page 302.</p>

Object Mapping Dialog Box

This dialog box enables you to map an object of an unidentified or custom class to a standard Windows test object class.



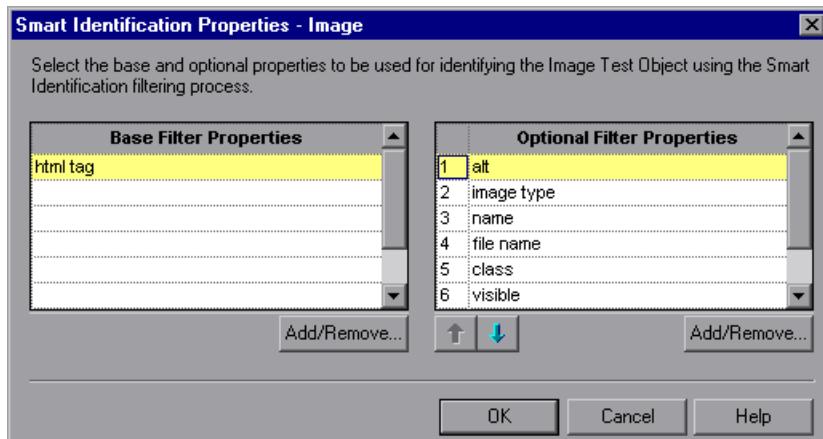
To access	In the Object Identification Dialog Box (described on page 304), select the Standard Windows environment, and click the User Defined button.
Relevant tasks	"How to Map an Unidentified or Custom Class to a Standard Windows Class" on page 303
See also	"Test Object Mapping for Unidentified or Custom Classes" on page 299

User interface elements are described below:

UI Elements	Description
Class name	The name of the user-defined object selected by using the Pointing Hand .
	Pointing Hand. Enables you to add the object whose class you want to as a user-defined class by minimizing the QuickTest window and navigating to the object. The name of the user-defined object is displayed in the Class name box. For information on using the pointing hand, see "Tips for Using the Pointing Hand" on page 157.
Map to	The list of available standard Windows test object classes to map to the user-defined object.
Add	Enables you to add the selected class name and mapping in the object mapping list with values set in the Class Name and Map to fields.
Update	Enables you to update the selected class name and mapping in the object mapping list with new values set in the Class Name and Map to fields.
Delete	Enables you to delete the selected class name and mapping from the object mapping list.

Smart Identification Properties Dialog Box

The Smart Identification dialog box enables you to configure the Smart Identification mechanism for the selected test object class.



To access	In the Object Identification Dialog Box (described on page 304), select an environment and a test object class, select Enable Smart Identification , and click Configure .
Important information	By default, some test object classes already have Smart Identification configurations and others do not. Those with default configurations also have the Enable Smart Identification check box selected by default in the Object Identification Dialog Box (described on page 304).
See also	"Smart Identification" on page 292

User interface elements are described below:

UI Elements	Description
Base Filter Properties	The list of properties that QuickTest learns as base filter properties for this test object class. The Smart Identification mechanism uses these properties to create a list of possible candidate objects.
Optional Filter Properties	The list of properties that QuickTest learns as optional filter properties for this test object class. The Smart Identification mechanism uses these properties in the specified order to narrow down the object candidate list to one object.
	Up and Down arrows. Enable you to set the preferred order for the Optional Filter Properties list. When QuickTest uses Smart Identification, it creates a list of possible candidate objects according the Base Filter Properties , and then checks the values of the Optional Filter Properties one by one according to the order you set, until it narrows down the candidate list to one object.
Add/Remove	Opens the Add/Remove Properties Dialog Box, enabling you to modify the list of properties learned for Smart Identification for this test object class. For more information, see "Add/Remove Properties Dialog Box" on page 308.

8

Object Repository Comparison Tool

This chapter includes:

Concepts

- Object Repository Comparison Tool Overview on page 316

Tasks

- How to Compare Two Object Repositories on page 318

Reference

- Object Repository Comparison Tool Main Window on page 320
- New Comparison Dialog Box on page 335

Concepts

Object Repository Comparison Tool Overview

The QuickTest Object Repository Comparison Tool enables you to compare two shared object repositories, and view the differences in their objects, such as different object names, different test object descriptions, and so on. The tool is accessible from the Object Repository Manager.

Differences between objects in the two object repository files are identified according to default rules. During the comparison process, the object repository files remain unchanged. For more information about the types of comparisons identified by the Object Repository Comparison Tool, see "Understanding Object Differences" on page 332.

After the comparison 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 specify. Objects that are included in one object repository only are indicated 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, "Object Repository Merge Tool."

This section also includes:

- "When to Use the Object Repository Comparison Tool" on page 317
- "Understanding the Repository Panes" on page 317

When to Use the Object Repository Comparison Tool

In addition to this tool, you can perform merge and comparison operations using the Asset Comparison Tool and the Object Repository Merge Tool.

Consider the following when selecting which tool to use:

- The Object Repository Comparison Tool is designed for comparing repositories that are different, but have a set of overlapping objects. This tool is useful if you want to decide whether to merge two repositories without performing the actual merge and addressing object conflicts in the Object Repository Merge Tool. For details, see "How to Compare Two Object Repositories" on page 318 and "Object Repository Merge Tool" on page 337.
- The Asset Comparison Tool can also compare two repositories, but it is designed for comparing different versions of the same repository to identify changes between versions. For details, see "Viewing and Comparing Versions of QuickTest Assets" on page 1669.

Understanding the Repository Panes

The object repository panes of the Object Repository Comparison Tool 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 comparison found.

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. You can synchronize the object repositories to display the same object in both views.

Tasks

How to Compare Two Object Repositories

This task describes how to compare two object repositories according to predefined settings that define how comparisons between objects are identified.

This task includes the following steps:

- "Prerequisites" on page 318
- "Select the Shared Object Repositories to compare" on page 319
- "Analyze the initial comparison results" on page 319
- "Analyze the detailed comparison results" on page 319
- "Utilize additional tools to help you perform the comparison - Optional" on page 319

1 Prerequisites

- Determine the shared object repositories you want to compare. Generally, shared object repositories should contain objects from the same application, and may have differences in objects or test object descriptions because the repositories were created at different times or under different circumstances.
- Make sure that the Object Repository Manager window is open. For information on working with the Object Repository Manager, see "Object Repository Manager Main Window" on page 266.
- Make sure the color settings are configured to match your needs. For details, see "Color Settings Dialog Box (Object Repository Comparison Tool)" on page 334.

2 Select the Shared Object Repositories to compare

- a In the Object Repository Manager window, select **Tools > Object Repository Comparison Tool** to open the Object Repository Comparison Tool. The New Comparison Dialog Box (described on page 335) opens.
- b Specify the two object repository files you want to compare.

3 Analyze the initial comparison results

After the comparison is complete, you can view the results summary in the Comparison Statistics Dialog Box (described on page 331).

4 Analyze the detailed comparison results

Review and analyze the comparisons between the repositories in the Object Repository Comparison Tool Main Window (described on page 331).

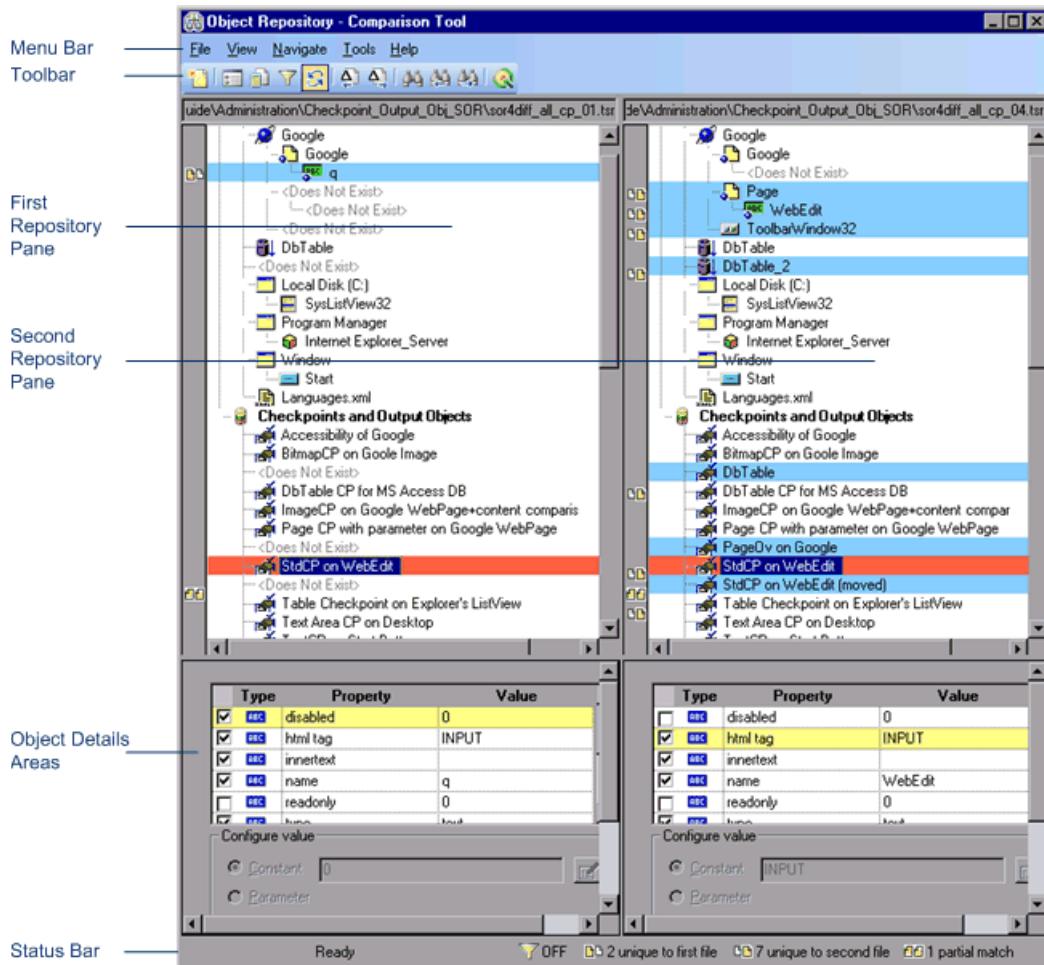
5 Utilize additional tools to help you perform the comparison - Optional

- Synchronize the object repositories to display the same object in both views by clicking the **Synchronized Nodes** button using the Menu Commands and Toolbar Buttons (described on page 321).
- Filter the objects and show only the objects that you want to view using the Filter Dialog Box (Object Repository Comparison Tool) (described on page 327).
- Locate one or more objects in a selected object repository whose name contains a specified string using the Find Dialog Box (Object Repository Comparison Tool) (described on page 329).
- Adjust the colors of text and background of object names, and empty nodes representing objects that exist in the other object repository only, using the Color Settings Dialog Box (Object Repository Comparison Tool) (described on page 334).

Reference

Object Repository Comparison Tool Main Window

This window displays the two repositories selected for comparison and provides tools for analyzing the comparison.



To access	In the Object Repository Manager , select Tools > Object Repository Comparison Tool .
Important information	<ul style="list-style-type: none"> ➤ You cannot work with the Object Repository Manager or the Object Repository Merge Tool while the Object Repository Comparison Tool is open. ➤ The Object Repository Comparison Tool gives precedence to matching test object descriptions over the matching of test object names.
Relevant tasks	"How to Compare Two Object Repositories" on page 318

The Object Repository Comparison Tool window contains the following key elements:

- "Menu Commands and Toolbar Buttons" on page 321
- "Repository Panes" on page 323
- "Test Object Details Areas" on page 325
- "Status Bar" on page 325

Menu Commands and Toolbar Buttons

File Menu

	Command	Shortcut Key	Function
	New Comparison	CTRL+N	Opens the New Comparison Dialog Box (described on page 335), enabling you to specify two object repositories on which to perform a new comparison operation.
	ALM/QC Connection		Enables you to connect the Object Repository Comparison Tool to a Quality Center project. For more information, see "HP ALM Connection Dialog Box" on page 1635.
	Exit		Closes the Object Repository Comparison Tool window.

View Menu

	Command	Function
	Statistics	Opens the Comparison Statistics Dialog Box (described on page 331), which summarizes the comparison between the two repositories, including the number and type of any differences found.
	Collapse All	Collapses the entire hierarchy in both comparison panes. Tip: While Synchronized Nodes is on, 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

	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 (Object Repository Comparison Tool) (described on page 329).
	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

	Command	Function
	Synchronized Nodes	Enables you to navigate the two object repository panes simultaneously or independently of one another.
	Filter	<p>Opens the Filter Dialog Box (Object Repository Comparison Tool) (described on page 327), enabling you to specify the comparison types that you want to show.</p> <p>Possible values:</p> <ul style="list-style-type: none"> ➤ 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.
	Color Settings	Opens the Color Settings Dialog Box (Object Repository Comparison Tool) (described on page 334), enabling you to specify the text and background color of the object names and empty nodes displayed in the comparison panes.

Help Menu

Command	Shortcut Key	Function
Object Repository Comparison Tool Help	F1	Opens the Object Repository Comparison Tool Help.

Repository Panes

The repository panes display a hierarchical view of the objects in the object repositories being compared. The differences and similarities between objects in the two panes are indicated by colored text and background colors for each object.

Differences can also be identified by the icons displayed to the left of the objects in the object repository panes, as follows:

UI Elements	Description
	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 comparison types, see "Understanding Object Differences" on page 332.

The object repository panes provide the following functionality:

- While in **Synchronized Nodes** mode, 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 "Menu Commands and Toolbar Buttons" on page 321.
- You can drag the edges of the panes to resize them in the Object Repository Comparison Tool window.

Test Object Details Areas

Shows the properties and values of the object selected in an object repository pane. For more information, see "Understanding the Repository Panes" on page 317.

Status Bar

Shows the status of the comparison process and details of the comparisons found during the object repository comparison.

User interface elements of the status bar are described below (unlabeled elements are shown in angle brackets):

UI Elements	Description
<progress bar>	Displays the comparison process status on the left of the status bar. Ready is displayed when the process is complete.
	ALM/QC Connection. Displayed when QuickTest is connected to a Quality Center project.
<filter>	Opens the Filter Dialog Box (Object Repository Comparison Tool) (described on page 327). The icon image indicates the filter status of the repository panes. Possible values: ➤ 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.

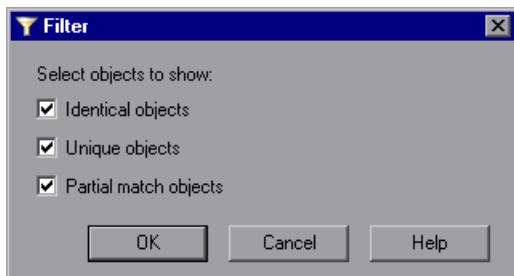
UI Elements	Description
	The number of objects that are unique to the first file.
	The number of objects that are unique to the second file.
	The number of objects in the first and second file that are not identical, but partially match.

This section also includes:

- "Filter Dialog Box (Object Repository Comparison Tool)" on page 327
- "Find Dialog Box (Object Repository Comparison Tool)" on page 329
- "Comparison Statistics Dialog Box" on page 331
- "Color Settings Dialog Box (Object Repository Comparison Tool)" on page 334
- "New Comparison Dialog Box" on page 335

Filter Dialog Box (Object Repository Comparison Tool)

This dialog box enables you to filter objects in the repository panes of the Object Repository Comparison Tool Main Window. For more information, see "Object Repository Comparison Tool Main Window" on page 320.



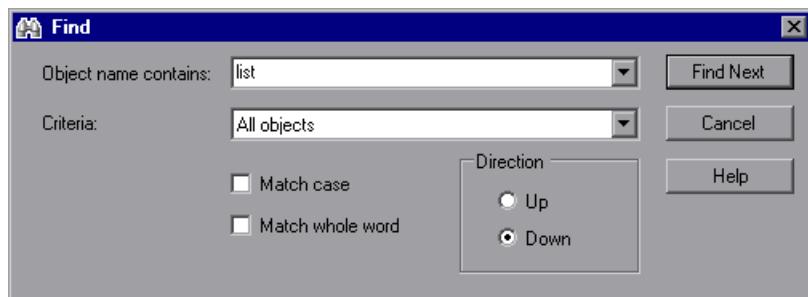
To access	<p>In the Object Repository Comparison Tool, do one of the following:</p> <ul style="list-style-type: none"> ▶ Select Tools > Filter. ▶ Click the Filter  button in the toolbar.
Important information	<ul style="list-style-type: none"> ▶ To view all the objects in both object repositories, select all of the check boxes. ▶ After closing the dialog box, the Filter icon in the status bar indicates the filter status of the repository panes. <p>Possible values:</p> <ul style="list-style-type: none"> ▶ 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.
Relevant tasks	"How to Compare Two Object Repositories" on page 318
See also	"Understanding the Repository Panes" on page 317

User interface elements are described below:

UI Elements	Description
Identical objects	Instructs the Object Repository Comparison tool to display objects that appear in both object repository files and have no differences in their name or in their properties.
Unique objects	Instructs the Object Repository Comparison tool to display objects that appear only in the first or only in the second object repository file.
Partial match objects	Instructs the Object Repository Comparison tool to display objects that are similar but have name or description differences.

Find Dialog Box (Object Repository Comparison Tool)

This dialog box enables you to find objects in the selected object repository pane according to predefined search criteria.



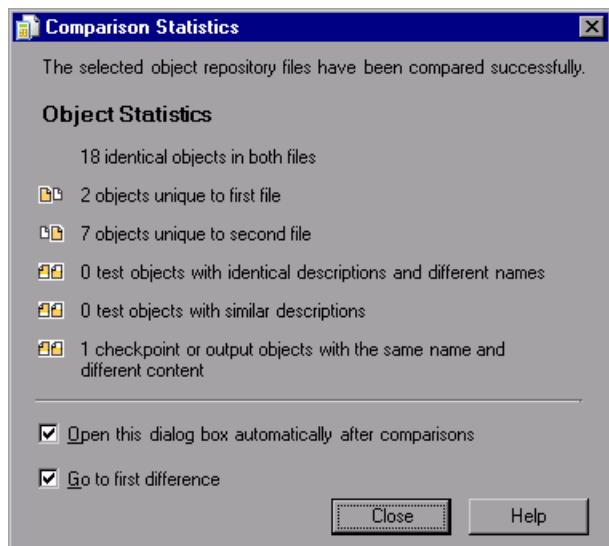
To access	In the Object Repository Comparison Tool , click the object repository pane that contains the required object and do one of the following: <ul style="list-style-type: none"> ▶ Select Navigate > Find. ▶ Click the  button in the toolbar.
Important information	After you set the Find options, you can close the Find dialog box and use menu commands, toolbar buttons, or shortcut keys to navigate to the next or previous node that matches your search criteria. For more information, see "Menu Commands and Toolbar Buttons" on page 321.
Relevant tasks	"How to Compare Two Object Repositories" on page 318
See also	"Understanding the Repository Panes" on page 317

User interface elements are described below:

UI Elements	Description
Object name contains	The full or partial name of the object you want to find.
Criteria	<p>The criteria to use to refine your search.</p> <p>The following criteria are available:</p> <ul style="list-style-type: none"> ➤ All objects ➤ Unique objects ➤ Partial match objects ➤ Unique or partial match objects
Match case	Instructs the Object Repository Comparison tool to distinguish between upper-case and lower-case characters in the search. When Match case is selected, QuickTest finds only those occurrences with the exact capitalization match to the text you entered in the Object name contains box.
Match whole word	Instructs the Object Repository Comparison tool to search only for whole word occurrences that match the text you entered in the Find Objects dialog box, and not as part of larger words.
Direction	<p>The direction from the current cursor location in which you want to search. The following options are available:</p> <ul style="list-style-type: none"> ➤ Up ➤ Down <p>The Find operation continues to search the entire file after it reaches the beginning or end of the object repository.</p>
Find Next	Highlights the next object that matches the specified criteria in the object repository.

Comparison Statistics Dialog Box

This dialog box lists the quantity for each type of comparison identified by the Object Repository Comparison Tool.



To access	In the Object Repository Comparison Tool , do one of the following: ► Select View > Statistics . ► Click the Statistics button  in the toolbar.
Important information	The icons displayed for each comparison type in the object statistics are the same as those used in the object repository panes. For more information, see "Understanding Object Differences" on page 332.
Relevant tasks	"How to Compare Two Object Repositories" on page 318
See also	"Understanding the Repository Panes" on page 317

User interface elements are described below:

UI Elements	Description
Object Statistics	The number and type of objects meeting each comparison criteria. Comparison types are described in "Understanding Object Differences" on page 332.
Open this dialog box automatically after comparisons	Indicates whether the Statistics dialog box is automatically displayed every time the Object Repository Comparison Tool completes a comparison.
Go to first difference	When selected, the Comparison tool highlights the first difference in the object repositories immediately after you close the Statistics dialog box.

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 are identical in both object repository files.
- **Unique to first file, or Unique to second file.** Objects that appear in only one of the object repository files.
- **Test objects with identical descriptions and different names.** Test objects that appear in both object repository files that have different names, but the same description properties and values.
- **Test objects with similar descriptions.** Test objects that appear in both object repository files that have similar, but not identical, description properties and values. One of the test objects always has a subset of the properties set of the other test object. This implies that it is likely to be a less detailed description of the same test object. For example, a test object named Button in the second object repository has the same description properties and values as a test object named Button in the first object repository, but also has additional properties and values.

Test objects that do not have any description properties, such as Page or Browser objects, are compared by name only. If the same test object is contained in both object repositories but with different names, they will be shown in the object repositories as two separate objects.

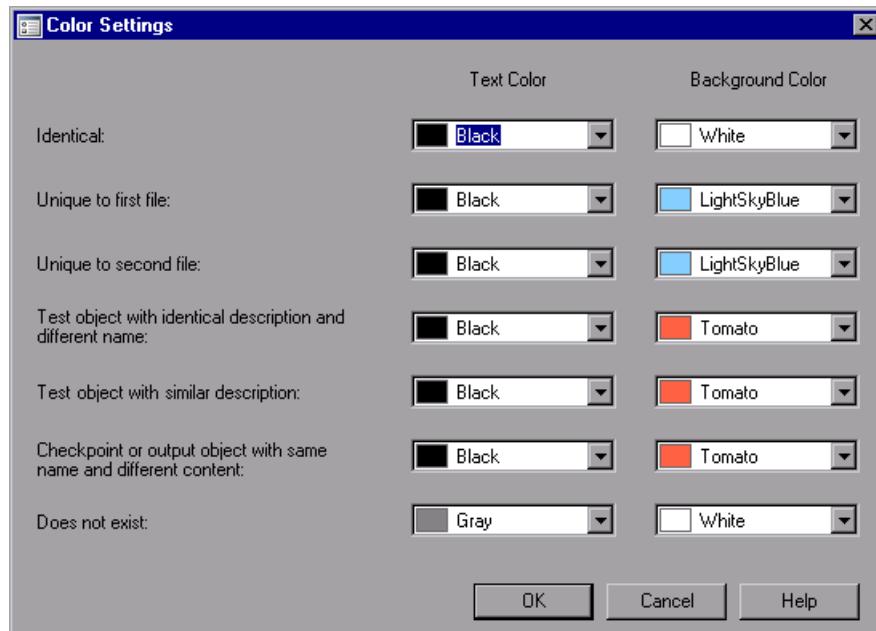
Note: Test objects with different visual relation identifier definitions are treated as objects with different descriptions.

- **Checkpoint or output objects with the same name and different content.** This option is relevant for checkpoint and output objects that are not completely identical for all settings.
- **Does not exist.** Objects that do not exist in one of the repository files, but do exist in the other file.

Object differences can also be viewed in the Object Repository Comparison Tool Main Window (described on page 320), according to settings defined in the Color Settings Dialog Box (Object Repository Comparison Tool) (described on page 334).

Color Settings Dialog Box (Object Repository Comparison Tool)

This dialog box enables you to define text and background color settings to help differentiate between the compared objects.



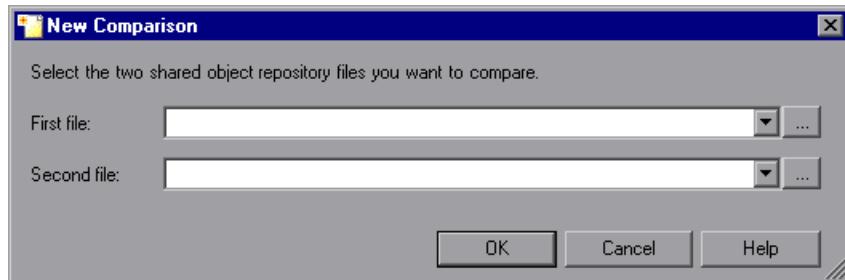
To access	In the Object Repository Comparison Tool , do one of the following: ► Select Tools > Color Settings . ► Click the Color Settings  button in the toolbar.
Relevant tasks	"How to Compare Two Object Repositories" on page 318
See also	"Understanding the Repository Panes" on page 317

User interface elements are described below (unlabeled elements are shown in angle brackets):

UI Elements	Description
<comparison types>	The list of comparison options on the left side of the dialog box. Difference types are described in "Understanding Object Differences" on page 332.
Test Color	Displays the current color used for the object names matching the corresponding comparison type.
Background Color	Displays the current color used for the repository pane rows matching the corresponding comparison type.

New Comparison Dialog Box

This dialog box enables you to select two object repositories to compare.



To access	<p>In the Object Repository Comparison Tool, do one of the following:</p> <ul style="list-style-type: none"> ► Select File > New Comparison. ► Click the New Comparison  button in the toolbar. <p>Note: This dialog box also opens automatically when you initially open the Object Repository Comparison Tool Main Window (described on page 320).</p>
-----------	---

Important information	<ul style="list-style-type: none"> ➤ 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. ➤ If you want to compare an object repository that was last saved using a version of QuickTest earlier than version 9.0, you must first open and save the repository in the Object Repository Manager to update it to the new format. ➤ 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 "How to Compare Two Object Repositories" on page 318, and then perform the comparison.
Relevant tasks	"How to Compare Two Object Repositories" on page 318

User interface elements are described below:

UI Elements	Description
First file	The file path for the first object repository.
Second file	The file path for the second object repository.
	Warning icon. 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. Tip: Position your pointer over the icon to see a tooltip explanation of the error. Supply an existing .tsr file with the correct path.

9

Object Repository Merge Tool

This chapter includes:

Concepts

- ▶ Object Repository Merge Tool Overview on page 338
- ▶ Object Conflicts on page 340

Tasks

- ▶ How to Merge Two Shared Object Repositories on page 344
- ▶ How to Update a Shared Object Repository From a Local Object Repository on page 347

Reference

- ▶ New Merge Dialog Box on page 350
- ▶ Object Repository Merge Tool Main Window on page 352
- ▶ Object Repository Merge Tool - Multiple Merge Window on page 374
- ▶ Settings Dialog Box (Object Repository Merge Tool) on page 368
- ▶ Update from Local Repository Dialog Box on page 378

Concepts

Object Repository Merge Tool Overview

QuickTest Professional enables you to merge two shared object repositories into a single shared object repository using the Object Repository Merge Tool.

This 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.

When to Merge 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 manually 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 configurable 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.

For more information, see "How to Merge Two Shared Object Repositories" on page 344.

Note: If you want to compare two shared object repositories without merging, you can use the Object Repository Comparison Tool. For details, see "Object Repository Comparison Tool Overview" on page 316.

When to Update a Shared Object Repository from Local Object Repositories

You can update a shared object repository by merging local object repositories associated with specific actions from 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 actions are moved to the target shared object repository (and removed from the local object repository). The action's steps then use 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 "Settings Dialog Box (Object Repository Merge Tool)" on page 368.

If you choose to merge local object repositories from 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.

For more information, see "How to Update a Shared Object Repository From a Local Object Repository" on page 347.

Object Conflicts

Merging two object repositories can result in conflicts arising from similarities between the objects they contain.

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 "Settings Dialog Box (Object Repository Merge Tool)" on page 368.

Conflicts between checkpoint or output value objects with the same name but different content are always resolved by merging both objects into the new repository and renaming one of them.

The Object Repository Merge Tool also allows you to change the way the merge was performed for each individual object that causes a conflict.

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 a specific conflict differently, for example, by keeping both objects.

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.

The Object Repository Merge Tool identifies three possible conflict types:

- "Different Objects with the Same Name Conflict" on page 341
- "Identical Description Different Name Conflict (Test Objects Only)" on page 342
- "Similar Description Conflict (Test Objects Only)" on page 343



Different Objects with the Same Name Conflict

An object in the primary object repository and an object in the secondary object repository have the same name, but completely different content.

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).

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 "Settings Dialog Box (Object Repository Merge Tool)" on page 368.

Note: Test objects with different visual relation identifier definitions are treated as objects with different descriptions.



Identical Description Different Name Conflict (Test Objects Only)

A test object in the primary object repository and a test 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 test object name from the object in the primary object repository.
- Taking the test object name from the object in the secondary object repository.
- Ignoring the test object from the local object repository and keeping the test object from the shared object repository (when updating a shared object repository from a local object repository).

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 "Settings Dialog Box (Object Repository Merge Tool)" on page 368.



Similar Description Conflict (Test Objects Only)

A test object in the primary object repository and a test object in the secondary object repository have the same name, and they have similar, but not identical, description properties and values. One of the test objects always has a subset of the properties set of the other test object. For example, a test object named Button in the secondary object repository has the same description properties and values as a test object named Button in the primary object repository, but also has additional properties and values.

You can resolve this conflict type by:

- Keeping the test object added from the primary object repository only.
- Keeping the test object added from the secondary object repository only.
- Keeping the test objects from both object repositories. In this case, the Object Repository Merge Tool automatically renames the test object that is added from the secondary file by adding an incremental numeric suffix to the name, for example, Button_1.
- Ignoring the test object from the local object repository and keeping the test object from the shared object repository (when updating a shared object repository from a local object repository).

By default, the conflict resolution settings for conflicts of this type are configured so that the target object repository takes the test object that has fewer identifying properties than the test object with which it conflicts. For information on changing the default settings, see "Settings Dialog Box (Object Repository Merge Tool)" on page 368.

Tasks

How to Merge Two Shared Object Repositories

This task describes how to merge two shared object repositories according to predefined settings that define how conflicts between objects are resolved.

This task includes the following steps:

- "Prerequisites" on page 344
- "Select the shared object repositories to merge" on page 345
- "Analyze the initial merge results" on page 345
- "Analyze the detailed merge results" on page 345
- "Utilize additional tools to help you perform the comparison (Optional)" on page 345
- "Adjust object conflict resolutions" on page 346
- "Save the target object repository" on page 346
- "Associate the target object repository with actions" on page 346

1 Prerequisites

- Determine the shared object repositories you want to compare. You generally merge two shared object repositories that contain objects from the same application, and that may have differences in objects or test object descriptions because the repositories were created at different times or under different circumstances.
- Make sure that the Object Repository Manager window is open. For information on working with the Object Repository Manager, see "Object Repository Manager Main Window" on page 266.
- Make sure the resolution and color settings are configured to match your needs. For details, see "Resolution Tab (Settings Dialog Box - Object Repository Merge Tool)" on page 369 and "Colors Tab (Settings Dialog Box - Object Repository Merge Tool)" on page 372.

2 Select the shared object repositories to merge

- a In the Object Repository Manager window, select **Tools > Object Repository Merge Tool** to open the Object Repository Merge Tool. The New Merge Dialog Box (described on page 350) opens.
- b Specify the two object repository files you want to merge.

3 Analyze the initial merge results

After the merge is complete, you can view the results summary in the Merge Statistics Dialog Box (described on page 361).

4 Analyze the detailed merge results

Review and analyze the merge between the repositories in the Object Repository Merge Tool Main Window (described on page 352).

5 Utilize additional tools to help you perform the comparison (Optional)

- Change the view presented by the Object Repository Merge Tool according to your working preferences, by dragging the edges of the panes to resize them, or selecting the appropriate option from the **View** menu, as described in "Object Repository Merge Tool Main Window" on page 352.
- Filter the objects and show only the objects that you want to view by using the Filter Dialog Box (Object Repository Merge Tool) (described on page 358).
- Locate one or more objects in a selected object repository whose name contains a specified string using the Find Dialog Box (Object Repository Merge Tool) (described on page 359).

6 Adjust object conflict resolutions

If one or more of the merge resolutions does not match your needs, follow the steps below to adjust them:

- a** 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 "Object Conflicts" on page 340.
- b** 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.
- c** 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.

7 Save the target object repository

When the object conflicts are resolved satisfactorily, save the new merged shared object repository. 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.

If you are connected to Quality Center, you can save your merged shared object repository in the Test Resources module of your project.

8 Associate the target object repository with actions

You can now associate the new merged object repository with actions, especially ones that were previously associated with the original object repositories, so that the objects in the object repository can be accessed by the tests.

How to Update a Shared Object Repository From a Local Object Repository

This task describes how to add local object repositories to a shared object repository.

This task includes the following steps:

- "Prerequisites" on page 347
- "Select the shared object repository into which you want to merge the local repositories" on page 348
- "Analyze the initial merge results" on page 348
- "Analyze the detailed merge results" on page 348
- "Utilize additional tools to help you perform the comparison (Optional)" on page 348
- "Adjust object conflict resolutions" on page 349
- "Save the target object repository" on page 349

1 Prerequisites

- Make sure that the shared object repository you want to update from the local object repositories is already associated with the relevant actions.
- Make sure the test containing the local object repositories is closed.
- Make sure that the Object Repository Manager window is open. For information on working with the Object Repository Manager, see "Object Repository Manager Main Window" on page 266.
- Make sure the resolution and color settings are configured to match your needs. For details, see "Resolution Tab (Settings Dialog Box - Object Repository Merge Tool)" on page 369 and "Colors Tab (Settings Dialog Box - Object Repository Merge Tool)" on page 372.

2 Select the shared object repository into which you want to merge the local repositories

- a** In the Object Repository Manager, open the shared object repository into which you want to merge the local repositories. If the object repository opened in read-only mode, select **Enable Editing**.
- b** Select **Tools > Update from Local Repository** to open the Update from Local Repository Dialog Box (described on page 378), and then select the local object repositories to add.

3 Analyze the initial merge results

View the initial merge results in the "Merge Statistics Dialog Box" on page 361.

4 Analyze the detailed merge results

Review and analyze the detailed merge results in the "Object Repository Merge Tool - Multiple Merge Window" on page 374.

5 Utilize additional tools to help you perform the comparison (Optional)

- Change the view presented by the Object Repository Merge Tool according to your working preferences, by dragging the edges of the panes to resize them, or selecting the appropriate option from the **View** menu, as described in "Object Repository Merge Tool - Multiple Merge Window" on page 374.
- Filter the objects and show only the objects that you want to view by using the Filter Dialog Box (Object Repository Merge Tool) (described on page 358).
- Locate one or more objects in a selected object repository whose name contains a specified string using the Find Dialog Box (Object Repository Merge Tool) (described on page 359).

6 Adjust object conflict resolutions

If one or more of the merge resolutions does not match your needs, follow the steps below to adjust them:

- a 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 object is highlighted in the local object repository.

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 "Object Conflicts" on page 340.

- b 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.
- c 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.

7 Save the target object repository

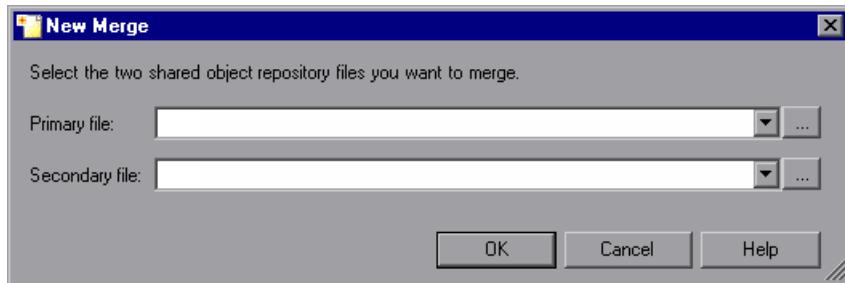
When the object conflicts are resolved satisfactorily, save the new merged shared object repository. QuickTest saves the object repository with a **.tsr** extension in the specified location and displays the file name and path above the target object repository in the Object Repository Merge Tool window.

If you are connected to Quality Center, you can save your merged shared object repository in the Test Resources module of your project.

Reference

New Merge Dialog Box

This dialog box enables you to select two object repositories to merge.



To access	In the Object Repository Merge Tool window, do one of the following: ► Select File > New Merge . ► Click the New Merge  button in the toolbar. Note: This dialog box also opens automatically when you initially open the Object Repository Merge Tool Main Window (described on page 352).
------------------	---

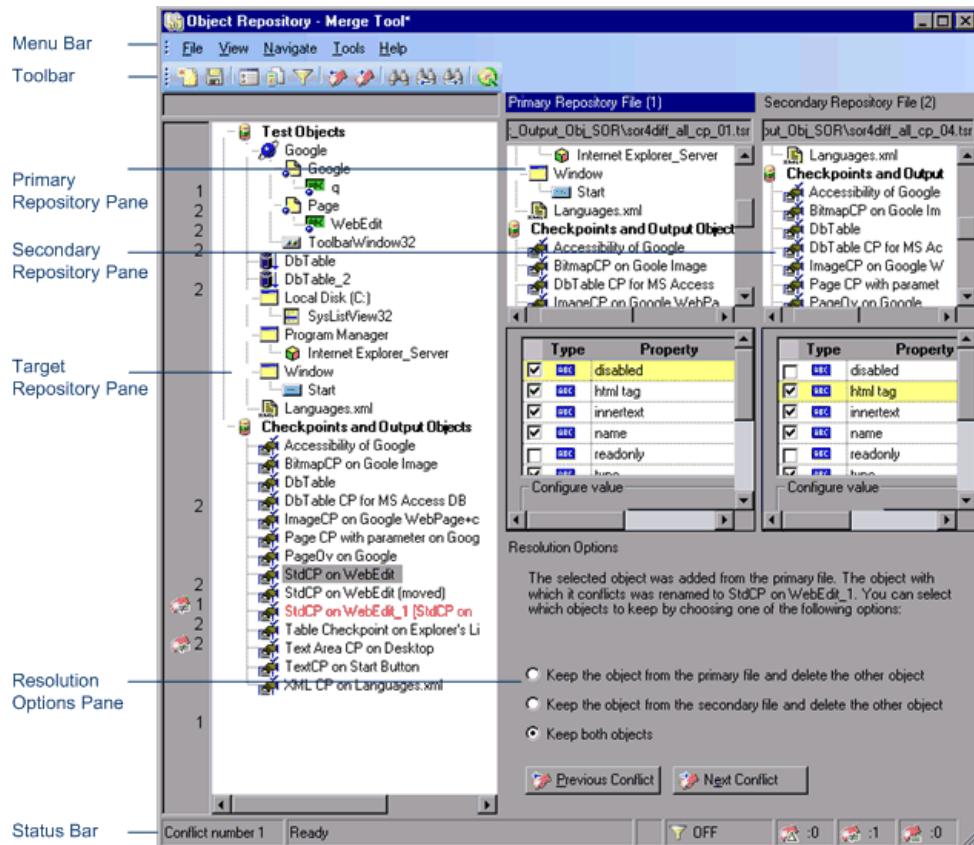
Important information	<ul style="list-style-type: none"> ➤ An object repository that is currently open by another user is locked. If you try to merge a locked file, a warning message displays, but you can still perform the merge because the merge process does not modify the source files. Note that changes made to the locked file by the other user may not be included in the merged object repository. ➤ If this dialog box opens automatically, but 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 "Settings Dialog Box (Object Repository Merge Tool)" on page 368, and then open this dialog box again to perform the merge. ➤ To improve efficiency of the merge process, 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. ➤ If you want to merge an object repository that was last saved using a version of QuickTest earlier than version 9.0, you must first open and save the repository in the Object Repository Manager to update it to the new format.
Relevant tasks	"How to Merge Two Shared Object Repositories" on page 344

User interface elements are described below:

UI Elements	Description
Primary file	The file system or Quality Center path for the first object repository.
Secondary file	The file system or Quality Center path for the second object repository.
	Warning icon. 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. Tip: Position your pointer over the icon to see a tooltip explanation of the error. Supply an existing .tsr file with the correct path.

Object Repository Merge Tool Main Window

This window displays the two repositories selected for merging and the target repository containing the merged content. This window also provides tools for analyzing the merge and resolving conflicts.



To access	In the Object Repository Manager , select Tools > Object Repository Merge Tool .
Important information	<ul style="list-style-type: none"> ➤ You cannot work with the Object Repository Manager or the Object Repository Comparison Tool while the Object Repository Merge Tool is open. ➤ Test objects that do not have description properties, 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.
Relevant tasks	"How to Merge Two Shared Object Repositories" on page 344

The Object Repository Merge Tool main window contains the following key elements:

- "Menu Bar and Toolbar" on page 353
- "Target Repository Pane" on page 353
- "Primary and Secondary Repository Panes" on page 355
- "Resolution Options Pane" on page 355
- "Status Bar" on page 356

Menu Bar and Toolbar

Displays menus with Object Repository Merge Tool commands. These commands are described in "Object Repository Merge Tool Menu Commands and Toolbar Buttons" on page 362.

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 "Colors Tab (Settings Dialog Box - Object Repository Merge Tool)" on page 372.

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.

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 "Find Dialog Box (Object Repository Merge Tool)" on page 359.
- 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 UI elements (unlabeled elements are shown in angle brackets):

UI Elements	Description
Conflict number	The conflict number (if any) of the object selected in the target object repository pane.
<progress bar>	Displayed during the merge process. Ready is displayed when the merge is complete.
	Filter. Displays the filter status of the target object repository pane. Possible states: <ul style="list-style-type: none"> ➤ ON. Indicated that a filter is currently in use. ➤ OFF. indicates that the object repositories are not filtered and all objects are shown. Note: Click the Filter icon to open the "Filter Dialog Box (Object Repository Merge Tool)" on page 358.

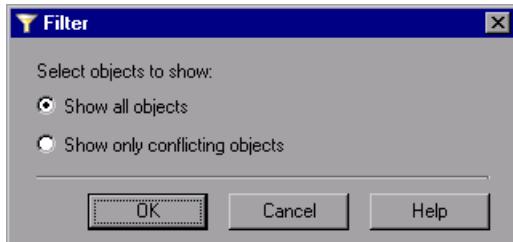
UI Elements	Description
	ALM/QC Connection. Displayed when QuickTest is connected to a Quality Center project
	Similar Description Conflict. For information on object conflicts, see "Object Conflicts" on page 340.
	Same Name Different Description Conflict. For information on object conflicts, see "Object Conflicts" on page 340.
	Same Description Different Name Conflict. For information on object conflicts, see "Object Conflicts" on page 340.

This section also includes:

- ▶ "Filter Dialog Box (Object Repository Merge Tool)" on page 358
- ▶ "Find Dialog Box (Object Repository Merge Tool)" on page 359
- ▶ "Merge Statistics Dialog Box" on page 361
- ▶ "Object Repository Merge Tool Menu Commands and Toolbar Buttons" on page 362

Filter Dialog Box (Object Repository Merge Tool)

This dialog box enables you to filter the target repository pane in the Object Repository Merge Tool Main Window. For more information, see "Object Repository Merge Tool Main Window" on page 352.



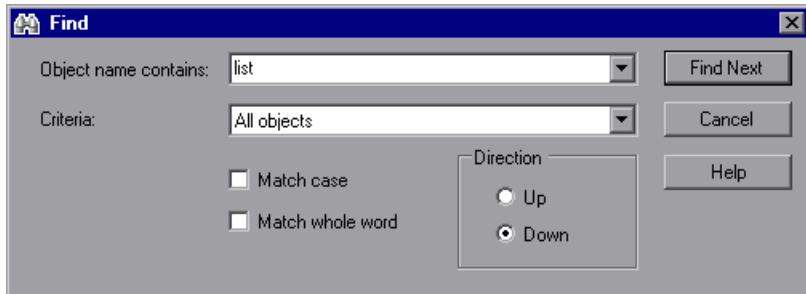
To access	<p>In the Object Repository Merge Tool main window, do one of the following:</p> <ul style="list-style-type: none"> ➤ Select Tools > Filter. ➤ Click the Filter  button in the toolbar.
Important information	<p>The filter only affects which objects are displayed in the target object repository pane. It does not affect which objects are included in the target object repository.</p>
Relevant tasks	<p>"How to Merge Two Shared Object Repositories" on page 344</p>

User interface elements are described below:

UI Elements	Description
Show all objects	<p>Instructs the Object Repository Merge Tool to display all objects in the target object repository</p>
Show only conflicting objects	<p>Instructs the Object Repository Merge Tool to display only objects in the target object repository that have conflicts.</p>

Find Dialog Box (Object Repository Merge Tool)

This dialog box enables you to find objects in the target object repository pane according to predefined search criteria. The located object is also highlighted in the relevant primary and/or secondary object repositories.



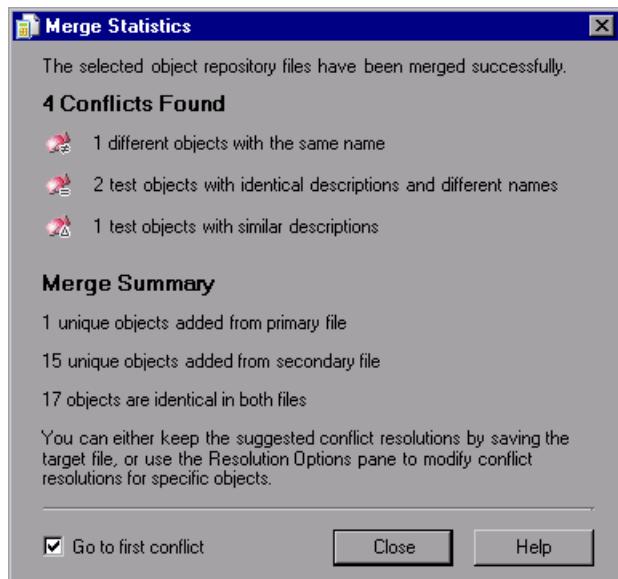
To access	<p>In the Object Repository Merge Tool main window, do one of the following:</p> <ul style="list-style-type: none"> ▶ Select Navigate > Find. ▶ Click the Find  button in the toolbar.
Important information	<p>After you set the Find options, you can close the Find dialog box and use menu commands or toolbar buttons to navigate to the next or previous node that matches your search criteria. For more information, see "Object Repository Merge Tool Menu Commands and Toolbar Buttons" on page 362.</p>
Relevant tasks	<ul style="list-style-type: none"> ▶ "How to Merge Two Shared Object Repositories" on page 344 ▶ "How to Update a Shared Object Repository From a Local Object Repository" on page 347
See also	<p>"Object Conflicts" on page 340</p>

User interface elements are described below:

UI Elements	Description
Object name contains	The full or partial name of the object you want to find.
Criteria	<p>The criteria to use to refine your search.</p> <p>The following criteria are available:</p> <ul style="list-style-type: none"> ➤ All objects ➤ Objects from one source ➤ Objects with conflicts ➤ Objects with conflicts or from one source
Match case	Instructs the Object Repository Merge Tool to distinguish between upper-case and lower-case characters in the search. When Match case is selected, QuickTest finds only those occurrences with the exact capitalization match to the text you entered in the Object name contains box.
Match whole word	Instructs the Object Repository Merge Tool to search only for whole word occurrences that match the text you entered in the Find Objects dialog box, and not as part of larger words.
Direction	<p>The direction from the current cursor location in which you want to search. The following options are available:</p> <ul style="list-style-type: none"> ➤ Up ➤ Down <p>The Find operation continues to search the entire file after it reaches the beginning or end of the object repository.</p>
Find Next	Highlights the next object that matches the specified criteria in the target object repository.

Merge Statistics Dialog Box

This dialog box displays the result of the merge, and the number and type of any conflicts that were resolved during the merge.



To access	In the Object Repository Merge Tool main window, do one of the following: <ul style="list-style-type: none"> ▶ Select View > Statistics. ▶ Click the Statistics button  in the toolbar.
Important information	▶ The Statistics dialog box shown after performing an Update from Local Repository merge differs slightly from the dialog box shown above.
Relevant tasks	<ul style="list-style-type: none"> ▶ "How to Merge Two Shared Object Repositories" on page 344 ▶ "How to Update a Shared Object Repository From a Local Object Repository" on page 347

User interface elements are described below:

UI Elements	Description
Conflicts Found	The number and type of any conflicts between the objects added to the target object repository. Conflict types are described in "Object Conflicts" on page 340.
Merge Summary	The number of objects merged from each object repository during the merge.
Go to first conflict	When selected, the Object Repository Merge Tool highlights the first conflict in the target object repository immediately after you close the Statistics dialog box.

Object Repository Merge Tool Menu Commands and Toolbar Buttons

The tables below describe commands available for:

- ▶ The Object Repository Merge Tool Main Window (described on page 352), for merging two shared object repositories
- ▶ The Object Repository Merge Tool - Multiple Merge Window (described on page 374), for updating objects from local object repositories

File Menu

	Command	Shortcut Key	Function
	New Merge (shared object repositories merge only)	CTRL+N	Enables you to specify two object repositories with which to perform a new merge operation.
	Save (shared object repositories merge only)	CTRL+S	Saves the merged shared object repository.

	Command	Shortcut Key	Function
	Save As (shared object repositories merge only)		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.
	Save and Merge Next (update from local repository only)		When merging multiple local object repositories, saves the current merge and merges the next local object repository.
	Revert to Original Merged Files (update from local repository only)		Cancels any manual conflict resolution adjustments and returns the target object repository to the original state at the time of the merge.
	ALM/QC Connection		Enables you to connect QuickTest to a Quality Center project. For more information, see "HP ALM Connection Dialog Box" on page 1635.
	Exit		Closes the Object Repository - Merge Tool window. If you did not yet save the merged repository, you are prompted to save it.

View Menu

	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 355 and "New Merge Dialog Box" on page 350.
	Secondary Repository (shared object repositories merge only)	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 355 and "New Merge Dialog Box" on page 350.
	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 353.
	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 355 and "Object Conflicts" on page 340.
	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 "Object Repository Merge Tool Menu Commands and Toolbar Buttons" on page 362.

	Command	Function
	Set as Default Layout	Enables you to save the current view so that each time you open the Object Repository - Merge Tool, 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 "Object Repository Merge Tool Menu Commands and Toolbar Buttons" on page 362.
	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 "Merge Statistics Dialog Box" on page 361.
	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

	Command	Shortcut Key	Function
	Next Conflict	F4	Navigates to the next conflicting object in the merged object repository.
	Previous Conflict	SHIFT+F4	Navigates to the previous conflicting object in the merged object repository.
	Find	CTRL+F	Opens the Find Dialog Box (Object Repository Merge Tool) (described on page 359).
	Find Next	F3	Navigates to the next object in the merged object repository according to the search specifications in the Find dialog box.
	Find Previous	SHIFT+F3	Navigates to the previous object in the merged object repository according to the search specifications in the Find dialog box.

Tools Menu

	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 "Settings Dialog Box (Object Repository Merge Tool)" on page 368.</p>
	Filter	<p>Opens the Filter dialog box, enabling you to show all of the objects in the Target Repository pane, or to show only the objects that had conflicts that were resolved during the merge. For more information, see "Filter Dialog Box (Object Repository Merge Tool)" on page 358.</p>

Help Menu

Command	Shortcut Key	Function
Object Repository Merge Tool Help	F1	Opens the Object Repository Merge Tool Help.

Settings Dialog Box (Object Repository Merge Tool)

The Object Repository Merge Tool uses predefined settings when merging object repositories or when updating a shared object repository from local object repositories.

You can change these settings at any time. After you change the settings, all new merges are performed according to the new settings.

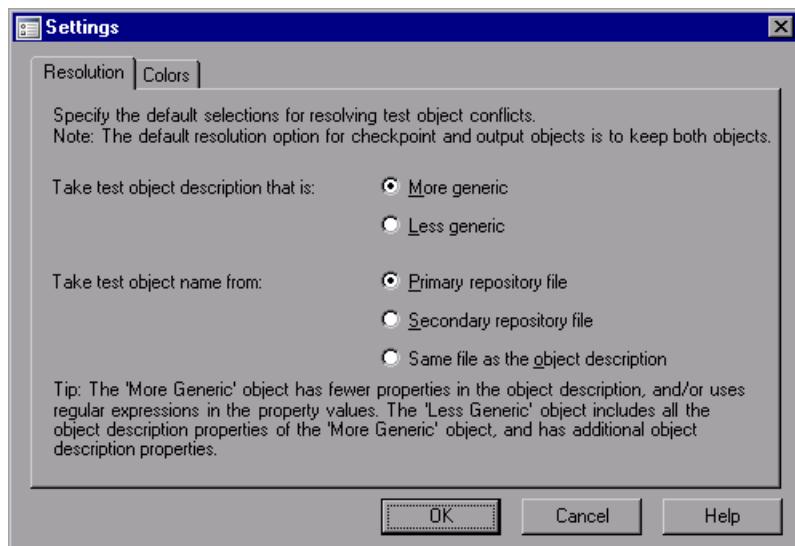
The Settings dialog box contains the following tabs:

- **Resolution Tab** (described on page 369). Enables you to 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).
- **Colors Tab** (described on page 372). Enables you to specify the text color of the object names that are displayed in the target object repository.

Tip: If the New Merge dialog box opens when you open the Object Repository Merge Tool, and you want to change the settings before merging two object repositories, click **Cancel** to close the New Merge dialog box, change the settings as described in the next sections, and then open this dialog box again to perform the merge.

Resolution Tab (Settings Dialog Box - Object Repository Merge Tool)

This tab enables you to 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.



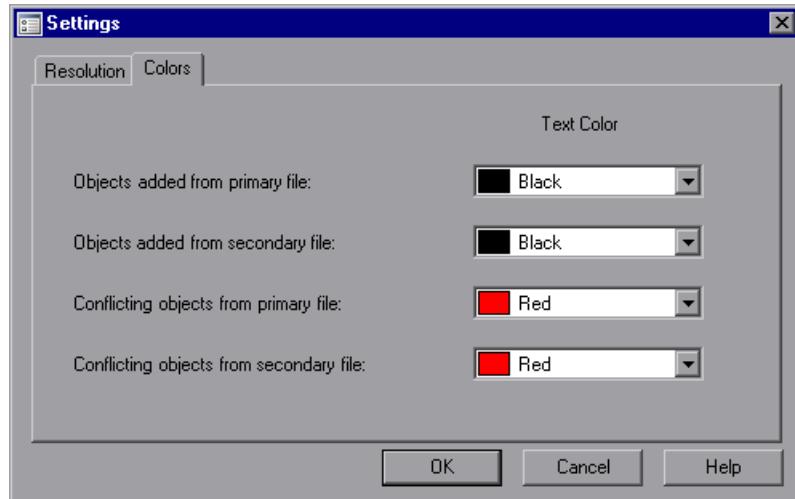
To access	In the Object Repository Merge Tool , do one of the following: <ul style="list-style-type: none"> ➤ Select Tools > Settings, and select the Resolution tab. ➤ Click the Settings  button in the toolbar, and select the Resolution tab.
Important information	<ul style="list-style-type: none"> ➤ The resolution settings are relevant only for test objects. Conflicts between checkpoint or output value objects with the same name but different content are always resolved by merging both objects into the new repository and renaming one of them. ➤ When updating a shared object repository from a local object repository, the object repositories are referred to as the Local and Shared object repository. ➤ 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. If you click No, the new settings will apply only to future merges.
Relevant tasks	<ul style="list-style-type: none"> ➤ "How to Merge Two Shared Object Repositories" on page 344 ➤ "How to Update a Shared Object Repository From a Local Object Repository" on page 347
See also	<ul style="list-style-type: none"> ➤ "Object Conflicts" on page 340 ➤ "Colors Tab (Settings Dialog Box - Object Repository Merge Tool)" on page 372

User interface elements are described below:

UI Elements	Description
Take test object description that is	<p>Specifies how to resolve conflicts in which two test objects have the same name, but their descriptions differ. You can specify that the target object repository takes the test object description that is more generic or less generic, as follows:</p> <ul style="list-style-type: none"> ▶ More generic. Instructs the Object Repository Merge Tool to take the test object that has fewer identifying properties than the test 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 test object that has all the identifying properties of the test object with which it conflicts, plus additional identifying properties.
Take test object name from	<p>Specifies how to resolve conflicts where two test objects have the same or similar descriptions, but their names differ. You can select the source from which the target test object repository takes the object name, as follows:</p> <ul style="list-style-type: none"> ▶ Primary repository file. The target object repository takes the test object name from the test 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 test object name from the test 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.

Colors Tab (Settings Dialog Box - Object Repository Merge Tool)

This tab enables you to specify the color in which object names are displayed in the target object repository according to their source, and whether they caused a conflict.



To access	<p>In the Object Repository Merge Tool, do one of the following:</p> <ul style="list-style-type: none"> ► Select Tools > Settings, and select the Colors tab. ► Click the Settings  button in the toolbar, and select the Colors tab.
Important information	<p>When performing an Update from Local Repository operation, the options in the Colors tab of the Settings dialog box also apply to objects added from the local (primary) and shared (secondary) object repositories.</p>

Relevant tasks	"How to Merge Two Shared Object Repositories" on page 344 "How to Update a Shared Object Repository From a Local Object Repository" on page 347
See also	► "Object Conflicts" on page 340 ► "Resolution Tab (Settings Dialog Box - Object Repository Merge Tool)" on page 369

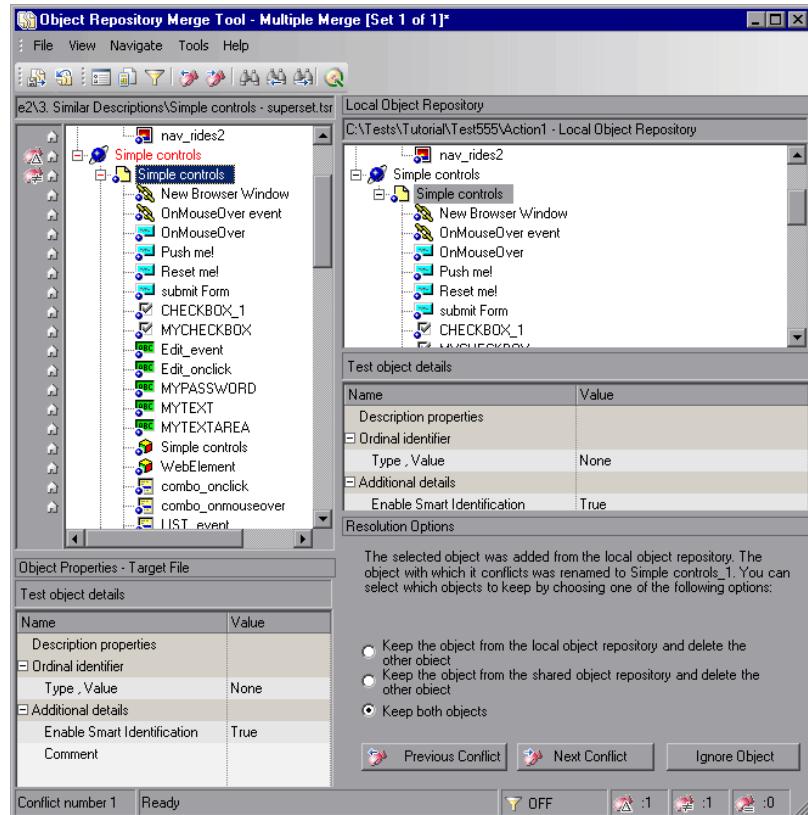
User interface elements are described below (unlabeled elements are shown in angle brackets):

UI Elements	Description
<object criteria>	The list of object criteria on the left side of the dialog box.
Text Color	Displays the current color used for the object names that match the corresponding object criteria.

Object Repository Merge Tool - Multiple Merge Window

This window displays merge results of the selected local object repositories into the target shared object repository. The local object repositories are merged one by one into the shared object repository.

The active local object repository is treated as the primary object repository, and the shared object repository is treated as the target object repository.



To access	In the Object Repository Manager, select Tools > Update from Local Repository .
Important information	<ul style="list-style-type: none"> ▶ 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 image above shows the merge results of the first merge (the local object repository of the first action being merged into the shared object repository). ▶ The number of each merge set in a multiple merge is displayed in the title bar, for example, [Set 2 of 3]. ▶ When you click Save and Merge Next, the current merge is saved and cannot be modified without performing another merge. ▶ The Ignore Object button is visible in the Merge Tool window only for a local object repository merge, and is only enabled when an object in the local object repository is selected. ▶ 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).
Relevant tasks	"How to Update a Shared Object Repository From a Local Object Repository" on page 347

The Object Repository Merge Tool - Multiple Merge window contains the following key elements:

- ▶ "Menu Bar and Toolbar" on page 376
- ▶ "Target Repository Pane" on page 376
- ▶ "Primary Repository Pane" on page 376
- ▶ "Resolution Options Pane" on page 376
- ▶ "Status Bar" on page 377

Menu Bar and Toolbar

Displays menus of Object Repository Merge Tool commands. These commands are described in "Object Repository Merge Tool Menu Commands and Toolbar Buttons" on page 362.

Target Repository Pane

Displays the objects that were added from the local object repositories to the shared object repository.

At the left of each object in the target object hierarchy is an icon that indicates the source of the objects:

UI Elements	Description
	Indicates that the object was added from the local object repository.
	Indicates that the object already existed in the shared object repository.

Primary Repository Pane

Displays the objects in the local object repository that you are currently merging. For more information, see "Update from Local Repository Dialog Box" on page 378.

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 355.

In the Resolution Options pane, you can select from one of the following options:

- 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.
- **Ignore Object.** Exclude a specific local repository object from the merge process so that it is not included in the shared object repository. 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.



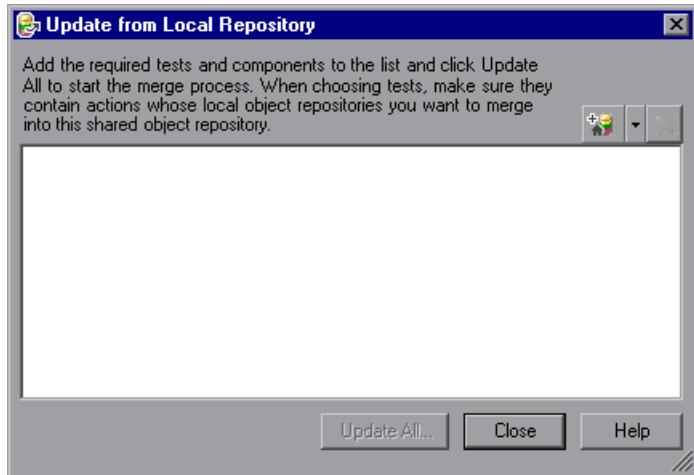
Caution: The **Ignore Object** operation cannot be reversed. To include the object again in the merge process, you must repeat the merge by clicking **Revert to Original Merged Files** in the toolbar.

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 356.

Update from Local Repository Dialog Box

This dialog box enables you to select tests containing actions, whose local object repositories you want to merge into a shared object repository.



To access	In the Object Repository Manager, select Tools > Update from Local Repository .
-----------	---

Important information	<ul style="list-style-type: none"> ▶ You can select multiple tests. ▶ If you are currently connected to a Quality Center project, you can select tests from the file system or from Quality Center. ▶ You can add only tests containing actions that are associated with the shared object repository you are updating and whose local object repositories contain objects. ▶ 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.
Relevant tasks	"How to Update a Shared Object Repository From a Local Object Repository" on page 347

User interface elements are described below (unlabeled elements are shown in angle brackets):

UI Elements	Description
<selected tests>	The list of tests that contain local object repositories to include in the merge.
	Add Tests. Enables you to browse to a test and add it to the list. Click the down arrow ▾ to browse for tests.
Update All	<p>Instructs the Object Repository Merge Tool to automatically merge the local object repositories with the shared object repository according to the preconfigured settings and opens the Object Repository Merge Tool. You can then modify any conflict resolutions if necessary.</p> <p>Note: If you selected multiple tests, merges are performed one after the other, and you can review the shared object repository and modify any conflict resolutions after each merge.</p>

Part III

Designing Tests

10

Test Creation Overview

This chapter includes:

Concepts

- Methodologies for Creating Tests on page 384
- Enhancing Your Tests on page 387
- Sample Test on page 389
- Relative Paths in QuickTest on page 391
- Portable Copies of Tests on page 395
- Opening and Saving Tests with Locked Resources on page 396

Tasks

- How to Perform File Operations on Test Files on page 398

Reference

- Open Test Dialog Box on page 401
- Open <Resource> Dialog Box on page 406
- Save Test Dialog Box on page 412
- Save <Resource> Dialog Box on page 417
- Save Test with Resources Dialog Box on page 422
- Export to Zip File Dialog Box on page 426
- Import from Zip File Dialog Box on page 427
- Print Dialog Box on page 428

Troubleshooting and Limitations - Opening and Saving Testing Documents on page 430

Concepts

Methodologies for Creating Tests

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.

This section includes:

- "Creating Tests Using the Keyword-Driven Methodology" on page 384
- "Creating Tests By Recording Steps on Your Application" on page 386
- "Combining Methodologies to Create Tests" on page 387

Creating Tests Using the Keyword-Driven Methodology

This 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.

When to Use Keyword-Driven Testing

- 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 "Test Creation - Keyword-Driven Methodology" on page 435.

Creating Tests By Recording Steps on Your Application

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.

When to Record 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 "Recording Tests - Overview" on page 464.



Combining Methodologies to Create Tests

You can use a combination of the keyword-driven and recording methodologies (described above) to create your tests.



Enhancing Your Tests

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

You can also 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 15, "Checkpoints Overview."

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

When you test your application, you may want to check how it performs the same operations with different data. You can do this by replacing fixed values with values from an external source during your run session. This is called **parameterizing** your test. You can supply data from a data table, environment variables you define, or values that QuickTest generates during the run session. For more information, see Chapter 22, "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 23, "Output Values."

Actions

You can divide your test into actions to streamline the testing process of your application. For more information, see "Actions Overview" on page 523.

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 26, "User Interface-Based Programming Operations."

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 27, "Working in the Expert View and Function Library Windows."

Active Screen Updates

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 "How to Update Test Object Descriptions, Checkpoints, Output Values, or Active Screens" on page 1252.

Sample Test

The following is a sample test of a login procedure to the Mercury Tours site, the sample Web site. When you create tests, QuickTest creates both a graphical representation and script of the steps you perform on your application.

The graphical representation of these steps is displayed in the Keyword View tab.

Item	Operation	Value	Documentation
Action1			
Welcome: Mercury Tours			
Welcome: Mercury Tours			
userName	Set	"tutorial"	Enter "tutorial" in the "userName" edit field.
password	SetSecure	"477cdb71935682eda"	Enter the encrypted string "477cdb71935682eda" in the "password" edit field.
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 test object.
<code>userName</code> Set "tutorial"	userName is the name of the edit box test object. Set is the method performed on the edit box. tutorial is the value property of this edit box.
<code>password</code> SetSecure "4082986e39ea38c6efc"	password is the name of the edit box test object. 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 test object. Click is the method performed on the image. 2, 2 are the x- and y-coordinates where the image was clicked.

The Expert View displays these same steps 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
```



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 394.)

Notes:

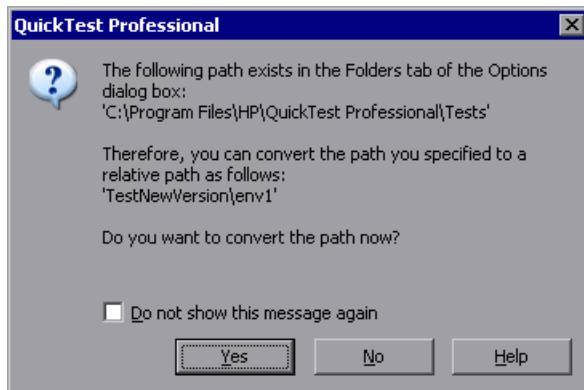
- Prior to QuickTest 9.0, if you specified a path for a resource starting with \.., it was considered to be a relative path. In QuickTest 9.0 and later, a path that starts with \.. is considered to be a full path, with the backslash representing the root folder of the current drive. If you defined paths starting with \.. using earlier versions of QuickTest, you should change the path to be a standard relative path by removing the backslash (\).
 - If you are working with the Resources and Dependencies model with Quality Center 10.00 or HP ALM, specify an absolute Quality Center path. For more information, see "Relative Paths and Quality Center" on page 1656.
-

When you specify a path to a function library, shared object repository, recovery scenario, data table file, 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 message boxes, depending on whether the path you specified, or a part of the path, exists in the Folders pane. (For more information on the Folders pane, see "Folders Pane (Options Dialog Box)" on page 1431.)

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.

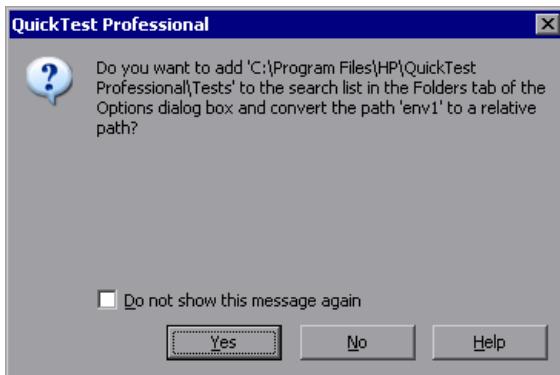


- Clicking **Yes** truncates the path to a relative path.
- Clicking **No** defines the path to the resource as an absolute path.

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.



- Clicking **Yes** adds the resource's location path to the Folders pane and truncates the path to a relative path.
- Clicking **No** defines the path to the resource as an absolute path.

Notes:

- You can choose not to show one or both of these message boxes when you enter a path to a resource by selecting the **Do not show this message again** check box. To show these message 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.
 - If you are connected to Quality Center 10.00 or HP ALM, these message boxes are displayed only if you select a path in the file system or in a Quality Center 9.2 project.
-



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.

- 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.

If you are working with the Resources and Dependencies model with Quality Center 10.00 or HP ALM, you must specify an absolute Quality Center path to enable your tests to access your resource files.

- 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. If you are not working with the Resources and Dependencies model with Quality Center 10.00 or HP ALM, we recommend that you use relative paths when saving resources in QuickTest.

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.

Portable Copies of 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.
- **Calls to Service Test tests.** If your test contains a call to a Service Test test, QuickTest saves a copy of that test together with the QuickTest-generated XML file in which the parameter values are defined.

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*.

Opening and Saving Tests with Locked Resources

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

QuickTest resource files are locked in the following scenarios:

- The file is being used by another QuickTest user.
- The file is checked into a Quality Center version control database or any other version control software.
- The file is marked as read-only.

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

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

- Opening a test whose QuickTest resource file is locked. For more information, see "Open Test with Locked Resources - Message Box" on page 404.
 - Saving a test whose external resource files were not opened in read-write mode (because they were locked when you opened the test) and modified while editing the test. For more information, see "Save Tests with Locked Resources - Message Box" on page 416.
-

Tip: To create a writable copy of a QuickTest test with locked resources, use the Save As option from the File menu and save the test under a new name.

Tasks

How to Perform File Operations on Test Files

This task describes how to use the **File** menu to create, open, print, save, zip, and unzip tests, as well as create standalone, portable tests and open tests with locked resources.

This section includes the following steps:

- "Create a new test" on page 398
- "Open an existing test" on page 398
- "Print a test" on page 399
- "Save a test" on page 399
- "Save a portable copy of a test" on page 400
- "Zip and unzip a test" on page 400

Create 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.

Open an existing test

- 1 If your test is stored in Quality Center, connect to the relevant Quality Center server and project. For more information, see "HP ALM Connection Dialog Box" on page 1635.
- 2 Do one of the following:
 - Select the test in the Open Test dialog box, which you open by selecting **File > Open > Test**. For more information, see "Open Test Dialog Box" on page 401.
 - Select the test from the list of recent files list in the **File** menu.

Note: For details on opening a test with locked resources, see "Opening and Saving Tests with Locked Resources" on page 396.

Tip: For troubleshooting information on opening tests, see "Troubleshooting and Limitations - Opening and Saving Testing Documents" on page 430.

Print a test

- To print an action from the Keyword View, select **File > Print** and use your computer's default print dialog box.
- To print an action from the Expert View, select **File > Print** and use the Print Dialog Box (described on page 428).
- To display the Keyword View on screen as it will look when printed, select **File > Print Preview**. (This option is not relevant for the Expert View.)

Save a test

Do one of the following:



- Click the **Save** button or select **File > Save** to save the test.
- Select **File > Save As** to save the test with a different name.

If this is the first time you are saving the test, or if you are using the **Save As** option, the Save QuickTest Test dialog box opens. For more information, see "Save Test Dialog Box" on page 412.

Tip: For troubleshooting information on saving tests, see "Troubleshooting and Limitations - Opening and Saving Testing Documents" on page 430.

Save a portable copy of a test

- 1 If your test was created using an earlier version of QuickTest, make sure that the test and its resources are upgraded to the current version of QuickTest in one of the following ways:
 - 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), open and save those tests, too.
 - If your tests are stored in Quality Center, you can use the QuickTest Professional Asset Upgrade Tool for Quality Center to convert your test's attached resource files to linked assets and upgrade your tests to the current version of QuickTest.
- 2 Do one of the following:



- Click the **Save Test with Resources** toolbar button.
- Select **File > Save Test with Resources**.

The Save Test with Resources dialog box opens. For more information, see "Save Test with Resources Dialog Box" on page 422.

Zip and unzip a test

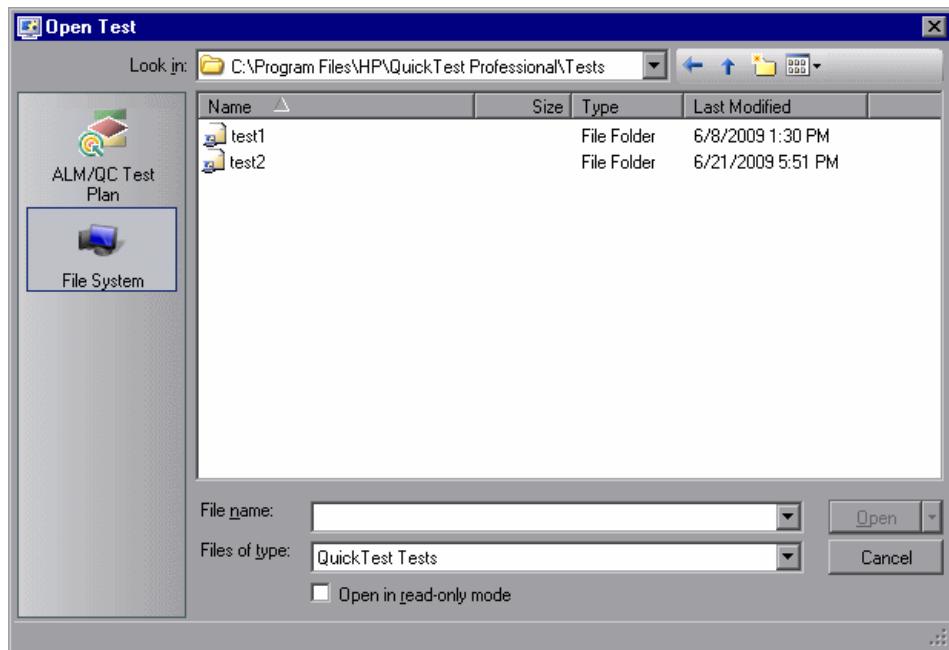
- To **zip** a test, select **File > Export Test to Zip File** and use the Export to Zip File Dialog Box (described on page 426).

This enables you to save configuration, run-time, setup data, and (optionally) Active Screen files together, conserving space and making it easier to move a test.
- To **unzip** a test, select **File > Import Test from Zip File** and use the Import from Zip File Dialog Box (described on page 427).

Reference

Open Test Dialog Box

This dialog box enables you to open an existing test from the file system or your Quality Center project (if QuickTest is currently connected to a Quality Center project).



To access	<p>1 Prerequisite for Quality Center: Connect to your Quality Center project, if needed. For information see, "HP ALM Connection Dialog Box" on page 1635.</p> <p>2 Use one of the following:</p> <ul style="list-style-type: none"> ➤ Select File > Open > Test. ➤ Click the Open down arrow and select Test.
Important information	<p>When you save a new test to the file system, QuickTest suggests a default folder called Tests. For all supported operating systems prior to Windows Vista, this folder is located under your QuickTest Professional installation folder. For Windows Vista and later operating systems, this folder is located under MyDocuments\HP\QuickTest Professional.</p> <p>Tip: You can also open a test by dragging it onto the document area from the file system.</p>

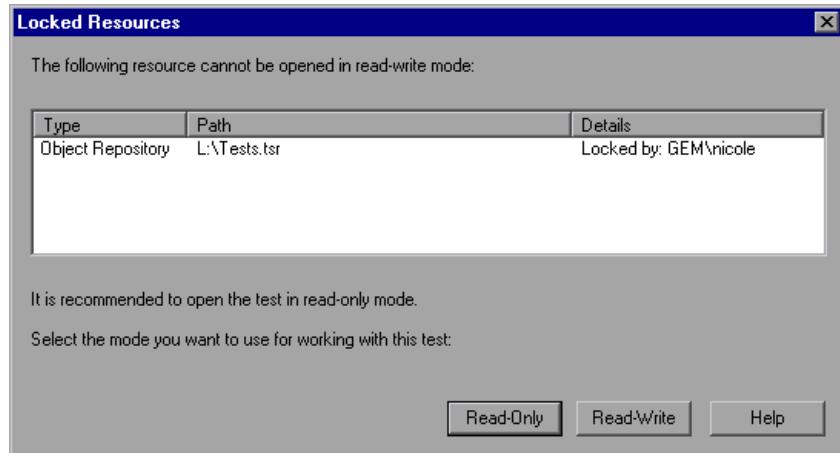
User interface elements are described below (unlabeled elements are shown in angle brackets):

UI Elements	Description
<sidebar>	<p>Displays the location in which the test is stored, for example, File System and ALM/QC Test Plan.</p> <p>Note: You may need to select a different location before browsing to your test.</p>
Look in	<p>Lists the path for the test. You can use the down arrow to navigate to the required folder, or you can double-click a folder in the test list area to navigate to the required folder.</p>
<test list area>	<p>Displays the folders and/or tests stored in the current path.</p> <p>Note for Quality Center users: If the test is stored in a Quality Center project with version control support, you can view version control information for the test by clicking the Views down arrow and selecting Details. For more information, see "Checking Assets Out of the Version Control Database" on page 1700.</p>
File name	<p>Displays the name of the test selected in the test list area.</p>

UI Elements	Description
Files of type	Filters the list of displayed assets to show only QuickTest Tests . If the current folder contains other file types, they are hidden.
Open in read-only mode	<p>Select this check box to open the test in read-only mode. This option enables you to view the test but not modify it.</p> <p>Note for Quality Center users: A test also opens in read-only mode if:</p> <ul style="list-style-type: none"> ▶ You opened a test that is currently checked in to the version control database (for projects that support version control) ▶ You opened a test that is currently checked out to another user (for projects that support version control) ▶ You opened a test from an earlier version of Quality Center, and the test has not yet been updated to the current format. <p>For details, see "Managing Versions of Assets in Quality Center Overview" on page 1696.</p>
Open	<p>Click this button to open the selected test.</p> <p>Note: If the test is checked into a version-control-enabled Quality Center project, the Open button contains a down arrow with an Open and Check out option, 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 1700.</p>

Open Test with Locked Resources - Message Box

This message box opens when you try to open a test that is associated with a locked QuickTest resource file, such as a locked data table or object repository file.



To access	Open a test with a locked resource file, such as a locked data table or shared object repository.
Important Information	When you try to open a test with locked resource files, the locked resource files open in read-only mode. You can open the test in Read-Only or Read-Write mode.
See also	"Opening and Saving Tests with Locked Resources" on page 396

User interface elements are described below:

UI Elements	Description
Read-Only	<p>Opens the test in read-only mode, enabling you to run it but not make any changes to it. All editing options are disabled and you cannot edit the test in the Keyword View or the Expert View.</p> <p>Note: Even though the test itself cannot be saved, if you run the test and choose to save the results, the results file is saved.</p> <p>You can choose the Save As option from the File menu, save the test under a new name, and then edit the test. The same is true for a shared object repository file opened in read-only mode.</p>
Read-Write	<p>Opens the test in read-write mode, enabling you to run and edit it, and save any changes you make to it. However, any changes to locked, QuickTest resource files cannot be saved. Locked resource files are opened in read-only mode even if the test is opened in read-write mode.</p> <p>Example: Suppose you open a test in read-write mode that has a locked data table file associated with it. When you edit the test, any changes to the data table will not be saved. If any of the data table values change as a result of editing the test, and you save the changes to the test, the test may fail the next time it calls the unmodified data table values.</p> <p>It is therefore recommended to open the test in read-only mode or to close the test without saving.</p>

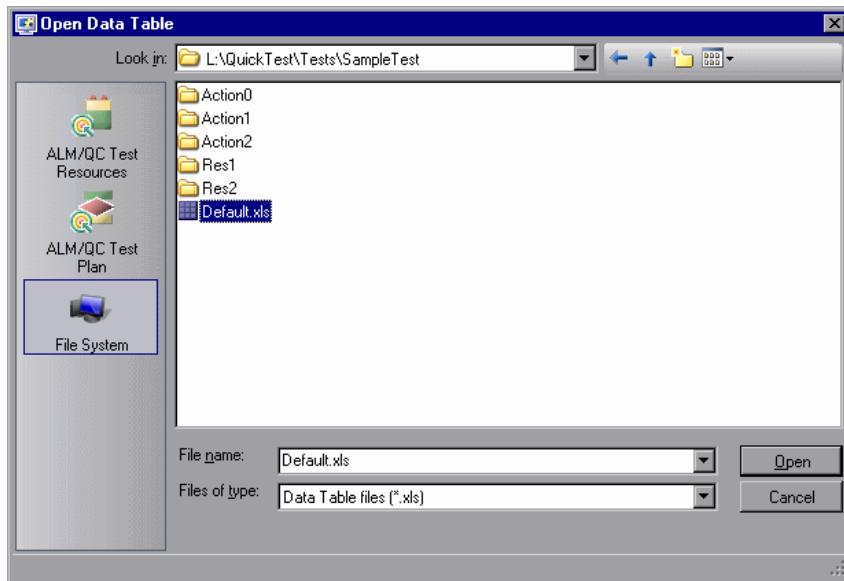
Open <Resource> Dialog Box

This section describes the following dialog boxes:

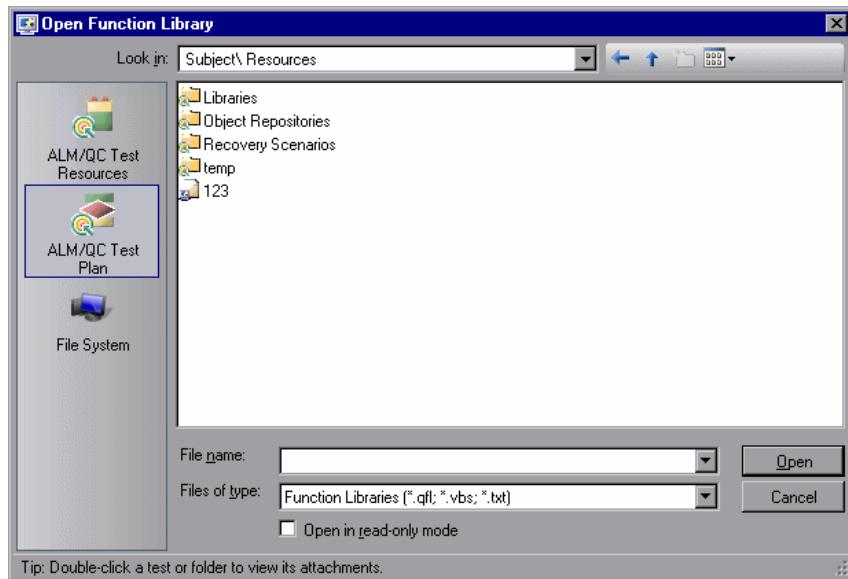
- Open Data Table
- Open Function Library
- Open Recovery Scenario
- Open Shared Object Repository
- Open XML File
- Open Environment Variable File

This dialog box enables you to open an existing resource file from the file system or your Quality Center project (if QuickTest is currently connected to a Quality Center project).

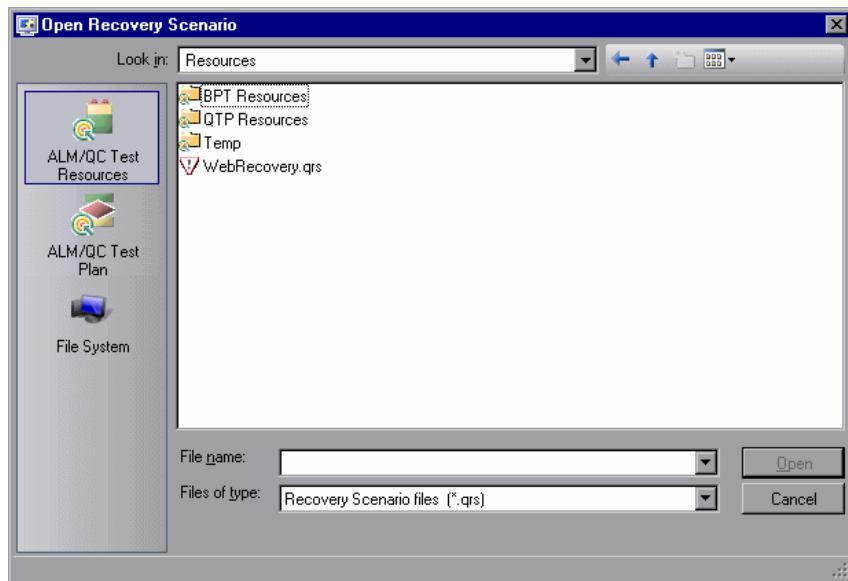
Open Data Table



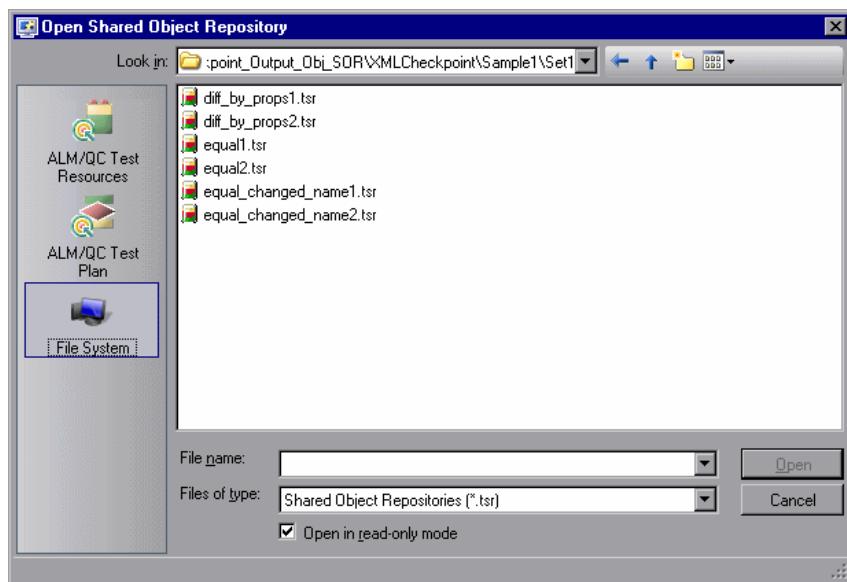
Open Function Library



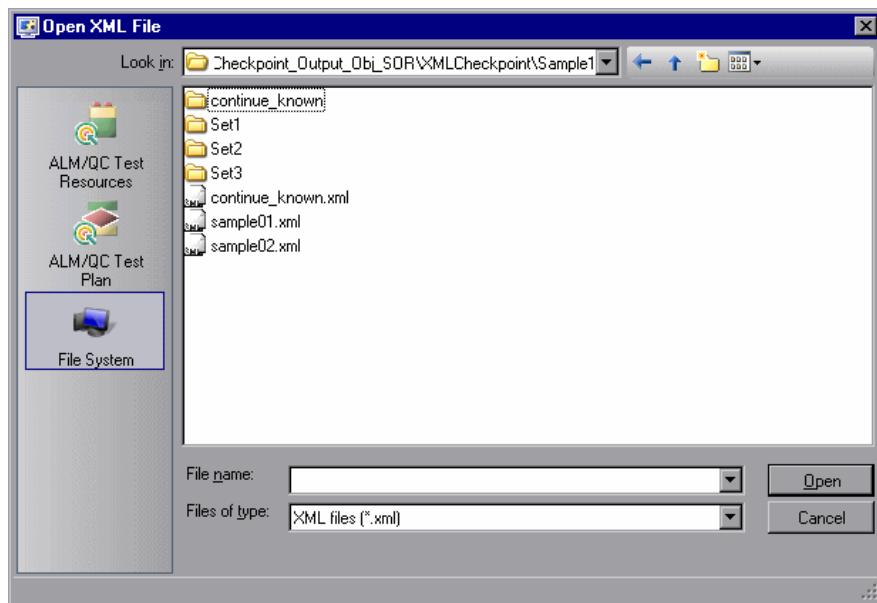
Open Recovery Scenario



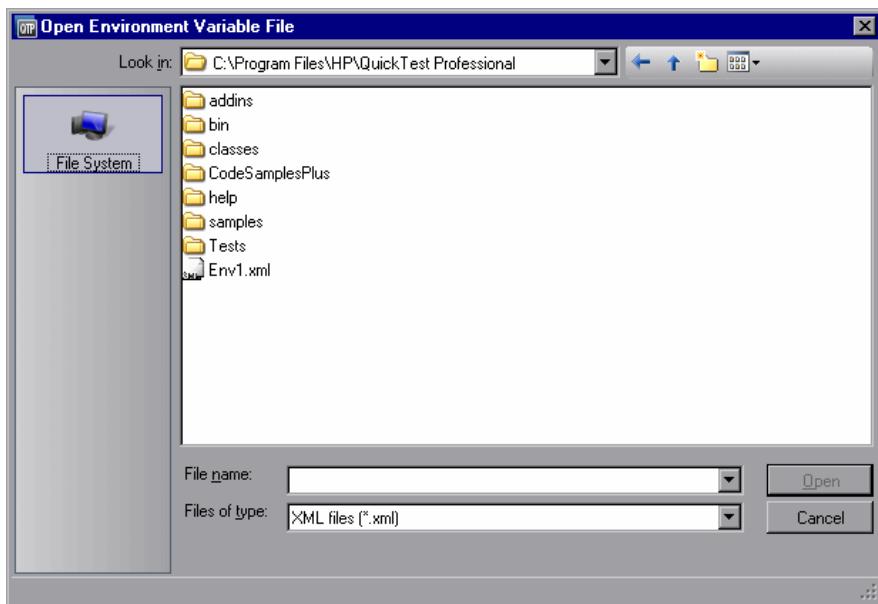
Open Shared Object Repository



Open XML File



Open Environment Variable File



To access:	
Prerequisite for Quality Center	Make sure you are connected to your Quality Center project. For details see, "HP ALM Connection Dialog Box" on page 1635.
Open Data Table dialog box	Select File > Settings > Resources tab > Other location radio button. Then click the browse button.
Open Function Library dialog box	Select File > Open > Function Library .
Open Recovery Scenario dialog box	<p>Use one of the following:</p> <ul style="list-style-type: none"> ▶ Select File > Settings > Recovery node > Add Recovery Scenario button. ▶ Select Resources > Recovery Scenario Manager > Open button. <p>For details on recovery scenarios, see "Recovery Scenarios Overview" on page 1524.</p>

Open Shared Object Repository dialog box	<p>Use one of the following:</p> <ul style="list-style-type: none"> ➤ Select Resources > Associate Repositories > Add Repository button. ➤ Select Resources > Object Repository Manager. Then, in the Object Repository Manager window, select File > Open or click Open .
Open XML File dialog box	<p>Use one of the following:</p> <ul style="list-style-type: none"> ➤ Select Insert > Checkpoint > XML Checkpoint (from resource). Then, in the XML Source Selection - Checkpoint Properties dialog box, select Create checkpoint from XML file and click the browse button. ➤ Select Insert > Output Value > XML Output (from resource). Then, in the XML Source Selection - Output Value Properties dialog box, select Create output value step from XML file and click the browse button.
Open Environment Variable File dialog box	<ol style="list-style-type: none"> 1 Select File > Settings > Environment tab > User-defined type. 2 Select the Load variables and values from external file checkbox and click the browse button.

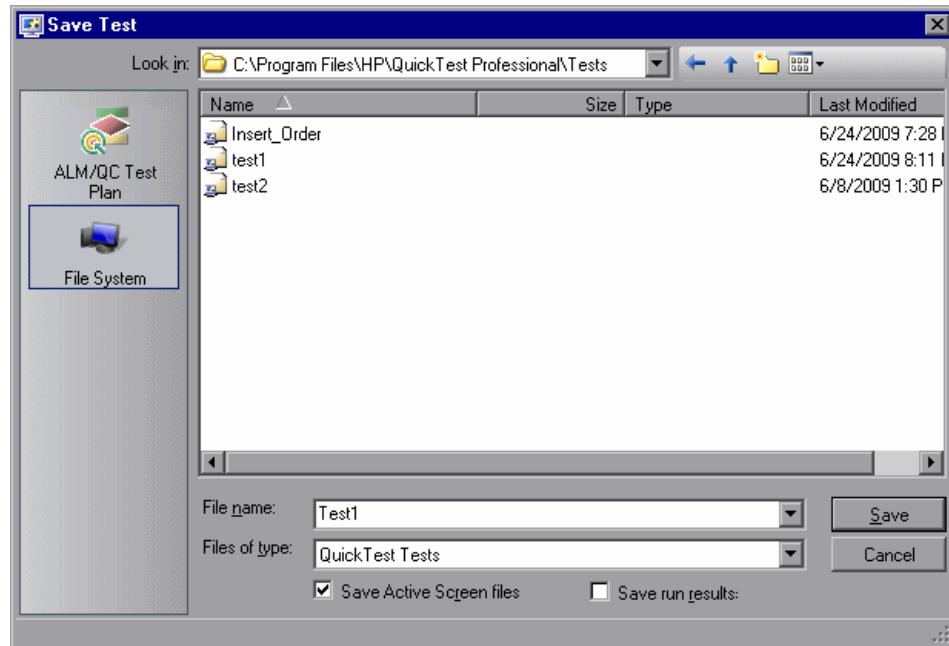
Important Information	<ul style="list-style-type: none"> ➤ If you are working with the QuickTest Script Editor, only the Open Function Library dialog box is relevant. For details on the Script Editor, see "QuickTest Script Editor" on page 1571. ➤ Version control. If the resource is checked into a version-control-enabled Quality Center project, the Open button contains a down arrow with an Open and Check out option, enabling you to open the resource and immediately check it out. For more information, see "Checking Assets Out of the Version Control Database" on page 1700.
See also	For details on a specific type of resource, see the relevant section in this guide.

User interface elements are described below (unlabeled elements are shown in angle brackets):

UI Elements	Description
<sidebar>	<p>The location in which the resource file is stored, for example, File System or ALM/QC Test Resources.</p> <p>Note: You may need to select a different location before browsing to your file.</p>
Look in	<p>The path for the resource file. You can use the down arrow to navigate to the required folder, or you can double-click a folder in the resource list area to navigate to the required folder.</p>
<resource list area>	<p>The folders and/or resource files stored in the current path.</p> <p>Note: If the file is stored in a Quality Center project with version control support, you can view version control information for the file by clicking the Views down arrow and selecting Details. For more information, see "Checking Assets Out of the Version Control Database" on page 1700.</p>
File name	<p>The name of the resource file selected in the resource list area.</p>
Files of type	<p>Filters the list of displayed assets to show only the file types for the resource. If the current folder contains other file types, they are hidden. To display all file types, select All files.</p>
Open in read-only mode (function libraries and shared object repositories)	<p>Opens the resource in read-only mode. This option enables you to view and associate the resource but not to modify it.</p>

Save Test Dialog Box

This dialog box enables you to save a new or existing test, or to save an existing test with another name. You can save a test to the file system and your Quality Center project (if QuickTest is currently connected to a Quality Center project).



To access	<p>Prerequisite for Quality Center: Connect to your Quality Center project, if needed. For information see, "HP ALM Connection Dialog Box" on page 1635. Then continue as described below for all users.</p> <p>To save a new test do one of the following:</p> <ul style="list-style-type: none"> ▶ Select the File > Save menu command. ▶ Click the Save toolbar button . <p>To save an existing test with another name:</p> <p>Select the File > Save As menu command.</p>
Important information	<ul style="list-style-type: none"> ▶ 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. ▶ If changes are made to an existing test, an asterisk (*) is displayed in the title bar until the test is saved.

User interface elements are described below (unlabeled elements are shown in angle brackets):

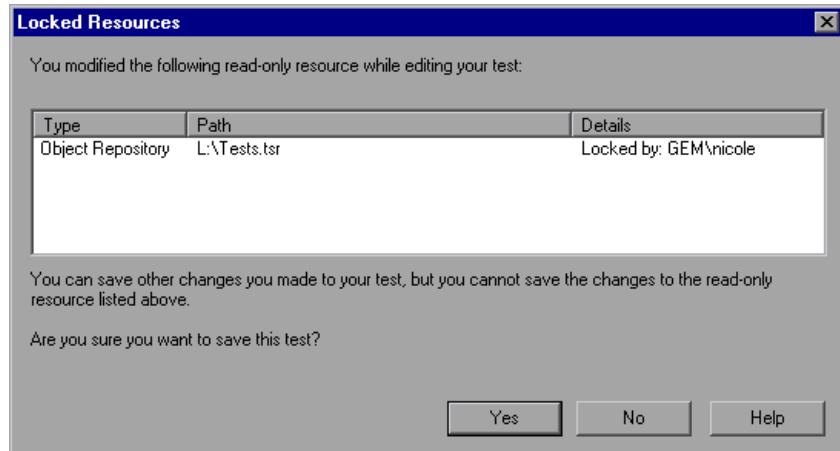
UI Elements	Description
<sidebar>	<p>Displays the location in which the test is stored, for example, File System and ALM/QC Test Plan.</p> <p>Note: You may need to select a different location before saving your test.</p>
Look in	<p>Lists the path for the test. You can use the down arrow to navigate to the required folder, or you can double-click a folder in the test list area to navigate to the required folder.</p>
<test list area>	<p>Displays the folders and/or tests stored in the current path.</p> <p>Note: If the test is stored in a Quality Center project with version control support, you can view version control information for the test by clicking the Views down arrow and selecting Details. For more information, see "Checking Assets Out of the Version Control Database" on page 1700.</p>

UI Elements	Description
File name	<p>The name of the test. Use a descriptive name that will help you and others identify the test easily.</p> <p>Naming conventions: See "Naming Conventions" on page 1779.</p>
Files of type	<p>Filters the file types displayed in the test list area. You can choose to display only QuickTest Tests or all file types.</p>
Save Active Screen files	<p>Saves the Active Screen files with your test. 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.</p> <p>Tips:</p> <ul style="list-style-type: none"> ➤ To conserve disk space after you finish designing the test and are using the test only for test runs, clear the Save Active Screen files check box. ➤ To regenerate Active Screen information if you clear the Save Active Screen files check box and later want to edit your test using Active Screen options, perform an Update Run operation. For more information, see "How to Update Test Object Descriptions, Checkpoints, Output Values, or Active Screens" on page 1252. ➤ To set Active Screen preferences, such as instructing QuickTest not to capture Active Screen files while recording or to only capture Active Screen information under certain conditions, you can modify the Active Screen pane of the Options dialog box (Tools > Options > Active Screen node). For more information, see "Active Screen Pane (Options Dialog Box)" on page 1434. <p>For more information on the Active Screen and its uses, see "Active Screen Overview" on page 1308.</p>

UI Elements	Description
Save run results	<p>Saves any existing run results with your test. (The check box is available only for tests that are saved in the file system and that ran at least once.)</p> <p>If you clear this check box, your run result files will not be saved, and you will not be able to view them later.</p> <p>Tip: To conserve disk space if you do not require the run results for later analysis, or if you are saving an existing test under a new name and do not need the run results, clear the Save run results check box.</p> <p>For more information on run results, see "Run Results Viewer Overview" on page 1086.</p>

Save Tests with Locked Resources - Message Box

This message box opens if anything in a locked resource file has changed when you save a test with locked resources that you opened in read-write mode.



To access	Save a test with a locked resource file that was modified during the current QuickTest session.
Important Information	It is recommended not to save a test whose locked resource files have been modified. If you do save the test, the next time you run the test, the test may fail because the test may expect different values from the resource files.
See also	"Opening and Saving Tests with Locked Resources" on page 396

Save <Resource> Dialog Box

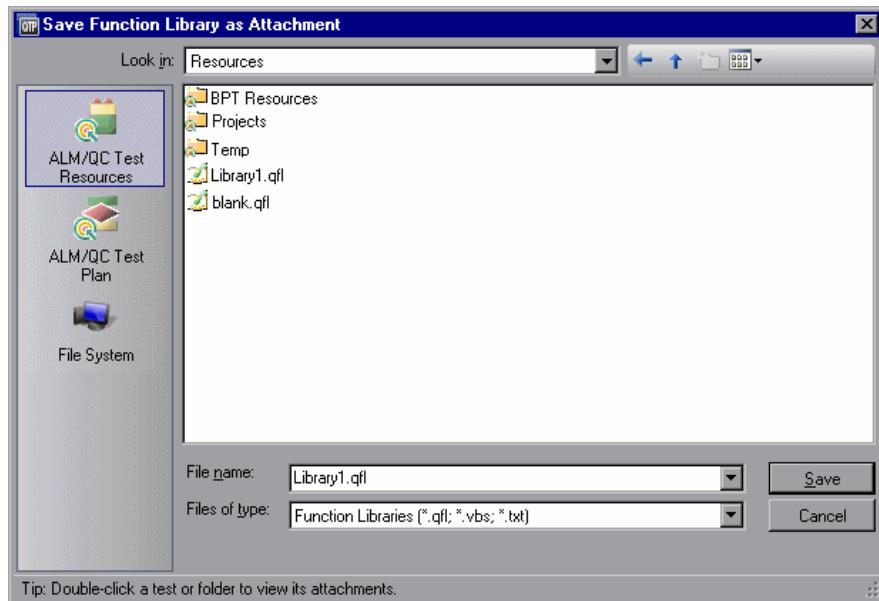
This section describes the following dialog boxes:

- ▶ Save Function Library
- ▶ Save Recovery Scenario
- ▶ Save Shared Object Repository
- ▶ Save Environment Variable File

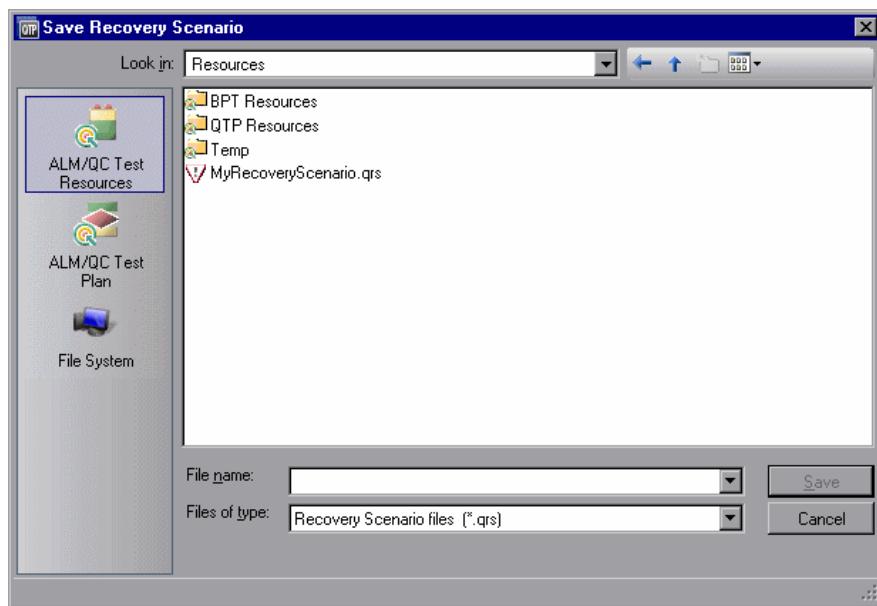
This dialog box enables you to save an existing resource file to the file system or your Quality Center project (if QuickTest is currently connected to a Quality Center project).

Save Function Library

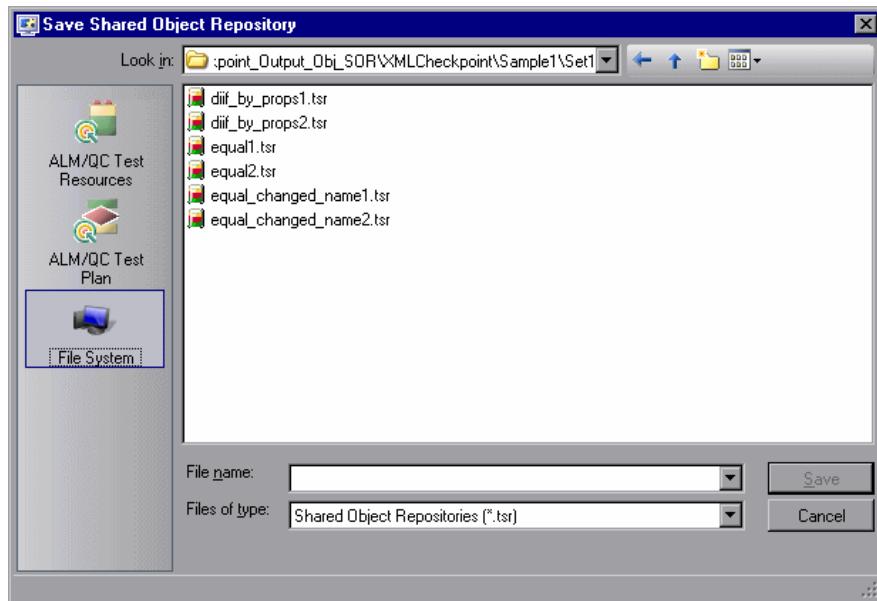
This image shows the dialog box when saving a function library as an attachment in the Quality Center Test Plan module.



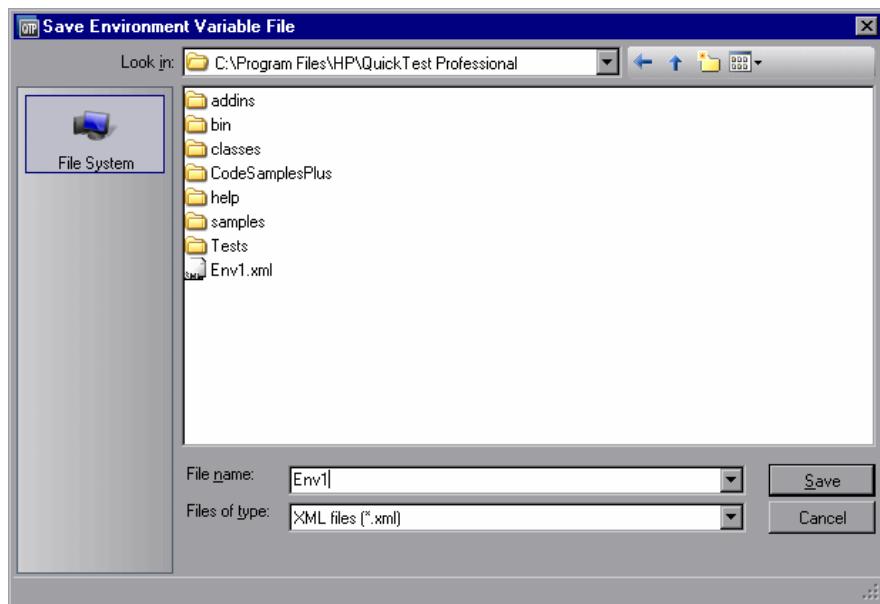
Save Recovery Scenario



Save Shared Object Repository



(Save Environment Variable File



To access:	
Prerequisite for Quality Center	Make sure you are connected to your Quality Center project. For details see, "HP ALM Connection Dialog Box" on page 1635.
Save Function Library dialog box	<ol style="list-style-type: none">1 Make sure that the function library is the active document.2 Select File > Save As.

Save Recovery Scenario dialog box	Select Resources > Recovery Scenario Manager > Save As.
Save Shared Object Repository dialog box	In the Object Repository Manager window, select File > Save As. For details on the Object Repository Manager, see "Object Repository Manager Main Window" on page 266.
Save Environment Variable File dialog box	<p>1 Select File > Settings > Environment tab > User-defined type.</p> <p>2 Create a new environment variable or open an existing environment variable file, and click Export.</p>

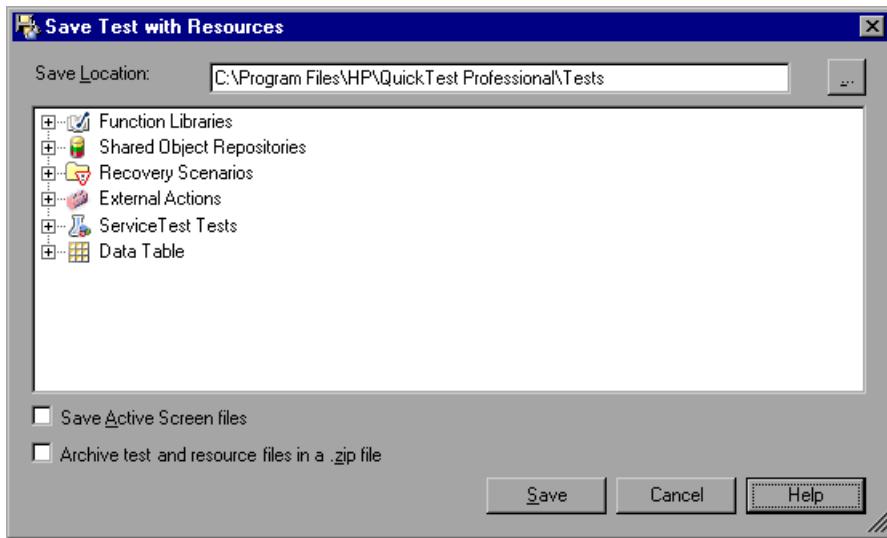
Important information	Where resources are stored: You can save a resource as an asset in the ALM/QC Test Resources module or as an attachment to a test in the ALM/QC Test Plan module. <ul style="list-style-type: none"> ➤ Saving your resource files in the ALM/QC Test Resources module enables you to take advantage of the Resources and Dependencies model. For details, see "Resources and Dependencies Model Overview" on page 1652. ➤ Saving a resource file to the ALM/QC Test Plan module, stores the file as an attachment to a test. To display a test's attachments, double-click the test file or folder.
See also	For details on a specific type of resource, see the relevant section in this guide.

User interface elements are described below:

UI Elements	Description
<sidebar>	<p>The location in which the resource file is stored, for example, File System or ALM/QC Test Resources.</p> <p>Note: You may need to select a different location before browsing to your file.</p>
Look in	<p>The path for the resource file. You can use the down arrow to navigate to the required folder, or you can double-click a folder in the resource list area to navigate to the required folder.</p>
<resource list area>	<p>The folders and/or resource files stored in the current path.</p> <p>Note: If the file is stored in a Quality Center project with version control support, you can view version control information for the file by clicking the Views down arrow and selecting Details. For more information, see "Checking Assets Out of the Version Control Database" on page 1700.</p>
File name	<p>The name of the resource file selected in the resource list area.</p>
Files of type	<p>Filters the list of displayed assets to show only the file types for the specified resource. If the current folder contains other file types, they are hidden.</p>

Save Test with Resources Dialog Box

This dialog box 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.



- | | |
|-----------|---|
| 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 the test and its resources are all updated to the current QuickTest version, as described in "Save a portable copy of a test" on page 400. ➤ 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,< external data table, shared object repository, or Service Test test) -- 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> <ul style="list-style-type: none"> ➤ 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.
See also	Conceptual overview: "Portable Copies of Tests" on page 395

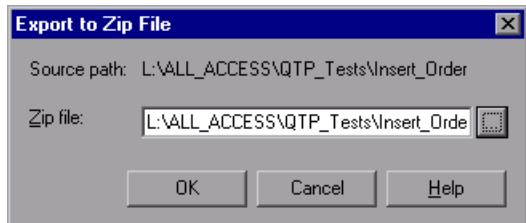
User interface elements are described below (unlabeled elements are shown in angle brackets):

UI Elements	Description
Save Location	<p>Specifies the root folder in which to save the test. By default, for any operating system prior to Windows Vista, the root folder is <QuickTest installation folder>\Tests.</p> <p>For Windows Vista and later operating systems, the root folder is: ...\\MyDocuments\\HP\\QuickTest Professional</p> <p>However, you can specify any folder on a local, network, or portable drive.</p> <p>Important: The folder you specify must not already contain a sub-folder with the same name as the test.</p>
<Resources tree>	<p>Lists the external resources that are currently associated with or attached to your test.</p> <p>Note: If an external resource is stored in the ALM/QC Test Resources module, the resources tree displays the internal name of the resource (the name assigned by Quality Center when the file was uploaded), which may be different from its actual name.</p> <p>You can view a resource's actual name and path by positioning the cursor over the branch that displays the resource's internal name.</p>

UI Elements	Description
Save Active Screen files	<p>(Relevant only for recorded tests.) Instructs QuickTest to save any existing Active Screen files with your test. Clearing the Save Active Screen files check box can be especially useful for conserving disk space if you have finished designing the test and are using the test only for test runs.</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 "How to Update Test Object Descriptions, Checkpoints, Output Values, or Active Screens" on page 1252.</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 "Export to Zip File Dialog Box" on page 426.</p>

Export to Zip File Dialog Box

This dialog box enables you to zip your QuickTest tests together with configuration, run-time, setup data, and (optionally) Active Screen files. Zipping these files together helps conserve space and makes tests easier to transfer.



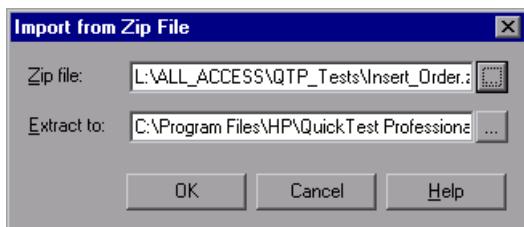
To access	<p>Use one of the following:</p> <ul style="list-style-type: none"> ▶ Select File > Export Test to Zip File to open the Export to Zip File dialog box. ▶ Select the Archive test and resource files in a .zip file check box in the Save Test with Resources dialog box (File > Save Test with Resources). For more information, see "Save Test with Resources Dialog Box" on page 422.
------------------	--

User interface elements are described below:

UI Elements	Description
Source Path	The read-only current location of the test to be zipped.
Zip File	The path location in which to store the zipped file. You can enter a zip file name and path, or accept the default name and path.

Import from Zip File Dialog Box

This dialog box enables you to extract your zipped QuickTest tests when needed. When the extraction process is complete, the test opens.



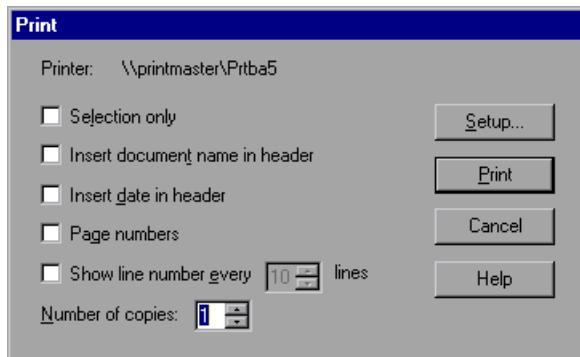
To access	Select File > Import Test from Zip File .
-----------	---

User interface elements are described below:

UI Elements	Description
Zip File	The path location of the zipped file. You can enter a zip file name and path, or accept the default name and path.
Extract to	The path location to export the test and its files. You can enter a zip file name and path, or accept the default name and path.

Print Dialog Box

This dialog box enables you to print the document displayed in the Expert View or function library to your default Windows printer. You can also include additional information in the document printout.



To access	In the Expert View or a QuickTest function library, use one of the following: <ul style="list-style-type: none"> ► Select File > Print. ► Click the Print  button.
------------------	---

User interface elements are described below:

UI Elements	Description
Printer	Displays the printer to which the document will be printed. You can print to a different printer by clicking the Setup button and modifying your printing preferences.
Selection only	Prints only the text that is currently selected (highlighted) in the Expert View or the function library.
Insert document name in header	Includes the name of the active test or function library at the top of the printout.

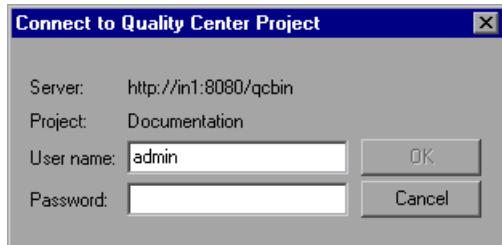
UI Elements	Description
Insert date in header	Includes today's date at the top of the document printout. The date format is taken from your Windows regional settings.
Page numbers	Includes page numbers on the bottom of the document printout (for example, page 1 of 3).
Show line numbers every __ lines	Displays line numbers to the left of a step, every specified number of lines.
Number of copies	Specifies the number of times to print the document.
Setup	Displays the standard Print Setup dialog box, in which you can specify your printing preferences.
Print	Prints the document displayed in the Expert View or function library according to the selections you make in the Print dialog box.

Troubleshooting and Limitations - Opening and Saving Testing Documents

This section describes general troubleshooting and limitations for opening and saving testing documents, as well as opening tests created in previous versions of QuickTest.

Opening tests stored in Quality Center

Opening a test from the Recent Files. If you select a test located in a Quality Center project, but QuickTest is not currently connected to that project, the Connect to Quality Center Project dialog box opens and displays the correct server, project, and the name of the user who most recently opened the test on this computer.



This dialog box also opens if you choose to open a test that was last edited on your computer using a different Quality Center user name. You can either log in using the displayed name or you can click **Cancel** to stay logged in with your current user name.

Opening a test that was created in an earlier version of QuickTest. If a test is stored in Quality Center and was created using an earlier version of QuickTest, 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**.

Opening tests stored in the file system

Opening a test that was created in earlier version of QuickTest. 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.

- QuickTest does not support hidden files. If you mark a QuickTest test, folder, or other QuickTest file as hidden (by selecting the **Hidden** attribute in the folder or file properties dialog box in Windows Explorer), QuickTest may behave unexpectedly.
- 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, the test opens 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 "QuickTest Automation Object Model Overview" on page 1590.

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**).

Opening tests created with a later version of QuickTest than the QuickTest version on your computer

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 11.00 on a computer running QuickTest 9.0.

Opening and saving tests with locked resources

If a test has locked QuickTest resource files, you can open the test in read-only mode or in read-write mode. In all cases, the locked resource file opens in read-only mode.

For conceptual information, see "Opening and Saving Tests with Locked Resources" on page 396.

For a description of the displayed message boxes, see "Open Test with Locked Resources - Message Box" on page 404 and "Save Tests with Locked Resources - Message Box" on page 416.

Opening and saving tests in Windows 7

The Libraries feature of Windows 7 is not supported. If you attempt to open a test from a library location or save a test to a library location, QuickTest may behave unexpectedly.

Workaround: Browse to your saved tests using standard file paths, even if they are included in a library, such as the Documents library. For example, to browse for a test that is stored in the default QuickTest Professional installation folder, use:

Computer > C: > Program Files > HP > HP QuickTest Professional > <TestName>

Instead of:

Libraries > Documents > HP QuickTest Professional > <TestName>

Unexpected read-only files

Assets that were saved in the file system in system folders like (%Windir% or Program files) while UAC was set to OFF can be opened only in read-only mode when the UAC is set to ON.

Workaround: Copy the asset to another location and open the files from the new location.

Multilingual support

Names and paths of tests and resources (for example, function libraries, object repositories, and recovery scenarios) are not Unicode compliant and therefore should be specified either in English or in the language of the operating system.

11

Test Creation - Keyword-Driven Methodology

This chapter includes:

Concepts

- ▶ Keyword-Driven Methodology - Overview on page 436
- ▶ Test a Flight Application Using the Keyword-Driven Methodology - Use-Case Scenario on page 446

Tasks

- ▶ How to Create a Test Using the Keyword-Driven Methodology on page 455

Concepts

Keyword-Driven Methodology - Overview

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 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, after the basic infrastructure is in place, both types of users can often do their jobs simultaneously.

Using the keyword-driven methodology, you create tests that enable users to select keywords to indicate the operations to be performed on your application. 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 listed below provides guidance on the main steps involved in creating these resources and describes where you can find detailed documentation for these steps.

- "Analyzing Your Application" on page 437
- "Setting Up Object Repositories" on page 439
- "Creating Function Libraries" on page 441
- "Configuring QuickTest According to Your Testing Needs" on page 442
- "Building Your Tests" on page 444
- "Adding Steps to Your Test Actions" on page 444
- "Running and Troubleshooting Your Tests" on page 446

Analyzing Your Application

The first step in creating a test is to analyze your application to determine your testing needs.

What development environments need to be supported by QuickTest?

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 details, see the *HP QuickTest Professional Add-ins Guide*.

What information does QuickTest need to identify objects in your application?

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. This enables QuickTest to identify objects in your application and optionally open your application at the beginning of a run session. For details, see the sections describing the Record and Run options for your testing environment in the *HP QuickTest Professional Add-ins Guide*.

What actions do you need to create?

You need to analyze the various business processes that customers perform while using your application and to create an action for each sub-process, or task, a customer might perform.

To determine which actions you need, you navigate through your application from a customer's perspective. While doing this, you perform the steps 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 Purchaseltems action, and a Logout action.

By creating separate reusable actions for each sub-process, 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 details, see "Actions Overview" on page 523.

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, as they are easier to reuse and maintain over time.



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 includes one or more of the following:

Changing the way that QuickTest identifies specific objects

This is particularly helpful when your application contains objects that change frequently or are created using dynamic content, for example, from a database. This needs to be done before you create your object repository. For details, see "Configuring Object Identification" on page 283.

Deciding how 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 actions and multiple tests. If you are new to testing, you may want to use the default (local) object repository that is unique to each action. As you feel more comfortable with the basics of test design, you may want to take advantage of shared object repositories.

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 details, see "Managing Test Objects in Object Repositories" on page 159.

Adding (learning) objects from your application

You can instruct QuickTest to learn the objects in your application according to filters that you define. For details, see "Adding and Deleting Test Objects in a Local or Shared Object Repository" on page 160.

Creating new objects with easily understandable names

You may create new objects to represent objects that do not yet exist in your application. You then update the properties and values of these objects as necessary after they exist in the application. For details, see "Define New Test Object Dialog Box" on page 200.

When you create objects, make sure that they 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.

Copying or moving objects from one repository to another

For details, see Chapter 6, "Shared Object Repositories."

Merging objects from local repositories to shared object repositories

Application testers may create objects that are saved in an action's local object repository. You can merge these objects into a shared object repository. You can also merge two or more existing shared object repositories. For details, see "Object Repository Merge Tool" on page 337.



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.

When creating a function library for the test automation infrastructure, you may do the following:

- **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 details, see "User-Defined Functions and Function Libraries" on page 1011 and "How to Manage Function Libraries" on page 1028.
- **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 details, see "How to Create and Register a User-Defined Function Using the Function Definition Generator" on page 1042.
- **Optionally define functions as new or replacement methods for test objects.** For details, see "User-Defined Function Registration" on page 1020.
- **Debug your function libraries.** For details, see "Debug a function library" on page 1030.



Configuring QuickTest According to Your Testing Needs

After you set up the test automation infrastructure, you need to configure QuickTest to use this infrastructure. This involves one or more of the following:

Defining 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 details, see "Global Testing Options Overview" on page 1420.

Creating recovery scenarios

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 details, see "Recovery Scenarios" on page 1523.

Configuring 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 pane. For details, see "QuickTest Window Layout" on page 1279.



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 do the following when creating tests and test steps:

- **Create new tests, if needed.** For details, see "Create a new test" on page 398.
- **Create the required actions.** For details, see "Analyzing Your Application" on page 437.
- **Insert calls to the relevant actions.** For example, if the first step 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 details, see "Calls to Existing Actions and Copies of Actions" on page 529.
- **Associate your object repositories with the relevant actions.** This enables you to insert steps that perform operations on those objects. For details, see "Associated Repositories Tab (Action Properties Dialog Box)" on page 567.
- **Associate your function libraries with the relevant tests.** This enables you to use your special keywords in any of the associated tests. For details, see "How to Manage Function Library Associations" on page 1036.
- **Optionally associate recovery scenarios with your test.** For details, see "How to Manage Recovery Scenario Associations" on page 1530.



Adding Steps to Your Test Actions

When your actions are ready, you can add steps to them. This involves one or more of the following:

Adding steps by selecting the keywords (operations) that represent the application functionality you want to test

For details, see "Keyword View" on page 483.

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 details, see "How to Add a Standard Step to Your Test" on page 487, "Types of Steps to Add to Your Test" on page 485, and "Generating Statements in the Expert View or in a Function Library" on page 934.

Enhancing your tests by inserting checkpoints and output values

- You can insert checkpoints to check for differences in the text strings, objects, and tables in your application. For details, see "Checkpoints Overview" on page 591.
- 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 details, see "Output Values" on page 781.

Data-driving your test

You can use the data table to data-drive your test using different data input during subsequent run sessions. This enables you to check how your application behaves during multiple iterations of the same action during a single run session. For details, see "Data Table Pane" on page 1325.

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. This enables you to increase the power and flexibility of your test. For details, see "Parameterizing Values" on page 723.



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 details, see "QuickTest Run Sessions" on page 1063.
- **After your test runs, view the run results.** Expand the nodes in the Run Results Viewer to see where steps failed and to try to understand why. For details, see "Run Results Viewer" on page 1085.
- **Troubleshoot your test so that it runs correctly.** For example, you may need to add or modify test steps. For details, see "Maintaining and Updating Tests" on page 1239.

Test a Flight Application Using the Keyword-Driven Methodology - Use-Case Scenario

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 open 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 open 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.

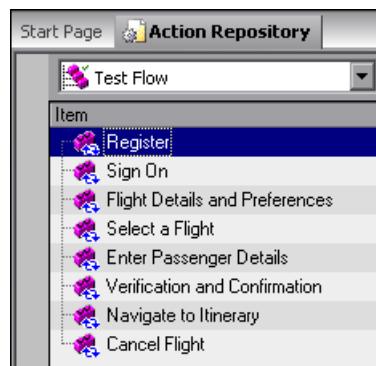
Plan and Create the Mercury Tours Test Action Repository

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 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.

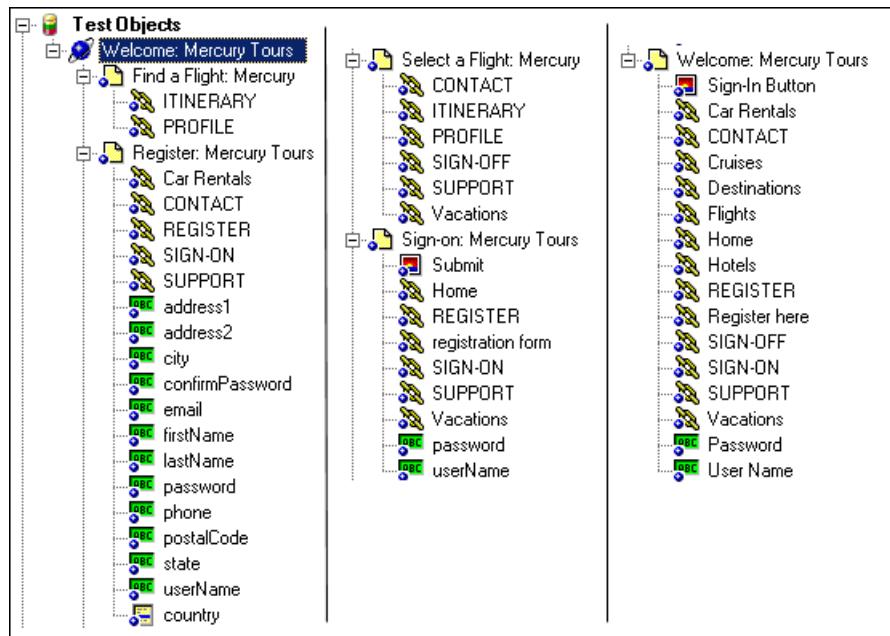
Most of the business processes we plan to test are in the central flight reservation area of the application and thus many of the same objects will be used in each of the relevant tests, but the sign on and registration processes are more standalone areas and it makes sense to store their objects separately. 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 toolbar (described on page 274), 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 toolbar, 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.

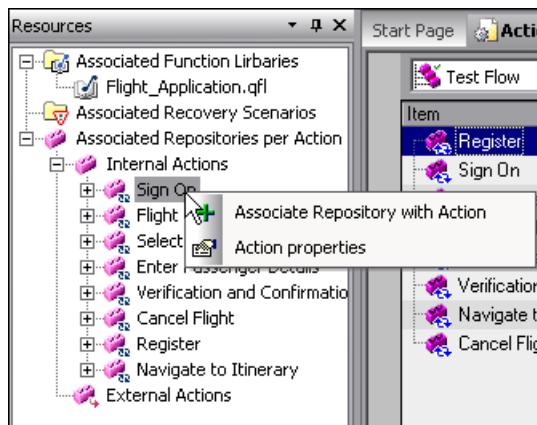
At this stage, we can create a function library containing functions such as:

- VerifyForm
- VerifyTicketPrice
- DataBaseFailureRecoveryFunction

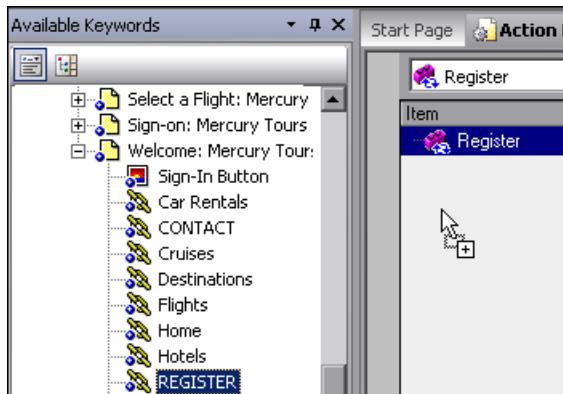
Create Tests and Test Steps for the Mercury Tours Business Processes

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.



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 447). We use the Resources pane to associate our function library with each test and then we insert calls to the relevant actions.

Tasks

How to Create a Test Using the Keyword-Driven Methodology

This task describes how to create a test using the keyword-driven methodology. For details on a specific step, see the parallel section in "Keyword-Driven Methodology - Overview" on page 436.

This task includes the following steps:

- "Analyze your application" on page 455
- "Prepare the testing infrastructure" on page 456
- "Add steps to the actions in your test action repository" on page 457
- "Enhance your test" on page 458
- "Run your test" on page 459
- "Analyze the run results and report any defects" on page 460

1 Analyze your application

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

- **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.
- **Determine the functionality that you want to test.** To do this, you 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?

- Decide how to divide these processes into smaller units that 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.

For an overview of this step, see "Analyzing Your Application" on page 437.

2 Prepare 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).
For details, see Chapter 4, "Managing Test Objects in Object Repositories".
 - **function libraries** containing functions that enhance QuickTest functionality.
For details, see Chapter 29, "User-Defined Functions and Function Libraries."
 - **recovery scenarios** that instruct QuickTest to recover from unexpected events and errors that occur in your testing environment during a run session. (Optional)
For details, see Chapter 49, "Recovery Scenarios."
 - **additional optional files**, such as data table files and environment variable files.

- **Configure QuickTest according to your testing needs.** This can include:
 - setting up your **global testing preferences**.
For details, see "Recovery Scenarios" on page 1523.
 - setting up any **test-specific preferences**.
For details, see "Individual Test Settings" on page 1455.
 - configuring your **run session preferences**.
For details, see "Run Pane (Options Dialog Box)" on page 1447 and "Run Pane (Test Settings Dialog Box)" on page 1471.
 - defining and associating **recovery scenarios**.
For details, see "Recovery Scenarios" on page 1523.
 - creating **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 details, see "QuickTest Automation Scripts" on page 1589.
- **Create one or more tests that serve as action repositories** in which you can store the actions to be used in your tests, as this enables you to maintain your actions in one central location.
- **Associate your shared object repositories with the relevant actions** (in these action repositories). 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.
For details, see "Associate Repositories Dialog Box" on page 229.

For an overview of this step, see "Setting Up Object Repositories" on page 439.

3 Add steps to the actions in your test action repository

- **(Prerequisite) Associate your function libraries and recovery scenarios with the relevant tests, so that you can insert steps using keywords.**
For details, see "Resources Pane User Interface" on page 1391.
- **Create steps using keyword-driven functionality.** You can use 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 created one or more object repositories when you prepared the testing infrastructure, as described in "Prepare the testing infrastructure" on page 456.
- **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. For details, see "Keyword View" on page 483.

For an overview of this step, see "Creating Function Libraries" on page 441.

4 Enhance 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.
For details, see "Checkpoints Overview" on page 591 and "Output Values" on page 781.
- **Replace fixed values with parameters** to broaden the scope of your test. For details, see "Parameterizing Values" on page 723.
- **Add user-defined functions** by creating function libraries and calling their functions from your test.
For details, see "User-Defined Functions and Function Libraries" on page 1011.
- **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 details, see "User Interface-Based Programming Operations" on page 887.

For an overview of this step, see:

- "Creating Function Libraries" on page 441
- "Configuring QuickTest According to Your Testing Needs" on page 442
- "Building Your Tests" on page 444
- "Adding Steps to Your Test Actions" on page 444

5 Run your test

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

- **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 on. 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 details, see "QuickTest Run Sessions" on page 1063.

➤ **Run your test to debug it. (Optional)**

You can:

- Control your run session to help you identify and eliminate defects in your test.
- Use the **Step Into**, **Step Over**, and **Step Out** commands to run your test step by step.
- Begin your run session from a specific step in your test, or run the test until a specific step is reached.
- Set breakpoints to pause your test at predetermined points.
- View or change the value of variables in your test each time it stops at a breakpoint in the Debug Viewer.
- Manually run VBScript commands in the Debug Viewer.

For details, see "Debugging Tests and Function Libraries" on page 1205.

- **Run your test using Maintenance Run Mode after your application changes. (Optional)**

Use **Maintenance Run Mode** to update your test 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. For details, see "Maintaining and Updating Tests" on page 1239.

- **Run your test using Update Run Mode to update one or more of the following: (Optional)**

- property sets used for test object descriptions
- expected checkpoint values
- data available to retrieve in output values
- Active Screen images and values

For an overview of this step, see "Running and Troubleshooting Your Tests" on page 446.

6 Analyze the run results and report any defects

After you run your test, you can:

- **View the results of the run in the Run Results Viewer.** 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 Run Results Viewer.
For details, see "Run Results Viewer" on page 1085.
 - If you enabled local system monitoring for your test, you can view the results in the System Monitor tab of the Run Results Viewer.
For details, see "System Monitor Pane (Run Results Viewer)" on page 1137.

► **Report defects detected during a run session. (Optional)**

If you have access to Quality Center, the HP centralized quality solution, you can report the defects you discover to the project database in either of the following ways:

- Instruct QuickTest to automatically report each failed step in your test.
- Report failed steps manually from the Run Results Viewer.

For details, see "Quality Center Integration" on page 1605.

12

Test Creation - Recording Mechanism

This chapter includes:

Concepts

- Recording Tests - Overview on page 464
- Recording Modes on page 467

Tasks

- How to Record a Test Using Normal Recording Mode on page 470
- How to Record a Test Using Analog Recording on page 473
- How to Record a Test Using Low Level Recording on page 474

Reference

- Analog Recording Settings Dialog Box on page 476
- Record and Run Settings Dialog Box on page 479

Troubleshooting and Limitations - Recording Tests on page 481

Concepts

Recording Tests - Overview

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

As you record the test, 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.

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.

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 14, "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 "Analog Recording" on page 467 and "Low Level Recording" on page 468.

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 "Test Creation - Keyword-Driven Methodology" on page 435.

Before you begin recording, you need to ensure that your tests cover your testing requirements. For more information on planning your tests, see "Test Creation Overview" on page 383.

This section also includes:

- "Guidelines for Recording Tests" on page 465
- "Tips for Recording Tests" on page 466

Guidelines for Recording Tests

- 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 723.

- Consider using actions to streamline the testing process. For more information, see "Actions Overview" on page 523.
- 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.
- 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*.

Tips for Recording Tests

- If you are creating a test on Web objects, you can record your test on Microsoft Internet Explorer or Mozilla Firefox and run it on another supported browser (according to the guidelines specified in the *HP QuickTest Professional Product Availability Matrix*, available from the Documentation Library Home page or the root folder of the QuickTest Professional DVD). 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*.
- 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 "How to Define Virtual Objects for Unsupported Objects in Your Test" on page 1516.
- After you create your test, you can enhance it using checkpoints and other special testing options. After creating your initial test, you can further enhance it by adding and modifying steps in the Keyword View or Expert View.

Recording Modes

QuickTest provides the following recording modes:

- **Normal Recording.** 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. For more information, see "How to Record a Test Using Normal Recording Mode" on page 470.

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

Analog Recording or Low Level Recording.

- **Analog Recording.** Enables you to record the exact mouse and keyboard operations relative to the screen or the application window. This method records mouse movements.
- **Low Level Recording.** Enables you to record the exact coordinates of any object, whether or not QuickTest recognizes the specific object or the specific operation. This method does not record mouse movements.

Analog Recording

This method 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.

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. For more information, see "How to Record a Test Using Analog Recording" on page 473.

Considerations for Analog Recording

- You can record in **Analog Recording** mode relative to the screen or relative to a specific window. For more information, see "Analog Recording Settings Dialog Box" on page 476
- 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.
- You cannot edit **Analog Recording** steps from within QuickTest.

Low Level Recording

This method 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.

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. For more information, see "How to Record a Test Using Low Level Recording" on page 474.

Considerations for 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.)
- Low-level recording mode is not fully supported for multibyte character input.
- Steps recorded using **Low Level Recording** mode may not run correctly on all objects.

Tasks

How to Record a Test Using Normal Recording Mode

This task describes how to create a test using normal recording mode.

This task includes the following steps:

- "General Prerequisites" on page 470
- "Web-based Applications Prerequisites" on page 471
- "Record the test" on page 471
- "Stop and save the test" on page 472

1 General Prerequisites

Close unnecessary applications. Before you start to record, close all applications not required for the recording session.

Determine application access. 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*.

Set global record and run options. Choose how you want QuickTest to record and run your test by setting global testing options in the Options dialog box and settings specific to your test in the Test Settings dialog box. For more information, see Chapter 45, "Global Testing Options" and Chapter 46, "Individual Test Settings."

2 Web-based Applications Prerequisites

Determine Browser activation timing. QuickTest can record only on Web browsers that were opened after QuickTest. If you plan to use the Record and run test on any open browser option in the Record and Run Settings dialog box, make sure that the browser you want to use for recording was opened after you opened QuickTest.

Set security zone. 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.

Modify object values. If you are recording on a Web object, you must modify the object's value to enable 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).

3 Record the test

- a Create a new test, or open an existing test, by doing one of the following:
 - 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 "How to Perform File Operations on Test Files" on page 398.

- b Click the **Record** button or select **Automation > Record**.

- If the Record and Run Settings dialog box opens, set the required settings. The settings in this dialog box can determine whether to open an application for you at the beginning of record and run settings. It can also control on which applications QuickTest records. For example, for performance reasons, the default setting in the Windows Applications tab is to record and run only on the applications you specify (and not on any open application). If you do not specify an application or change this option, QuickTest will not record or run on any Windows-based application.

If the dialog box does not open automatically (for example, you are recording additional steps in an existing test or you manually opened and set options in this dialog box before beginning the record session), but you want to view or modify the settings, you can open it using the **Automation > Record and Run Settings dialog box menu** command.

For more information, see "Record and Run Settings Dialog Box" on page 479.

- Navigate through your application. QuickTest records each step you perform and displays it in the Keyword View and Expert View.
 - You can insert text checkpoints, object checkpoints, and bitmap checkpoints to determine if your application is functioning correctly. For more information, see Chapter 15, "Checkpoints Overview."
 - 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 22, "Parameterizing Values."

4 Stop and save the test



- 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 "Run Pane (Options Dialog Box)" on page 1447.)



- 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 "Save Test Dialog Box" on page 412.

How to Record a Test Using Analog Recording

This task describes how to record a test using analog recording, in which your keyboard input, mouse movements, and clicks are recorded and saved in an external data file.

When QuickTest runs the test, this external data file is called, and every movement, mouse click, and operation are replicated exactly as you recorded them.

To record in Analog Recording mode:

- 1 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. For more information, see "Analog Recording Settings Dialog Box" on page 476.
- 3 Start recording, and perform the operations you want to record in **Analog Recording** mode.
- 4  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.

Example

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 14, "Actions".

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

To stop an analog step in the middle of a run session, press **CTRL + ESC**, then click **Stop** in the Testing toolbar.

How to Record a Test Using Low Level Recording

This task describes how to record a test using low level recording, in which 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.

To record in Low Level Recording mode:

- 1 Click the **Record** button to begin a recording session.
- 2 Click the **Low Level Recording** button or select **Automation > Low Level Recording**.





- 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.

Example

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:

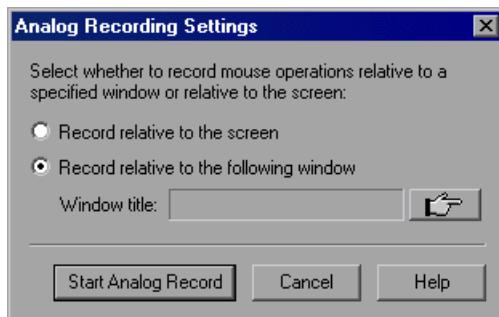
▼ Microsoft Internet Explorer			
└─ Internet Explorer_Server	Click	564,263	Click the "Internet Explorer_Server" object.
└─ Internet Explorer_Server	Type	"tutorial"	Type "tutorial" in the "Internet Explorer_Server" object.
└─ Internet Explorer_Server	Type	micTab	Type micTab in the "Internet Explorer_Server" 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
```

Reference

Analog Recording Settings Dialog Box

This dialog box enables you to select whether QuickTest records mouse movements or keyboard input relative to the coordinates of your screen, or relative to the coordinates of the specified window.



To access	<p>Click the Record button to begin a recording session, and do one of the following:</p> <ul style="list-style-type: none">➤ Click the Analog Recording button .➤ Select Automation > Analog Recording.
------------------	---

Important information	<ul style="list-style-type: none"> ➤ You can switch to Analog Recording mode only while recording. The option is not available while editing. ➤ When you record in Analog Recording mode relative to the screen, the run session will fail if you change the screen resolution or the screen location after recording. ➤ 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. ➤ 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.
Relevant tasks	"How to Record a Test Using Analog Recording" on page 473
See also	"How to Record a Test Using Low Level Recording" on page 474

User interface elements are described below:

UI Elements	Description
	Pointing Hand. Click this button to select the specific area or window of the application to record. For information on using the pointing hand, see "Tips for Using the Pointing Hand" on page 157.
Window title	Displays the name of the window to record on in Analog Recording mode.

UI Elements	Description
Record relative to the screen	<p>QuickTest records any mouse movement or keyboard input relative to the coordinates of your screen, and disregards any open applications and any applications specified in the Record and Run Settings dialog box.</p> <p>Select this option 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.</p>
Record relative to the following window	<p>QuickTest records any mouse movement or keyboard input relative to the coordinates of the specified window.</p> <p>Select this option if all your operations are performed on objects within the same window and that window does not move during analog recording. This helps ensure 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.</p>

Record and Run Settings Dialog Box

This dialog box enables you to set your record and run settings before you begin a record or run session.



To access	<p>Select Automation > Record and Run Settings.</p> <p>Note: This dialog box opens automatically only before recording a new test, and does not open the next time you record in that test.</p>
Important information	<ul style="list-style-type: none">➤ You can open this dialog box at any time, for example:<ul style="list-style-type: none">➤ If you have already recorded one or more steps in the test, you can modify the settings before you continue recording.➤ If you want to run the test on a different application than the one you previously used, you can select a different application to test.➤ The tabs available in the Record and Run Settings dialog box depend on the loaded add-ins.➤ For details on the tab to use and the options available for the environment you are testing, see the relevant chapter in the <i>HP QuickTest Professional Add-ins Guide</i>.
Relevant tasks	<p>"How to Record a Test Using Normal Recording Mode" on page 470</p>
See also	<p>"Recording Modes" on page 467</p>

Troubleshooting and Limitations - Recording Tests

This section describes troubleshooting and limitations for recording tests.

- For troubleshooting and limitations for learning objects, and recording and running steps, see "Learning objects, and recording and running steps" on page 1083.
- **Visual relation identifier.** QuickTest does not record the visual relation identifier property when recording steps. This property can be added only manually from the Object Properties dialog box or the Object Repository Manager or window. For details, see "Visual Relation Identifiers" on page 174.
- **Start menu.** On Windows Vista, Windows 7, Windows Server 2008, and Windows Server 2008 R2, if you begin a recording session after installing QuickTest (or upgrading from an earlier version) without restarting your computer, QuickTest cannot record operations on the Windows **Start** menu or **Quick Launch Panel**.

Workaround: Restart your computer and start a new recording session.

- **Start menu.** QuickTest does not record launching Windows Help from the Start menu.
- **Start menu.** QuickTest does not record the selection of **Start** menu items that are customized as menus, such as My Computer, Control Panel, or Recent Documents.

Workaround: Customize the **Start** menu items as links so that QuickTest can record operations on them or record the activation of these items another way (and not through the **Start** menu).

- **Dual monitor support.** QuickTest records only on the primary monitor. Therefore, if you are working with dual monitors, make sure that your application is visible on the primary monitor during a recording session.

13

Keyword View

This chapter includes:

Concepts

- ▶ Keyword View Overview on page 484
- ▶ Comments in the Keyword View on page 486
- ▶ Conditional and Loop Statements in the Keyword View on page 486
- ▶ Standard Steps After a Conditional or Loop Block on page 487

Tasks

- ▶ How to Add a Standard Step to Your Test on page 487
- ▶ How to Add a Standard Step After a Conditional or Loop Block on page 493
- ▶ How to Move an Action or Step on page 494
- ▶ How to Delete a Step on page 496
- ▶ How to Navigate in the Keyword View and Other Tips on page 497
- ▶ How to Insert and Remove Breakpoints in the Keyword View on page 501
- ▶ How to View Properties of Step Elements in the Keyword View on page 502

Reference

- ▶ Keyboard Shortcuts in the Keyword View on page 503
- ▶ Keyword View User Interface on page 504
- ▶ Columns Tab (Keyword View Options Dialog Box) on page 508
- ▶ Fonts and Colors Tab (Keyword View Options Dialog Box) on page 511

► [Select Test Object Dialog Box on page 513](#)

► [Password Encoder Tool on page 519](#)

[Troubleshooting and Limitations - Keyword View on page 520](#)

Concepts

Keyword View Overview

The Keyword View enables you to create and view the steps of your test in a modular, table-like format. Each step is displayed as a row in the Keyword View, and is comprised of individual, modifiable parts. You can 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.



Types of Steps to Add to Your Test

You can insert the following types of steps using the relevant options from the **Insert** menu.

- A **standard** statement step. For details, see "How to Add a Standard Step to Your Test" on page 487.
- A **checkpoint** step. For details, see "Checkpoints Overview" on page 591.
- An **output value** step. For details, see "Output Values" on page 781.
- **Comments** in steps to separate parts of an action or a test and to add details about a specific part. For details, see "Insert Comment Dialog Box" on page 912.
- A step that uses **programming logic** to:
 - send information to the run results
 - put a comment line in your test
 - synchronize your test with your application
 - measure a transaction in your test

For details, see "User Interface-Based Programming Operations" on page 887.

- Steps containing **conditional statements** and **loop statements**. For details, see "Conditional and Loop Statements in the Keyword View" on page 486.

For information on adding a new step immediately after a conditional or loop statement block, see "Standard Steps After a Conditional or Loop Block" on page 487.

Comments in the Keyword View

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 action includes.

After you add a comment, it is always visible in the Keyword View as long as one or more columns are displayed. For information on selecting columns to display, see "Columns Tab (Keyword View Options Dialog Box)" on page 508.

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 26, "User Interface-Based Programming Operations."

Standard Steps 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.

When you finish 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 in "How to Add a Standard Step After a Conditional or Loop Block" on page 493. For details on conditional and loop statements, see Chapter 26, "User Interface-Based Programming Operations."

Tasks

How to Add a Standard Step to Your Test

This task includes the following steps:

- "Insert a new step" on page 487
- "Define the step" on page 488

1 Insert a new step

Do 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 shortcut 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 of an existing step, according to the QuickTest object hierarchy, as described in "QuickTest Test Object Hierarchy" on page 133.

2 Define the step

Click in the cell for the part of the step you want to modify and specify 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** (described on page 505) - For task details, see "How to Select an Item for Your Step" on page 489.
- **Operation** (described on page 506) - For task details, see "How to Select an Operation for Your Step" on page 490.
- **Value** (described on page 506) - For task details, see "How to Define Values for Your Step Arguments" on page 491.
- **Assignment** (described on page 506) - For task details, see "How to Navigate in the Keyword View and Other Tips" on page 497.
- **Comment** (described on page 506) - For task details, see "How to Navigate in the Keyword View and Other Tips" on page 497.

Tip: You can use the standard editing commands (**Cut**, **Copy**, **Paste**, and **Delete**) in the **Edit** menu or in the shortcut 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 details, see "Keyboard Shortcuts in the Keyword View" on page 503.

 **How to Select an Item for Your Step**

This task describes how to select an item for your step and includes the following steps:

- "Select an item from the displayed list" on page 489
- "Drag and drop an item from the Available Keywords pane" on page 489
- "Select a test object from an associated object repository or from your application" on page 489

Do one of the following:

Select an item from the displayed list

- 1 Click in the **Item** cell.
- 2 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.

Drag and drop an item from the Available Keywords pane

Drag and drop an object from the Available Keywords pane to your test. For details, see "Available Keywords Pane Overview" on page 1320.

Select a test object from an associated object repository or from your application

- 1 Click in the **Item** cell to activate it.
- 2 Click the down arrow and select **Object from repository** to open the Select Test Object dialog box. The test objects available in the list are the sibling and child test objects of the previous step's test object. 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.

For details, see "Select Test Object Dialog Box" on page 513.

How to Select an Operation for Your Step

This task describes how to specify the operation to be performed on the item listed in the **Item** column.

This task includes the following steps:

- "Prerequisite" on page 490
- "Select an operation" on page 490

1 Prerequisite

Select an item in the **Item** cell. For details, see "How to Select an Item for Your Step" on page 489.

2 Select an operation

- a Click in the **Operation** cell to activate it.
- b Click the down arrow 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: When you position the cursor over an operation in the list, a tooltip describes what this operation performs. For user-defined functions, the tooltip is taken from the description that you provided in the associated function library. For details, see "Add documentation details to the function - optional" on page 1045.



How to Define Values for Your Step Arguments

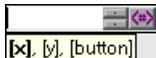
This task describes how to specify the arguments for the operation to be performed on the item listed in the **Item** column.

This task includes the following steps:

- "Define or modify a value" on page 491
- "Add multi-line arguments" on page 492
- "Parameterize the value for an argument" on page 492
- "Encode a password" on page 492

Define or modify a value

- 1 Click in the **Value** cell to activate it.
- 2 Click in each partition of the **Value** cell and enter the argument values for the selected operation. As 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.



You can enter **constant** or **parameterized** values. For details on parameterizing values, see "Parameterize the value for an argument" on page 492.

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.

Add multi-line arguments

In a **Value** cell, 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.

"Readability:good..."	Enter "Readability:good..." in the "positive" edit box.
-----------------------	---

Note:

- You can select the cell to display the entire argument value to be used in the step.
 - 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.
-

Parameterize the value for an argument

Click the  button in the required **Value** cell. The Value Configuration Options dialog box opens. For details, see "Value Configuration Options Dialog Box" on page 872.

Encode a password

You can encode passwords for use in method arguments and data table cells. For details, see "Password Encoder Tool" on page 519.

How to Add a Standard Step After a Conditional or Loop Block

This task includes the following steps:

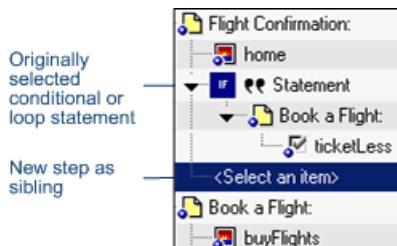
- "Select the end of the conditional or loop statement" on page 493
- "Insert a new step using the New Step After Block option" on page 493
- "Define the step" on page 493

1 Select the end of the conditional or loop statement

Select the conditional or loop statement step after and outside of which you want to add the new step.

2 Insert a new step using the New Step After Block option

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).



3 Define the step

Specify the content of the step by modifying it, as described in "How to Add a Standard Step to Your Test" on page 487.

How to Move an Action or Step

This task describes how to move an action to a different location within a test, as needed, and how to move a step to a different location within an action.

This task includes the following steps:

- "Drag and drop a step" on page 494
- "Use the Action toolbar to move a top-level action" on page 494
- "Copy/cut and paste a step" on page 494

Drag and drop a step

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.

Use the Action toolbar to move a top-level action

In the Action toolbar, display the Test Flow pane. Then drag the action up or down to the required location.

Copy/cut and paste a step

Copy or cut the step to the Clipboard and then paste it in the required location. When you move, copy, or cut an action or step, you also move, copy, or cut all of its sub-steps, if any.

- **Copy.** Use **Edit > Copy** or **CTRL + C**.
- **Cut.** Use **Edit > Cut** or **CTRL + X**.
- **Paste.** Use **Edit > Paste** or **CTRL + V**.

Notes:

- **Conditional and loop blocks.** These 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.
 - **Parent and child objects.** 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.
 - **Copying an action.** 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 box opens, which enables you to insert a call to a copy of an action. For details, see "Calls to Existing Actions and Copies of Actions" on page 529.
-

How to Delete a Step

Note: You cannot delete a step if one of its cells is in edit mode.

1 Prerequisites

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

Item	Operation	Value
Action1		
Welcome: Mercury	Navigate	"http://newtours.mercuryinteractive.com"
Welcome: Mercury		
userN	Set	"nicole"
password	SetSecure	"3ee357f628811830704e"
Sign-In	Click	21.2

2 Delete the step.

- a Select the step that you want to delete.
- b Select **Edit > Delete**, press the **DELETE** key, or right-click the step and select **Delete** from the shortcut menu. A confirmation message opens.

Note: If you are deleting a reusable action that is called by another action, and if your test is stored in Quality Center and is using the resources and dependencies model (described on page 1652), a list of the actions that call the action that you are deleting is displayed in the confirmation message box.

To copy any or all of the actions from the list to the Windows Clipboard:

- 1 Select the relevant actions from the list.
 - 2 Right-click and select **Copy Selected**, or press CTRL+C on your keyboard.
-

- c Click **Delete Step** to confirm. The step is deleted from the action.

How to Navigate in the Keyword View and Other Tips

This task includes the following steps:

- "General Tips" on page 498
 - "Display or Hide Columns" on page 499
 - "Item Column" on page 499
 - "Value Column" on page 499
 - "Assignment Column" on page 500
 - "Comment Column" on page 500
 - "Documentation Column" on page 500
-

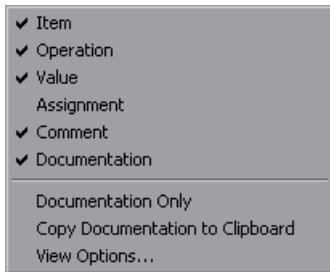
Note: For a list of menu shortcuts for the tips described in this section, see "QuickTest Commands" on page 82.

General Tips

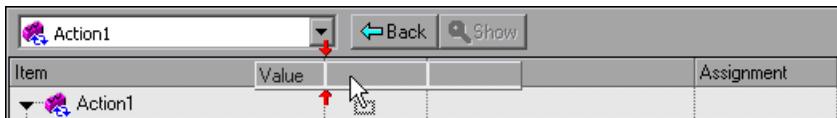
- Use the left and right arrow keys to move the focus one cell to the left or right, with the following exceptions:
 - In the **Item** column, the left and right arrow keys collapse or expand the item (if possible). If not possible, the arrow keys behave as in any other column.
 - When a cell is in edit mode (for example, when modifying a value or comment), the left and right arrow keys move the cursor within the edited cell.
- Use the standard editing commands (**Cut**, **Copy**, **Paste**, and **Delete**) in the Edit menu or in the context menu to define or modify your steps. For details, see "Keyboard Shortcuts in the Keyword View" on page 503.
- Drag and drop steps to move them to a different location within your action. For details, see "How to Navigate in the Keyword View and Other Tips" on page 497.

Display or Hide Columns

- Use the Keyword View Options dialog box. For details, see "Columns Tab (Keyword View Options Dialog Box)" on page 508.
- Right-click a column header row in the Keyword View. Then select or clear the required column name from the shortcut menu.



- 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.



Item Column

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.

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.

Value Column

When a **Value** cell is selected, press CTRL+F11 to open the Value Configuration Options dialog box. For details, see "Value Configuration Options Dialog Box" on page 872.

Assignment Column

- To create or edit an assignment to or from a variable, double-click in the left part of the **Assignment** cell.
- 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 the arrow button.
- To specify or modify the name of the variable, click in the right part of the **Assignment** cell.

Comment Column

- 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 details, see "Insert Comment Dialog Box" on page 912.

- To modify an existing comment, double-click the comment in the **Comment** column. The cell becomes a free text field.

Documentation Column

- To display only the **Documentation** column of a test, right-click the column header row and choose **Documentation Only** from the displayed menu.
- To copy the documentation, do one of the following:
 - Select **Edit > Copy Documentation to Clipboard**.
 - Right-click the column header row and choose **Copy Documentation to Clipboard** from the displayed menu.

Paste the documentation into a different application, as required.

How to Insert and Remove Breakpoints in the Keyword View

This task describes how to 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. For details on breakpoints, see "How to Use Breakpoints" on page 1219.

Insert a breakpoint in the Keyword View

Do any of the following:

- 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.

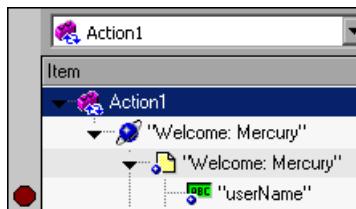
Remove a breakpoint from the Keyword View

Do any of the following:

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

Example

QuickTest automatically places the breakpoint next to the appropriate item for the step. In the example shown below, 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.



How to View Properties of Step Elements in the Keyword View

To view properties for a part of a step: Right-click the item whose properties you want to view, and select the relevant option from the context menu.

For example, you can view object properties, action properties, action call properties, checkpoint properties, and output value properties.

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.

Reference

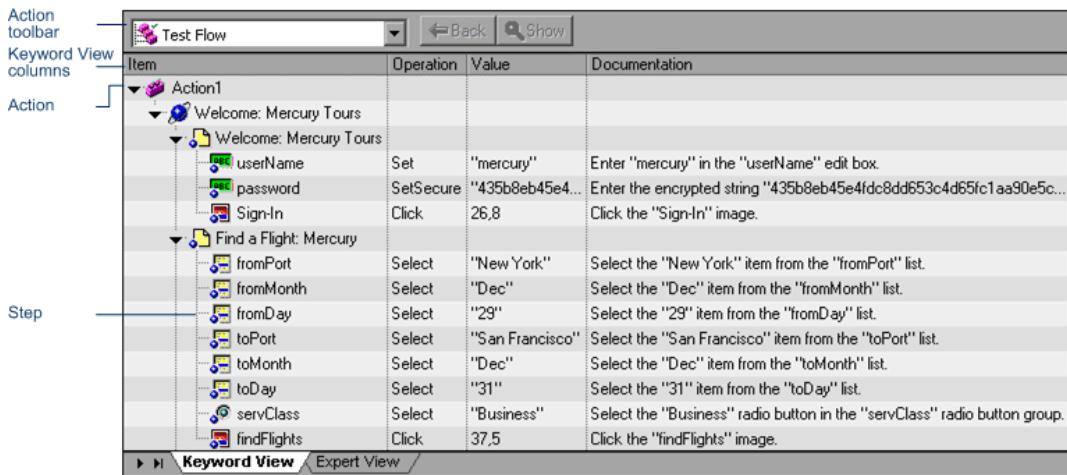
Keyboard Shortcuts in the Keyword View

If you prefer to use your keyboard to edit a test, you can use the following keyboard shortcuts 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.
- Use the TAB and SHIFT+TAB keys to 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.
 - 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.
 - 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.

Keyword View User Interface

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 details, see "Columns Tab (Keyword View Options Dialog Box)" on page 508.

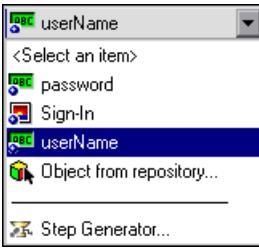


The screenshot shows the Keyword View window titled "Test Flow". It displays a hierarchical tree of test steps under "Action1". The tree includes "Welcome: Mercury Tours" and "Find a Flight: Mercury". Under "Welcome: Mercury Tours", there are three steps: "userNmae", "password", and "Sign-In". Under "Find a Flight: Mercury", there are six steps: "fromPort", "fromMonth", "fromDay", "toPort", "toMonth", "toDay", "servClass", and "findFlights". The "Action" annotation points to the tree structure, and the "Step" annotation points to the individual steps listed under "Find a Flight: Mercury". The table has columns for "Item", "Operation", "Value", and "Documentation".

Action toolbar	 Test Flow		
Keyword View columns			
Action	Action1	Operation	Value
	Welcome: Mercury Tours		
	>Welcome: Mercury Tours		
	>Welcome: Mercury Tours		
	userNmae	Set	"mercury"
	password	SetSecure	"435b8eb45e4fd..."
	Sign-In	Click	26,8
	Find a Flight: Mercury		
	fromPort	Select	"New York"
	fromMonth	Select	"Dec"
	fromDay	Select	"29"
	toPort	Select	"San Francisco"
	toMonth	Select	"Dec"
	toDay	Select	"31"
	servClass	Select	"Business"
	findFlights	Click	37,5
Step		Documentation	
			Enter "mercury" in the "userNmae" edit box.
			Enter the encrypted string "435b8eb45e4fd...".
			Click the "Sign-In" image.
			Select the "New York" item from the "fromPort" list.
			Select the "Dec" item from the "fromMonth" list.
			Select the "29" item from the "fromDay" list.
			Select the "San Francisco" item from the "toPort" list.
			Select the "Dec" item from the "toMonth" list.
			Select the "31" item from the "toDay" list.
			Select the "Business" radio button in the "servClass" radio button group.
			Click the "findFlights" image.

To access	<p>Use one of the following:</p> <ul style="list-style-type: none"> ► In QuickTest, display a test and click the Keyword View tab. ► Select View > Keyword View if the Expert View is displayed. <p>Note: If the Keyword View tab is not displayed, select Tools > Options and, in the General pane, select the Display the Keyword View for tests and scripted components command.</p>
-----------	---

User interface elements are described below:

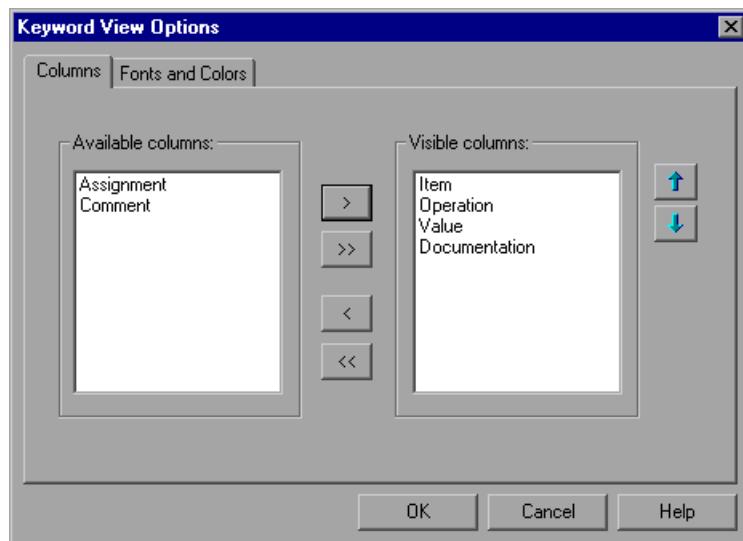
UI Elements	Description
Item	<p>The item on which you want to perform the step. An item can be any of the following:</p> <ul style="list-style-type: none"> ➤ a test object from the object repository ➤ a utility object ➤ a function call ➤ a statement, for example, a Dim statement ➤ a step generated by the Step Generator. For details, see "How to Insert Steps Using the Step Generator" on page 900. <p>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).</p> <p>Example of Item list</p> <p>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 object repository. The example below shows the objects available for the step following a userName test object.</p>  <pre> graph TD Root(userName) --> password Root(userName) --> SignIn Root(userName) --> userName Root(userName) --> ObjectFromRepository[Object from repository...] Root(userName) --> StepGenerator[Step Generator...] </pre> <p>For task details, see "How to Select an Item for Your Step" on page 489.</p>

UI Elements	Description
Operation	<p>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 Operation column displays the default operation for the item, by default. For task details, see "How to Select an Operation for Your Step" on page 490.</p>
Value	<p>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.</p> <p>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.</p> <p>For task details, see "How to Define Values for Your Step Arguments" on page 491.</p>
Assignment	<p>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.</p> <p>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.</p> <ul style="list-style-type: none"> ➤ Store in X. Value is equivalent to an X = <step> line in the Expert View. ➤ Get From X. Value is equivalent to a <step> = X line in the Expert View. For details on storing variables, see "Storage Location Options Dialog Box" on page 924.
Comment	<p>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.</p> <p>Note: QuickTest does not process comments when it runs a test.</p>

UI Elements	Description
Documentation	<p>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.</p> <p>Tips:</p> <ul style="list-style-type: none">➤ Displaying only this column (and hiding the other columns) is useful if you want to print or view manual testing instructions. For details, see "How to Navigate in the Keyword View and Other Tips" on page 497.➤ 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 details, see "Add documentation details to the function - optional" on page 1045 and "User-Defined Functions and Function Libraries" on page 1011.

Columns Tab (Keyword View Options Dialog Box)

This tab enables you to specify the columns you want to display in the Keyword View. You can also specify the order in which the columns are displayed.



To access	From the Keyword View: Tools menu > View Options item > Columns tab
Important Information	<ul style="list-style-type: none"> ➤ You can use standard SHIFT and CTRL key behaviors to select multiple column names in the Available columns or Visible columns list. ➤ As an alternative to using this dialog box tab, you can: <ul style="list-style-type: none"> ➤ Select which Keyword View columns to display by right-clicking the column header row in the Keyword View and selecting the columns to display or hide. ➤ Select the Documentation Only item in the displayed menu. This is useful if you want to print the documentation column information for use as instructions for manual testing. For details on printing from the Keyword View, see "Print Dialog Box" on page 428.
See also	<ul style="list-style-type: none"> ➤ "Keyword View User Interface" on page 504 ➤ "Fonts and Colors Tab (Keyword View Options Dialog Box)" on page 511

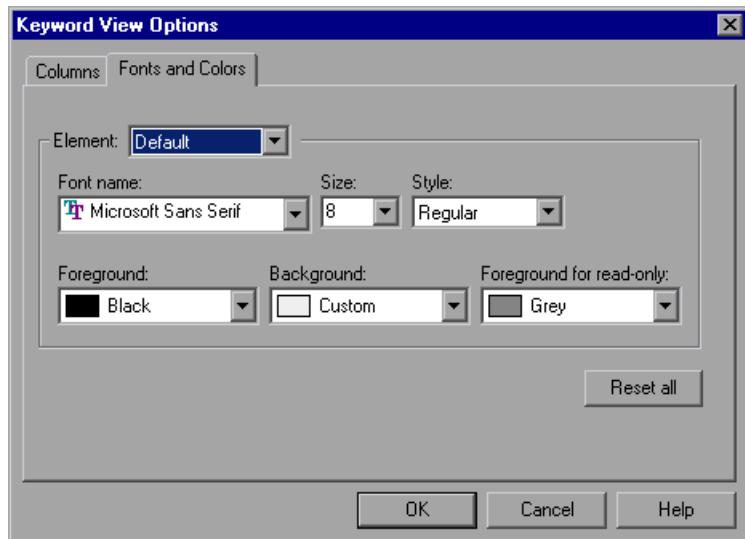
User interface elements are described below:

UI Elements	Description
Available columns	Columns not currently displayed in the Keyword View.
Visible columns	Columns currently displayed in the Keyword View.
>	Display Selected Columns. Moves the selected columns from the Available columns list to the Visible columns list. Alternatively, you can double-click an item to move it to the other list.
>>	Display All Columns. Moves all columns from the Available columns list to the Visible columns list.

UI Elements	Description
>	Hide Selected Columns. Moves the selected columns from the Visible columns list to the Available columns list. Alternatively, you can double-click an item to move it to the other list.
>>	Hide All Columns. Moves all columns from the Visible columns list to the Available columns list.
	Move Up / Move Down. Adjusts the order in which the selected columns appear in the Keyword View. 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 must still select the item first. Only then is the Operation column list refreshed to match the selection you made in the Item column.

Fonts and Colors Tab (Keyword View Options Dialog Box)

This tab enables you to specify text and color display options for different elements in the Keyword View.



To access	From the Keyword View, select Tools > View Options > Columns tab.
See also	<ul style="list-style-type: none">▶ "Keyword View User Interface" on page 504▶ "Columns Tab (Keyword View Options Dialog Box)" on page 508

User interface elements are described below:

UI Elements	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>

UI Elements	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.

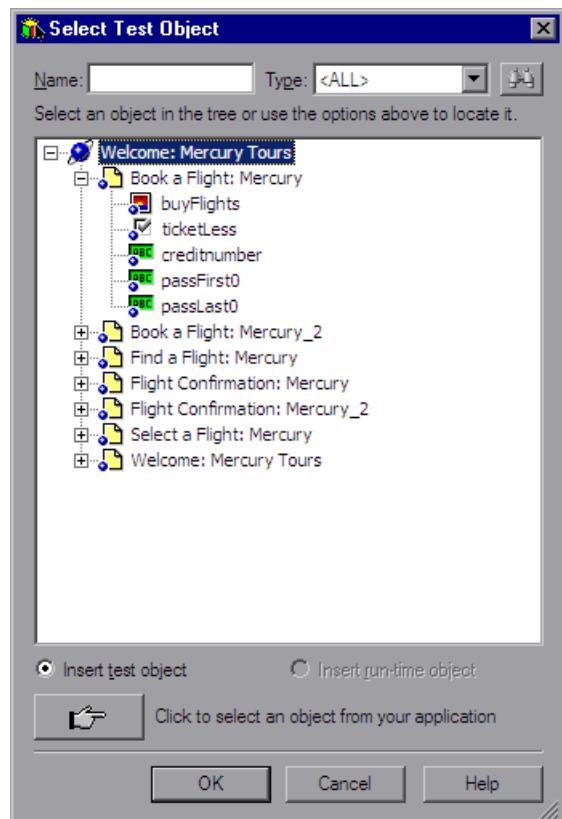


Select Test Object Dialog Box

This dialog box displays the object repository tree and enables you to:

- Select an object in the object repository tree for your step.
- Select an object from your application for your step. This adds a test object to the local object repository.
- Enter a **.Object** statement for the selected test object in your test.
- Select a related object to use in a visual relation identifier. This adds a test object to the same object repository as the test object you want to identify. For details on visual relation identifiers, see "Visual Relation Identifiers" on page 174.

The following image shows an example of this dialog box when opened from the Keyword View or from the Visual Relation Identifier dialog box:



Insert test object

Insert run-time object



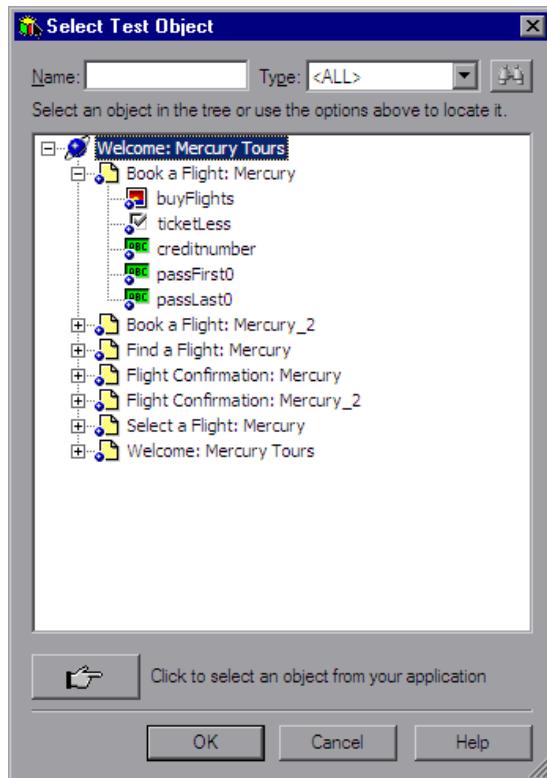
Click to select an object from your application

OK

Cancel

Help

The following image shows an example of this dialog box when opened from the Step Generator:



To access	<p>Use one of the following:</p> <ul style="list-style-type: none">► In the Item cell of the Keyword View, click the arrow button to display the Item list and then select Object from repository.► In the Step Generator dialog box, select Test Objects from the Category box list and click the Select Object button .► In the Visual Relation Identifier dialog box, click the Add button.
------------------	--

Important information	<ul style="list-style-type: none"> ➤ You can select any object in the object repository tree for your new step. For more information on the object repository, see Chapter 4, "Managing Test Objects 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 by using the pointing hand. ➤ If you select an object in your application that is not in an associated 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 <code>Edit</code> (that is used for entering a username) to <code>UserName</code>. 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 associated shared object repository and make the new test objects available to other tests. For details, see "When to Update a Shared Object Repository from Local Object Repositories" on page 339. ➤ 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 <code>Login > Confirm</code>. For details, see "Renaming Test Objects" on page 170.
Relevant tasks	<ul style="list-style-type: none"> ➤ "How to Insert Steps Using the Step Generator" on page 900 ➤ "How to Select an Item for Your Step" on page 489
See also	<ul style="list-style-type: none"> ➤ "Managing Test Objects in Object Repositories" on page 159 ➤ "Native Properties and Operations" on page 962 ➤ "Visual Relation Identifier Dialog Box" on page 210

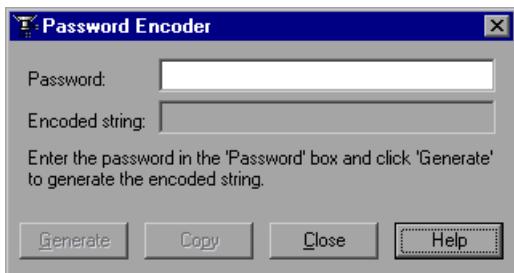
User interface elements are described below (unlabeled elements are shown in angle brackets):

UI Elements	Description
	<p>Find Next. Searches from the currently selected node, displaying all objects that match your criteria. The first object in the list that matches your criteria is highlighted.</p> <p>You can 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.</p> <p>Tip: Press F3 to find the next object that matches your search criteria, or SHIFT+F3 to find the previous match.</p>
	<p><Pointing hand>. Enables you to select an object from your application and add it to the shared object repository so that you can use it in your test steps. This is useful if the shared object repository does not include the test object that you need for a particular step.</p> <p>Click on the required object in your application to add it to the shared object repository tree. In some environments, as you move the pointer over your application, the available test objects are highlighted. For details, see "Tips for Using the Pointing Hand" on page 157.</p>
Name	<p>The full or partial name of the object to find. This option is useful if the object repository is very large. If you leave the box empty, all objects that match the selected object Type are displayed.</p> <p>Example: Enter p to search for all object names containing the letter p.</p>

UI Elements	Description
Type	<p>The type of object for which to search. Specify a type or select <All> to search for the object in all the object types.</p> <p>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.</p> <p>Example: Suppose you 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.</p>
<object repository tree>	<p>List of objects in the object repositories associated with the current action. You can select a test object from this tree to insert into a step.</p>
Insert test object	<p>Inserts the selected test object in the step.</p> <p>Note: This option is available only when this dialog box is opened from the Keyword View.</p>
Insert run-time object	<p>Inserts an Object statement step for the selected test object.</p> <p>(Relevant for any object that supports the .Object property.)</p> <p>Note: This option is available only when this dialog box is opened from the Keyword View.</p>

Password Encoder Tool

This tool enables you to encode passwords so that you can use the resulting strings as method arguments or data table parameter values (enabling you to place secure values into the data table). 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.



To access	From the Windows menu, select Start > Programs > HP QuickTest Professional > Tools > Password Encoder .
See also	You can also encrypt strings in data table cells using the Encrypt option in the Data Table menu. For details, see "Data Menu (Data Table)" on page 1345.

User interface elements are described below:

UI Elements	Description
Password	The password to encrypt.
Encoded string	The encrypted string that is generated when you click Generate .
Generate	Encrypts the string you entered in the Password box.
Copy	Copies the encrypted password to the Clipboard. You can then paste it as value for a method argument or insert it as a secure value in the data table.

Troubleshooting and Limitations - Keyword View

This section describes troubleshooting and limitations for the Keyword View.

When working with the **Object** property in the Keyword View, it may take a long time for QuickTest to retrieve IntelliSense information for the step.

Workaround: Use the Expert View when working with the **Object** property.

14

Actions

This chapter includes:

Concepts

- Actions Overview on page 523
- Action Types on page 525
- Action and Test Iterations Using the Data Table on page 526
- Calls to Existing Actions and Copies of Actions on page 529
- Action Parameters on page 531
- Sharing Action Information on page 533
- Action Syntax in the Expert View on page 535
- Considerations for Working with Actions on page 538

Tasks

- How to Use Actions in Your Test on page 542
- How to Nest Actions - Use-Case Scenario on page 547
- How to Use Action Parameters - Use-Case Scenario on page 548

Reference

- Action Call Properties Dialog Box on page 550
- Action Properties Dialog Box on page 557
- Action Toolbar in the Keyword View on page 576
- Insert Call to New Action Dialog Box on page 578
- Rename Action Dialog Box on page 580
- Select Action Dialog Box on page 582

Chapter 14 • Actions

► Split Action Dialog Box on page 585

Troubleshooting and Limitations - Actions on page 587

Concepts

Actions Overview

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.

When to Use 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.

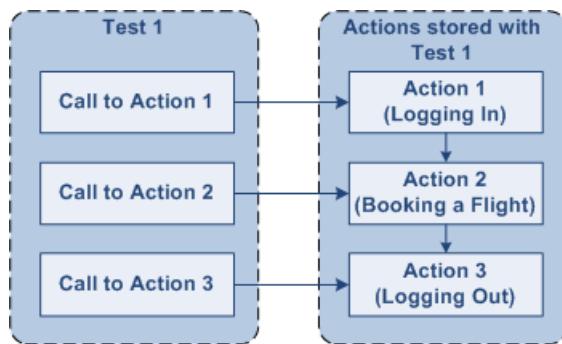
For every action called in your test, QuickTest creates a corresponding action sheet in the data table so that you can enter data table parameters that are specific to that action only. For more information on global and action data sheets, see "Action and Test Iterations Using the Data Table" on page 526. For information on parameterizing tests, see Chapter 22, "Parameterizing Values," and Chapter 23, "Output Values."

You can also 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 "Action Parameters" on page 531.

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:



 **Action Types**

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. For details, see "Test Flow Pane" on page 1395.

When you run a test with multiple actions, the run 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 Run Results Viewer, see Chapter 31, "Run Results Viewer."

QuickTest provides the following types 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. 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. 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.

Non-reusable action

An action that can be called only in the test with which it is stored, and can be called only once. 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.

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, see "Insert Call to New Action Dialog Box" on page 578.

Nested 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 "How to Nest Actions - Use-Case Scenario" on page 547). Complex tests may have many actions and may share actions with other tests.

Action and Test Iterations Using the Data Table

When you use the data table to output a value or to add a parameter to your test or action, you can specify whether to store the data in the **Global** data sheet or in the **action** data sheet.

- **Global sheet.** Enables you to define parameters for any action. 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. This enables

you to pass information between actions. For details, see "Global Sheet" on page 1328.

- **Action sheet.** Enables you to insert data that applies only to that action. 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. For details, see "Action Sheets" on page 1329.

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" on page 550.

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.

For more information on the data table, see Chapter 38, "Data Table Pane." For more information on parameterization, see Chapter 22, "Parameterizing Values." For more information on output values, see Chapter 23, "Output Values."

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.

A single test may include both global data table parameters and action (local) data table parameters.

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.

Note: If QuickTest is integrating with HP ALM, make sure to save your data table parameters in the Global sheet and not in a specific action sheet. For details, see "Data Awareness in HP ALM" on page 1613.

Calls to Existing Actions and Copies of 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.

You can also call actions dynamically during a run session using the `LoadAndRunAction` statement. For details, see the Utility section of the *HP QuickTest Professional Object Model Reference*.

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.

Calls to Existing Actions

You can insert a call to a reusable action that is stored in your current test (local action), or in any other test (external action). Inserting a call to an existing action is 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.

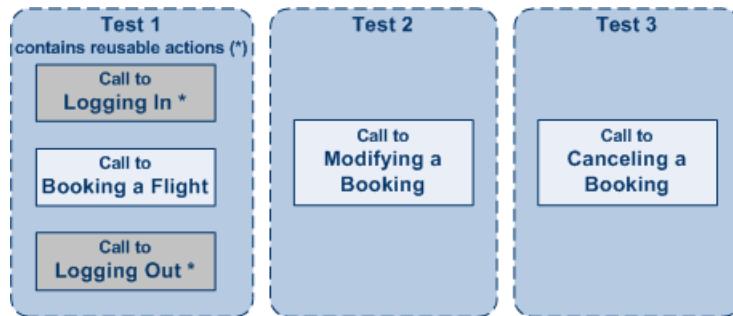
If the called external action has data in the data table, however, you can choose whether you want the data from the action's data sheet to be imported as a local, editable copy, or whether you want to use the (read-only) data from the original action. (Columns and data from the called action's global data sheet is always imported into the calling test as a local, editable copy.) For more information, see "External Action Tab (Action Properties Dialog Box)" on page 573.

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.

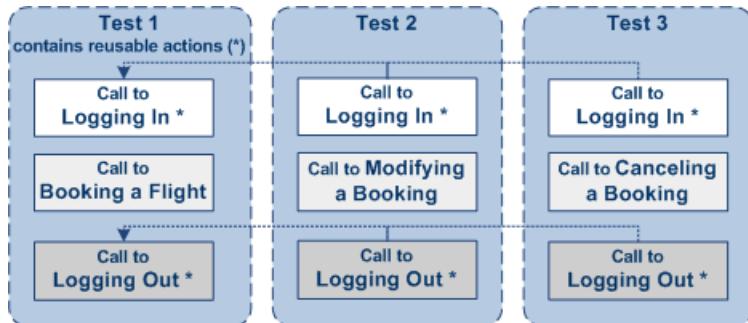
Example

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.



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 any of the following:

- the test (for a top-level action)
- the parameters of the parent action that calls it (for a nested action)
- 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 information on defining action parameters and the values used in action calls, see "Parameters Tab (Action Properties Dialog Box)" on page 563, and "Parameter Values Tab (Action Call Properties Dialog Box)" on page 554.

When to Use 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.

Storing Action Parameters

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.

 **Sharing Action Information**

There are several ways to share or pass values from one action to other actions:

- "Output Options Dialog Box" on page 808. 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.
- "Sharing Values Using the Global Data Table" on page 533. Store values from one action in the global data table and use these values as data table parameters in other actions.
- "Sharing Values Using Environment Variables" on page 534. Set a value from one action as a user-defined environment variable and then use the environment variable in other actions.
- "Sharing Values Using the Dictionary Object" on page 534. Add values to a Dictionary object in one action and retrieve the values in other actions.

 **Sharing Values Using 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 more information on output values, see Chapter 23, "Output Values." For more information on parameterization, see Chapter 22, "Parameterizing Values." For more information on `DataTable` objects and methods, see Chapter 38, "Data Table Pane," and the *HP QuickTest Professional Object Model Reference*.

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 purchases 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.

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 22, "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, 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 add the reserved Dictionary object to the registry and restart QuickTest, you can add and remove values to and from the Dictionary in one action and retrieve the values in another action from the same test.

For more information on the Dictionary object, see the VBScript Reference documentation (**Help > QuickTest Professional Help > VBScript Reference > Script Runtime**).

Example

Suppose 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")
```

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.

This section includes:

- ▶ "Calling Actions Using Basic Syntax" on page 535
- ▶ "Calling Actions with Parameters" on page 536
- ▶ "Storing Action Return Values" on page 537

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

Example

To call the **Select Flight** action and run it one iteration:

RunAction "Select Flight", onIteration

To call the **Select Flight** action and run it as many iterations as there are rows in the data table:

RunAction "Select Flight", allIterations

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, 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 from which the argument should take its value. This can be a data table parameter, an environment parameter, or an action input parameter of the calling action.
- ▶ 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

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", onelteration, "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", onelteration, 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", onelteration
RunAction "Action3", onelteration, 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 27, "Working in the Expert View and Function Library Windows." For more information on the RunAction statement, see the *HP QuickTest Professional Object Model Reference*.

Considerations for Working with Actions

Inserting Actions

- 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 Action** tab of the Action Properties dialog box, select **Use a local, editable copy**.

Storing Actions

- If you expect other users to open your tests and all actions in your tests are stored in the same drive, you should use relative paths for your reusable actions so that other users will be able to open your tests even if they have mapped their network drives differently.
- If you are working with the Resources and Dependencies model with Quality Center 10.00 or HP ALM, specify an absolute Quality Center path. For more information, see "Relative Paths and Quality Center" on page 1656.

Organizing Actions in Your Test

- 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.
- 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.

Naming Actions

- 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 "Insert Call to New Action Dialog Box" on page 578.)
- If a test contains a call to an action from another test, and that other test was renamed in Quality Center, the original test name still appears (in square brackets) in the Test Flow pane. The obsolete name in the Test Flow pane does not affect QuickTest's ability to locate and run the action. If it is important to display the correct test name, delete the action call from the test and reinsert it.

Associating Object Repositories

- 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 a test.
- 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 "Folders Pane (Options Dialog Box)" on page 1431.
- You can associate an object repository dynamically during a run session using the `RepositoriesCollection` statement. For details, see the Utility section of the *HP QuickTest Professional Object Model Reference*.

Action Parameters

For details on using action parameters, see "Considerations for Setting Action Parameters" on page 729.

Considerations for Removing Action Calls

- If a test contains a call to an action that you removed, 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. For more information, see "Missing Resources Pane" on page 1363.

- You can remove calls to actions from the following locations:

Location	Description
Resources pane	<p>Remove all calls to a specific action.</p> <p>Notes:</p> <ul style="list-style-type: none"> ➤ 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 / Keyword View	<p>Remove specific calls to an action.</p> <p>Notes:</p> <ul style="list-style-type: none"> ➤ 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.

Tasks

How to Use Actions in Your Test

This task describes the different operations you can perform to use actions in your test.

This task includes the following steps:

- "Insert a call to new action" on page 542
- "Insert a call to existing action or copy of action" on page 542
- "Modify action properties" on page 543
- "Create an action template" on page 543
- "Nest an action within an existing action" on page 543
- "Remove a call to an action or delete an action" on page 544
- "Split an action" on page 546
- "Exit an action using programming statements" on page 546

Insert a call to new action

- 1 Select **Insert > Call to New Action**, or click the **Insert Call to New Action** button  on the **Insert** toolbar.
- 2 Define the action name and location in the "Action Properties Dialog Box" on page 557.

Insert a call to existing action or copy of action

- 1 Select one of the following:
 - **Insert > Call to Copy of Action.** For details, see "Calls to Copies of Actions" on page 529.
 - **Insert > Call to Existing Action.** For details, see "Calls to Existing Actions" on page 529.
- 2 Define the action settings in the "Select Action Dialog Box" on page 582.

Modify action properties

Use the "Action Properties Dialog Box" on page 557 to modify action properties. The following tabs are included:

- "General Tab (Action Properties Dialog Box)" on page 559
- "Parameters Tab (Action Properties Dialog Box)" on page 563
- "Associated Repositories Tab (Action Properties Dialog Box)" on page 567
- "Used By Tab (Action Properties Dialog Box)" on page 571
- "External Action Tab (Action Properties Dialog Box)" on page 573

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.

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 "Insert Call to New Action Dialog Box" on page 578, or for inserting a call to a copy of an action or a call to an existing action as described in "Insert Call to New Action Dialog Box" on page 578.

For a user-case scenario, see "How to Nest Actions - Use-Case Scenario" on page 547.

Remove a call to an action or delete an action

- 1** In the Resources pane, the Test Flow pane, or the Keyword View, do one of the following:
 - 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.
-

Note: If you are deleting a reusable action that is called by another action, and if your test is stored in Quality Center and is using the resources and dependencies model (described on page 1652), a list of the actions that call the action that you are deleting is displayed in the confirmation message box.

To copy any or all of the actions from the list to the Windows Clipboard:

- 1** Select the relevant actions from the list.
 - 2** Right-click and select **Copy Selected**, or press **CTRL+C** on your keyboard.
-

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 the 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>Caution: 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.

Split an action

- 1 Select the step before which you want the new (second) action to begin.
 - 1 Select **Edit > Action > Split Action**, or click the **Split Action** button  in the toolbar to open the "Split Action Dialog Box" on page 585.
 - 2 Define the settings for the new actions.
-

Note: If you split an action 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.
-

Exit an action using programming statements

Use one of the following exit action statements:

- **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 Run 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 Run Results, see Chapter 31, "Run Results Viewer."

How to Nest Actions - Use-Case Scenario

Suppose you parameterized a step in which 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.

For more information on inserting conditional statements, see "Conditional Statements" on page 890.

View Example in the Keyword View

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	GetROPProperty	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.

View Example in the Expert View

```

Browser("Membership Preference").Page("Membership Preference").
WebRadioGroup("MemType").Select DataTable("memtype", dtGlobalSheet)

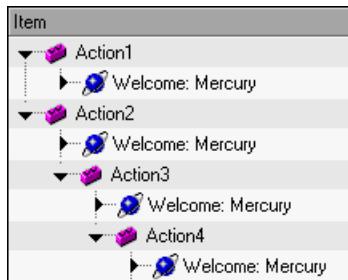
Mem_Type=Browser("Membership Preference").
Page("Membership Preference").WebRadioGroup("MemType").
GetROPProperty ("value")

If Mem_Type="paid" Then
    RunAction "Paid_Mem", onelteration
Elseif Mem_Type = "free" Then
    RunAction "Free_Mem", onelteration
Else
    RunAction "Preferred", onelteration
End If

```

How to Use Action Parameters - Use-Case Scenario

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 536.
- 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 536.

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 728.

Reference

Action Call Properties Dialog Box

This 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.

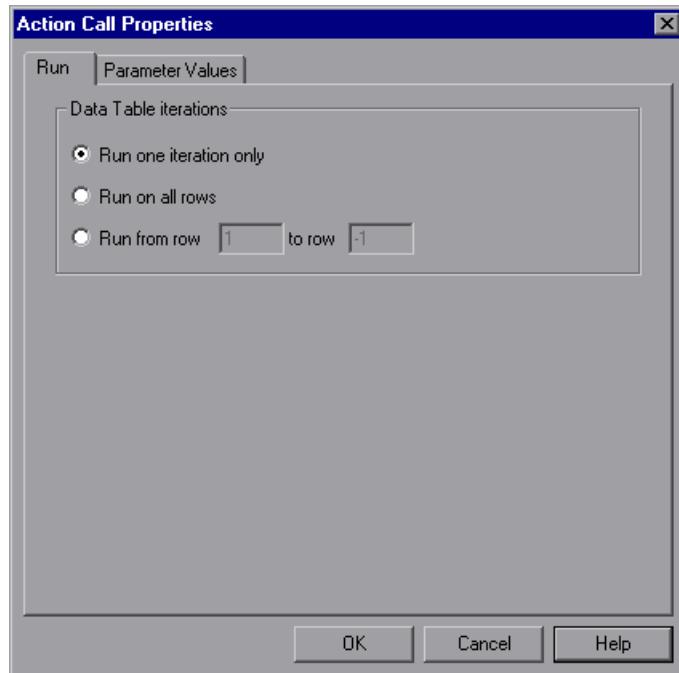
To access	<p>Do one of the following:</p> <ul style="list-style-type: none"> ► Right-click an action node in the Keyword View or Test Flow pane and select Action Call Properties. ► Select Edit > Action > Action Call Properties from the Keyword View when an action node is highlighted.
Important information	<ul style="list-style-type: none"> ► This dialog box enables you to set options that apply only to a specific action call. ► You can also define action calls and action call parameters in the Expert View. For more information, see "Action Syntax in the Expert View" on page 535
Relevant tasks	"How to Use Actions in Your Test" on page 542

The Action Call Properties dialog box contains the following tabs:

- Run tab (described below)
- Parameter Values tab (describe on page 552)

Run Tab (Action Call Properties Dialog Box)

This tab enables you 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.



To access	Select Edit > Action > Action Call Properties > Run tab .
Important information	<ul style="list-style-type: none"> ► 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. ► This tab 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 46, "Individual Test Settings."

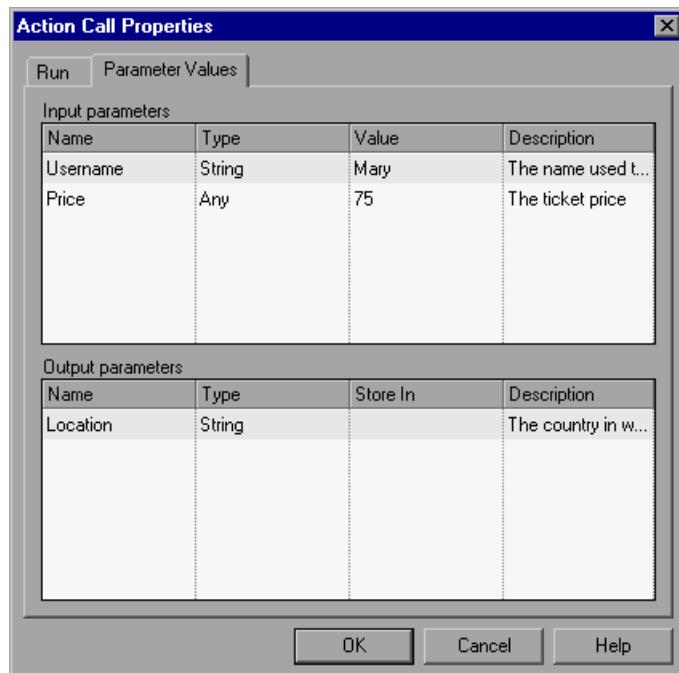
Relevant tasks	"How to Use Actions in Your Test" on page 542
See also	"Parameter Values Tab (Action Call Properties Dialog Box)" on page 554

User interface elements are described below:

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.

Parameter Values Tab (Action Call Properties Dialog Box)

This tab enables you 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.



To access	Select Edit > Action > Action Call Properties > Parameter Values tab.
Important information	<ul style="list-style-type: none">▶ 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.
Relevant tasks	"How to Use Actions in Your Test" on page 542
See also	"Run Tab (Action Call Properties Dialog Box)" on page 552

User interface elements are described below:

UI Elements	Description
Name	The name for the parameter. (Action parameter names are case sensitive.)
Type	<p>The value type for the parameter. The following options are available:</p> <ul style="list-style-type: none"> ▶ 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 this 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 this 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 this 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. If you select this 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.

UI Elements	Description
Default Value (Input Parameters)	The default value for the parameter. 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.
Description	The 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.

Action Properties Dialog Box

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.

To access	Select Edit > Action > Action Properties .
Important information	You can also define actions and action parameters in the Expert View. For more information, see "Action Syntax in the Expert View" on page 535.
Relevant tasks	"How to Use Actions in Your Test" on page 542

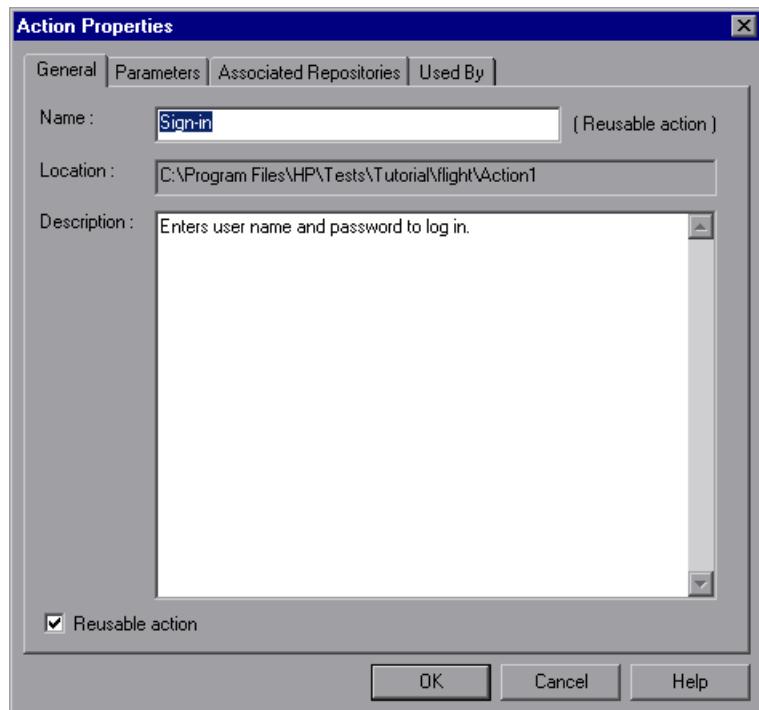
The Action Properties dialog box contains the following tabs:

- General tab (described on page 559). Enables you to modify the name of an action, add or edit an action's description, or change the reusability status of the action.
- Parameters tab (described on page 563). Enables you to define input and output parameters to be used by the action.
- Associated Repositories tab (described on page 567). Enables you to specify the object repositories that are associated with the action.

- Used By tab (described on page 571). Enables you to view a list of the tests and actions that contain calls to this particular action. Available only if your tests are stored in Quality Center and are using the resources and dependencies model.
- External Action tab (described on page 573). Enables you to set the data table definitions. Available only when viewing properties for external actions. When this tab is displayed, the other tabs are read-only.

General Tab (Action Properties Dialog Box)

This tab enables you to modify the name of an action, add or edit an action's description, or change the reusability status of the action.



To access	Select Edit > Action > Action Properties > General tab.
Important information	<ul style="list-style-type: none"> ➤ 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 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 "Action Toolbar in the Keyword View" on page 576.
Relevant tasks	"How to Use Actions in Your Test" on page 542
See also	<ul style="list-style-type: none"> ➤ "Parameters Tab (Action Properties Dialog Box)" on page 563 ➤ "Associated Repositories Tab (Action Properties Dialog Box)" on page 567 ➤ "Used By Tab (Action Properties Dialog Box)" on page 571 ➤ "External Action Tab (Action Properties Dialog Box)" on page 573

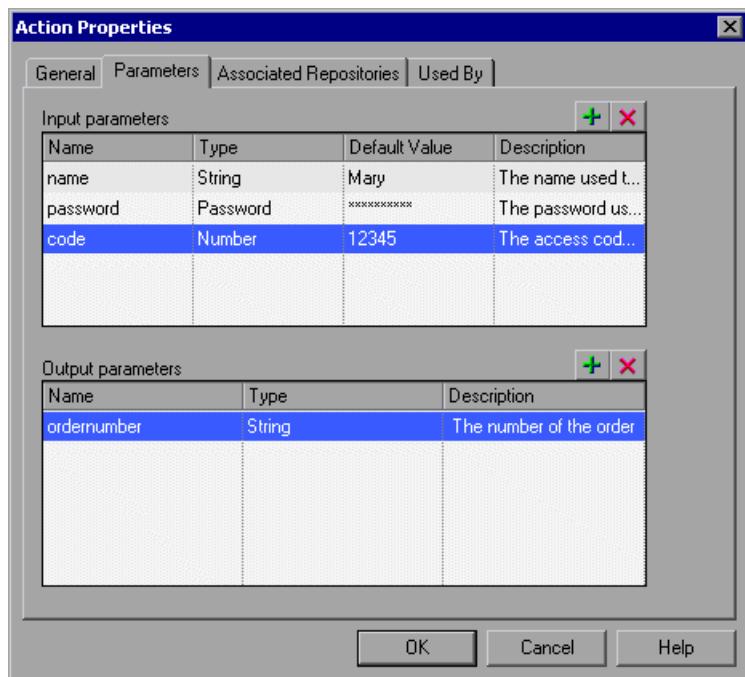
User interface elements are described below:

UI Elements	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>If the action is a reusable action or an external action, then Reusable action or External Action is displayed next to the action name.</p> <p>You can rename the action, as needed. For a list of naming conventions, see "Naming Conventions" on page 1779.</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 "How to Nest Actions - Use-Case Scenario" on page 547.</p> <p>Note: You can also add a description when inserting a call to a new action. For more information, see "Insert Call to New Action Dialog Box" on page 578.</p>

UI Elements	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> <p>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.</p>

Parameters Tab (Action Properties Dialog Box)

This tab enables you to define input and output parameters for use in the selected action's iterations.



To access	Select Edit > Action > Action Properties > Parameters tab.
Important information	When you delete an action parameter, make sure that you also delete any steps that use the action parameter.

Relevant tasks	"How to Use Actions in Your Test" on page 542
See also	<ul style="list-style-type: none">▶ "General Tab (Action Properties Dialog Box)" on page 559▶ "Associated Repositories Tab (Action Properties Dialog Box)" on page 567▶ "Used By Tab (Action Properties Dialog Box)" on page 571▶ "External Action Tab (Action Properties Dialog Box)" on page 573

User interface elements are described below:

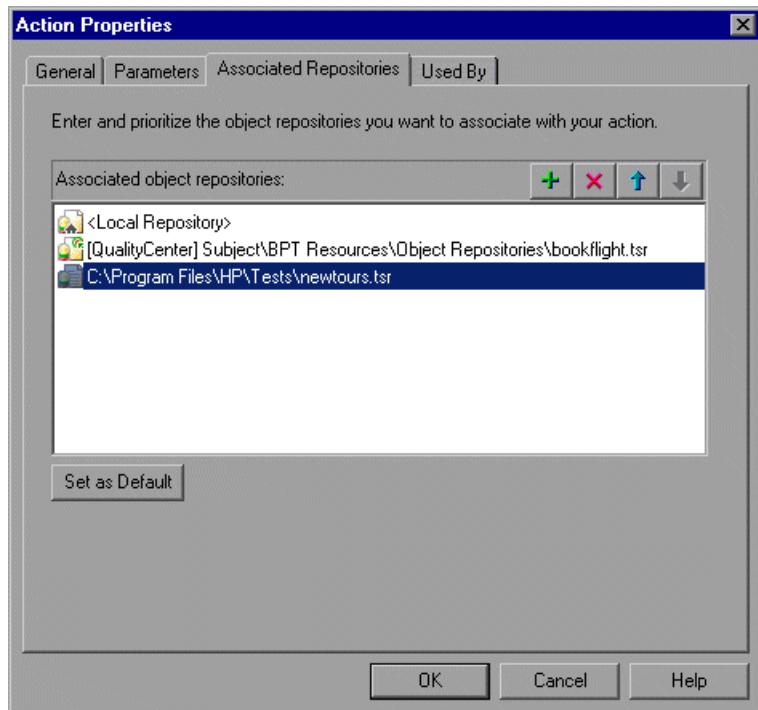
UI Elements	Description
	Add. Adds a new parameter to the list.
	Delete. Removes the selected parameter from the list.
Name	The name for the parameter. (Action parameter names are case sensitive.)

UI Elements	Description
Type	<p>The value type for the parameter. The following options are available:</p> <ul style="list-style-type: none"> ➤ 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 this 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 this 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 this 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. If you select this 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.

UI Elements	Description
Default Value (Input Parameters)	The default value for the parameter. 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.
Description	The 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.

Associated Repositories Tab (Action Properties Dialog Box)

This tab enables you to associate object repositories with an action, and manage the association order and default settings.



To access	Select Edit > Action > Action Properties > Associated Repositories tab.
Important information	<ul style="list-style-type: none"> ➤ You can also associate shared object repositories with multiple actions simultaneously, using the Associate Repositories dialog box. For more information, see "Associate Repositories Dialog Box" on page 229. ➤ If you are not connected to a Quality Center project, all associated object repositories that are stored in your Quality Center project are listed as missing in the Missing Resources pane. For more information on missing resources, see Chapter , "Missing Resources Pane." ➤ If an object repository cannot be found, QuickTest displays a warning message when you click this tab. 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 details on resolving missing resources, see Chapter , "Missing Resources Pane."
Relevant tasks	"How to Use Actions in Your Test" on page 542
See also	<ul style="list-style-type: none"> ➤ "General Tab (Action Properties Dialog Box)" on page 559 ➤ "Parameters Tab (Action Properties Dialog Box)" on page 563 ➤ "Used By Tab (Action Properties Dialog Box)" on page 571 ➤ "External Action Tab (Action Properties Dialog Box)" on page 573

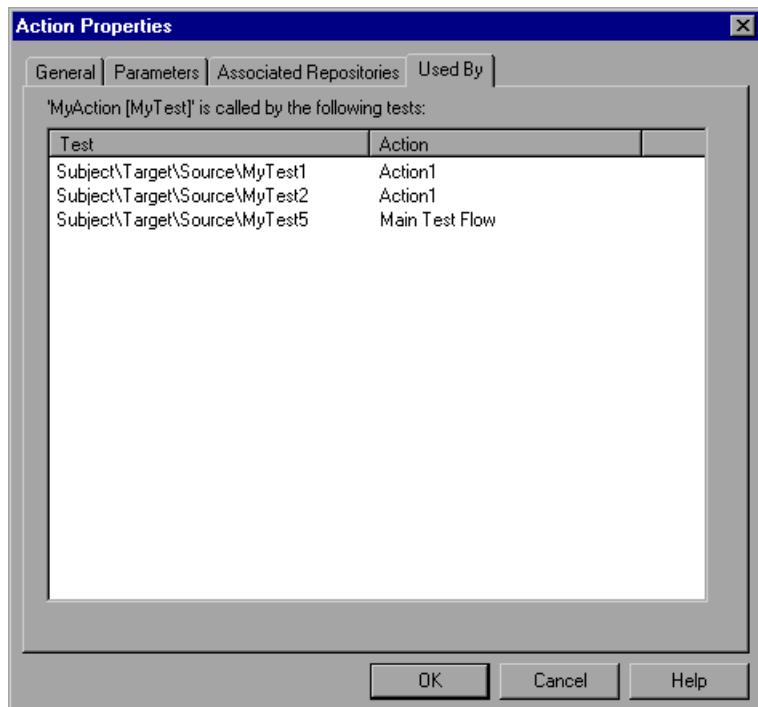
User interface elements are described below:

UI Elements	Description
Associated object repositories	<p>The list of all associated object repositories.</p> <p>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).</p> <p>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).</p> <p>For more information, see "Folders Pane (Options Dialog Box)" on page 1431, and "Relative Paths in QuickTest" on page 391.</p>
	<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 or HP ALM, specify an absolute Quality Center path. For more information, see "Relative Paths and Quality Center" on page 1656.</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.

UI Elements	Description
	Removes an associated object repository from the list.
	Assigns a higher priority to the selected object repository.
	Assigns a lower priority to the selected object repository.
Set as Default	<p>Sets the current list of object repositories as the default list to be associated with all new actions in this test.</p> <p>Note: The Set as Default option is enabled only when:</p> <ul style="list-style-type: none">➤ One or more shared object repositories are associated with any local action in the test.➤ The list of object repositories associated with this action is different from the list associated with other local actions in this test.

Used By Tab (Action Properties Dialog Box)

This tab enables you to view a list of the tests and actions that contain calls to this particular action. This tab is available only if your tests are stored in Quality Center and are using the resources and dependencies model.



To access	Select Edit > Action > Action Properties > Used By tab.
Important information	<ul style="list-style-type: none"> ▶ This list is the same list that is displayed in the Dependencies tab of the Test Plan module in Quality Center. For more information, see "Resources and Dependencies Model" on page 1651. ▶ Calls to external actions that are stored in Quality Center 10.00 or HP ALM via a relative path are not listed in this pane because they are not using the resources and dependencies model.

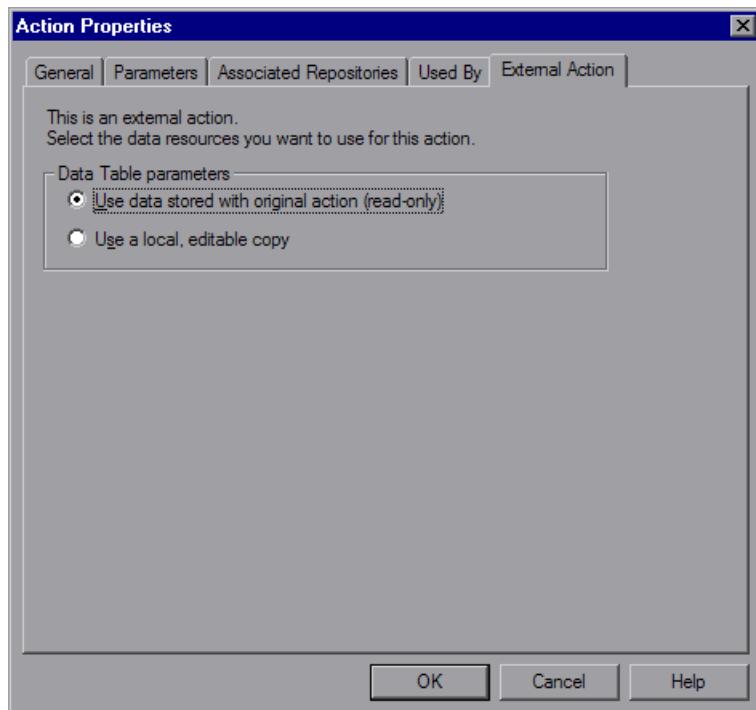
Relevant tasks	"How to Use Actions in Your Test" on page 542
See also	<ul style="list-style-type: none">▶ "General Tab (Action Properties Dialog Box)" on page 559▶ "Parameters Tab (Action Properties Dialog Box)" on page 563▶ "Associated Repositories Tab (Action Properties Dialog Box)" on page 567▶ "External Action Tab (Action Properties Dialog Box)" on page 573

User interface elements are described below:

UI Elements	Description
Test	The Quality Center path of the test containing a call to this action.
Action	The name of the action containing a call to this action. If the called action is the top-level action in the test from which it is called, Main Test Flow is displayed instead.

External Action Tab (Action Properties Dialog Box)

This tab enables you to specify the data source for the selected action. This tab is available only when viewing properties for external actions.



To access	Select Edit > Action > Action Properties > External Action tab.
Important information	<ul style="list-style-type: none"> ► When this tab is displayed, the other tabs are read-only. ► When you insert a call to an external action, the action is inserted in read-only format, and the Record button is disabled. <p>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.</p>

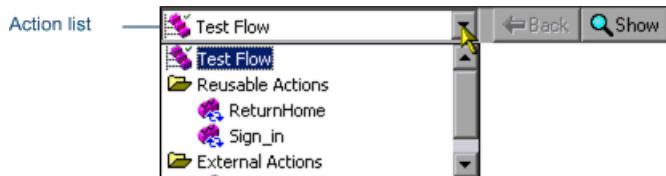
Relevant tasks	"How to Use Actions in Your Test" on page 542
See also	<ul style="list-style-type: none">▶ "General Tab (Action Properties Dialog Box)" on page 559▶ "Parameters Tab (Action Properties Dialog Box)" on page 563▶ "Associated Repositories Tab (Action Properties Dialog Box)" on page 567▶ "Used By Tab (Action Properties Dialog Box)" on page 571

User interface elements are described below:

UI Elements	Description
Data Table parameters	<p>Indicates where to store the data in the action's data table:</p> <ul style="list-style-type: none"> ➤ Use data stored with the original action (read-only). Uses the original action's data. 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. ➤ Use a local, editable copy. Uses an editable copy of the data in the test's data table. 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>

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).



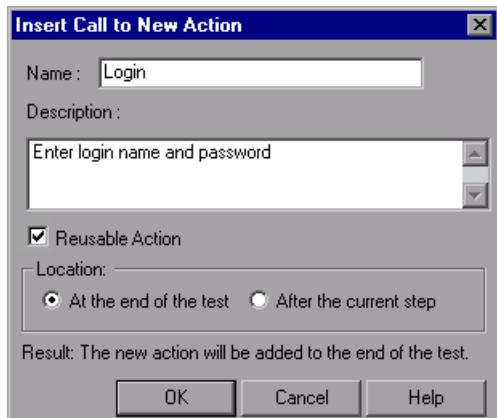
To access	Select View > Toolbars > Action . Note: The Action toolbar is automatically displayed above the Keyword View when a reusable or external action is included in test.
Important information	<ul style="list-style-type: none"> ➤ In the Expert View, the Action List is always visible and the Expert View always displays the steps for the selected action. For more information on the Expert View, see Chapter 27, "Working in the Expert View and Function Library Windows." ➤ 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.
Relevant tasks	"How to Use Actions in Your Test" on page 542

User interface elements are described below:

UI Elements	Description
Action List	<p>The list of actions in the current test. The following options are available:</p> <ul style="list-style-type: none">▶ <action>. All the details of the selected reusable or external action.▶ Test Flow. The overall flow of your test with all the calls to the top-level actions in your test. <p>Notes:</p> <ul style="list-style-type: none">▶ The test flow also enables you to view and edit the individual steps of non-reusable actions.▶ 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 "General Tab (Action Properties Dialog Box)" on page 559.

Insert Call to New Action Dialog Box

This dialog box enables you to insert a call to a new action from the Keyword View.



To access	In the Keyword View, do one of the following: <ul style="list-style-type: none"> ➤ Select Insert > Call to New Action. ➤ Click the Insert Call to New Action button  on the Insert toolbar.
Important information	<ul style="list-style-type: none"> ➤ 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 "How to Nest Actions - Use-Case Scenario" on page 547. ➤ 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 "Test Flow Pane" on page 1395 and "How to Move an Action or Step" on page 494.
Relevant tasks	"How to Use Actions in Your Test" on page 542

User interface elements are described below:

UI Elements	Description
Name	Enables you to type a new action name or accept the default name. If you rename the action, see "Naming Conventions" on page 1779.
Description	<p>The description of the action. You can also add an action description at a later time using the Action Properties dialog box.</p> <p>Note: Descriptions of actions are displayed in the Select Action dialog box. The description helps you to choose an existing action you want to call. For more information, see "General Tab (Action Properties Dialog Box)" on page 559.</p>
Reusable Action	When selected, enables you to call the action from other tests or multiple times from within this test (default). You can set or modify this setting at a later time using the Action Properties dialog box.
Location	<p>The location to insert the new action. The following options are available:</p> <ul style="list-style-type: none"> ➤ At the end of the test. Creates a call from the test flow to a top-level action. ➤ After the current step. Inserts the call to the action from within the current action (nests the action). <p>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).</p>

Rename Action Dialog Box

This dialog box enables you to rename an action from the Keyword View or Expert View.



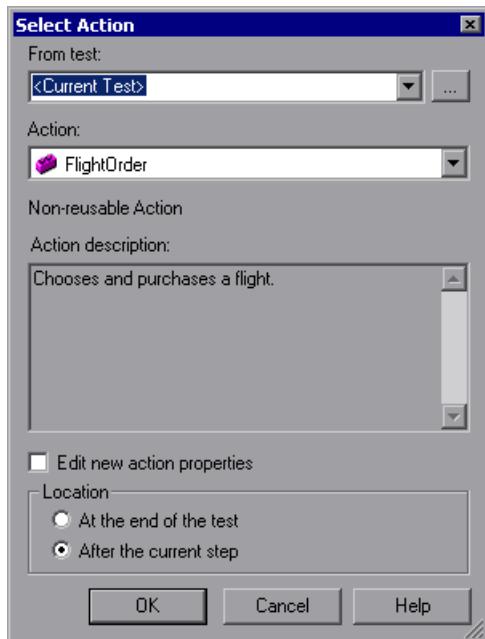
To access	<p>In the Keyword View, select the call to the action you want to rename and select Edit > Action > Rename Action.</p> <p>Note: You can also press SHIFT + F2 to open the Rename Action dialog box.</p>
Important information	<ul style="list-style-type: none"> ➤ When you rename an action, consider 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. ➤ 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. ➤ 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 "Used By Tab (Action Properties Dialog Box)" on page 571.
Relevant tasks	<p>"How to Use Actions in Your Test" on page 542</p>

User interface elements are described below:

UI Elements	Description
New name	The new name for the action. For a list of naming conventions, see "Naming Conventions" on page 1779.

Select Action Dialog Box

This dialog box enables you to select actions to use in your test as copies or as external actions.



To access	<p>This dialog box can be accessed by selecting one of the following:</p> <ul style="list-style-type: none"> ➤ Insert > Call to Copy of Action ➤ Insert > Call to Existing Action
Important information	<ul style="list-style-type: none"> ➤ When inserting a call to copy of action, you can view the location of the original action in the General tab of the Action Properties dialog box. ➤ When inserting a call to existing action, 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.

Related tasks	"How to Use Actions in Your Test" on page 542
See also	"Keyword View Overview" on page 484

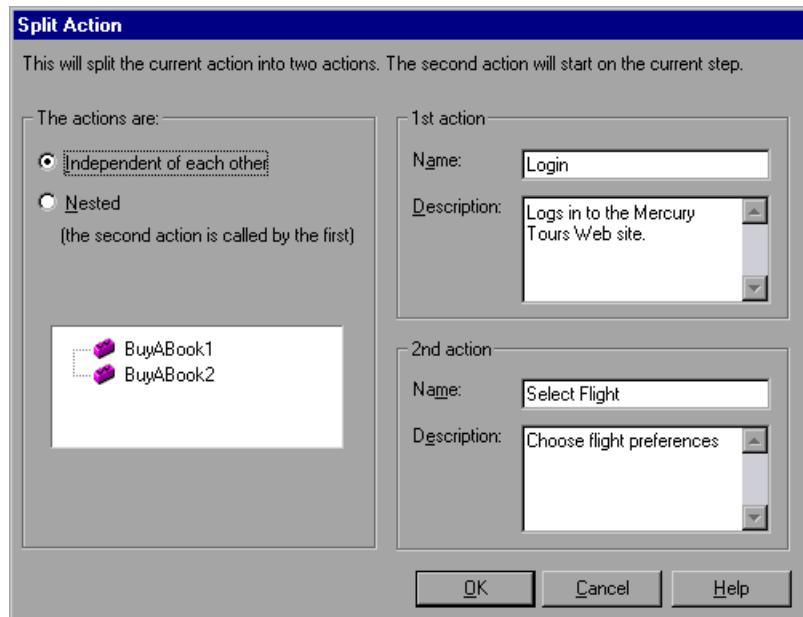
User interface elements are described below:

UI Elements	Description
From test	<p>The name of the test containing the action you want to use. You can also enter a Quality Center folder or a relative path.</p> <p>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 "Folders Pane (Options Dialog Box)" on page 1431 and "Relative Paths in QuickTest" on page 391.</p> <p>If you are working with the Resources and Dependencies model with Quality Center 10.00 or HP ALM, specify an absolute Quality Center path. For more information, see "Relative Paths and Quality Center" on page 1656.</p>
Action	<p>All local actions (actions that are stored with the test you selected).</p> <p>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 "General Tab (Action Properties Dialog Box)" on page 559.</p> <p>Note: When inserting an existing action, 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</p>

UI Elements	Description
Edit new action properties (Call to Copy of Action)	<p>When selected, the Action Properties dialog box is displayed when you click OK. You can then modify the action properties as described in "Action Properties Dialog Box" on page 557</p> <p>Tip: 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.</p>
Location	<p>The location to insert the action. The following options are available:</p> <ul style="list-style-type: none">➤ At the end of the test➤ After the current step <p>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, and the After the current step option is disabled.</p>

Split Action Dialog Box

This dialog box enables you to split an action and define the settings for the new actions.



To access	Do one of the following: ► Select Edit > Action > Split Action . ► Click the Split Action button  in the toolbar.
------------------	--

Important information	<p>You cannot split an action, and the option is disabled when:</p> <ul style="list-style-type: none"> ➤ 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 <p>Caution: 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. However, If a reusable action is called from another test, splitting it may cause the calling test to fail.</p>
Relevant tasks	"How to Use Actions in Your Test" on page 542

User interface elements are described below:

UI Elements	Description
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.
Name	The name for the first and second actions.
Description	The description for the first and second actions.

Troubleshooting and Limitations - Actions

This section describes troubleshooting and limitations for working with actions.

- If you make a copy of an existing test (either in the file system or in Quality Center), then you cannot insert a call to the same action from both of these tests into the same test.

Workaround: Instead of creating a copy of the test, use **Save As** to create a duplicate of the test.

- You cannot create a call to a new action called **Global**, since Global is the name reserved for the Global sheet in the data table. If you create an action called Global, you will not be able to select the local or global data sheet when parameterizing a identification property.
- Calling an external action from a function library using the **RunAction** statement results in a run-time error.

Workaround: Call the action using the **LoadAndRunAction** statement. For details, see the Utility section of the *HP QuickTest Professional Object Model Reference*.

Part IV

Enhancing Tests

15

Checkpoints Overview

This chapter includes:

Concepts

- Checkpoints Overview on page 592
- Checkpoint Types on page 593

Tasks

- How to Insert a Checkpoint Step in a Test on page 598

Reference

- Add Existing Checkpoint Dialog Box on page 601

Troubleshooting and Limitations - Creating Checkpoints on page 603

Concepts

Checkpoints Overview

QuickTest enables you to add checks to your test. A **checkpoint** is a verification that compares the current value for specified properties or current state of other characteristics of an object with the expected value or characteristics. This helps 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 Run Results Viewer.

Adding Existing Checkpoints to a Test

QuickTest enables you to reuse existing checkpoints. When you create checkpoints, consider which checkpoints can be reused in multiple locations in your test or in multiple tests. 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.

Checkpoint Types

You can insert the following checkpoint types to check objects in an application:

Checkpoint Type	Description
Standard Checkpoint	<p>Checks property values of an object in your application. 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.</p> <p>Standard checkpoints are supported for all add-in environments (see "Supported Checkpoints" on page 1784).</p> <p>For more information on standard checkpoints, see Chapter 16, "Standard Checkpoints."</p>
Image Checkpoint	<p>Checks the value of an image in your application. For example, you can check that a selected image's source file is correct.</p> <p>You create an image checkpoint by inserting a standard checkpoint on an image object.</p> <p>Image checkpoints are supported for the Web add-in environment (see "Supported Checkpoints" on page 1784).</p> <p>For more information on image checkpoints, see Chapter 16, "Standard Checkpoints."</p>

Checkpoint Type	Description
Bitmap Checkpoint	<p>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. Using the bitmap checkpoint, you can check that the map zooms in correctly.</p> <p>You can create a bitmap checkpoint for any area in your application.</p> <p>Bitmap checkpoints are supported for all add-in environments (see "Supported Checkpoints" on page 1784).</p> <p>For more information on bitmap checkpoints, see Chapter 17, "Bitmap Checkpoints."</p>
Table Checkpoint	<p>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.</p> <p>You create a table checkpoint by inserting a standard checkpoint on a table object. For more information on table checkpoints, see "Table Checkpoints" on page 635.</p> <p>Table checkpoints are supported for all add-in environments that have a *Table test object. 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, see "Supported Checkpoints" on page 1784.</p>

Checkpoint Type	Description
Text Checkpoint	<p>Checks that a text string is displayed in the appropriate place in an 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".</p> <p>Text checkpoints are supported for most add-in environments (see "Supported Checkpoints" on page 1784).</p> <p>For more information on text checkpoints, see Chapter 19, "Text Checkpoints."</p>
Text Area Checkpoint	<p>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.</p> <p>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 1784). Text area checkpoints are also supported for some other add-in environments, such as Java.</p> <p>For more information on text area checkpoints, see Chapter 19, "Text Checkpoints."</p>

Checkpoint Type	Description
Accessibility Checkpoint	<p>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.</p> <p>Accessibility checkpoints are supported for the Web add-in environment (see "Supported Checkpoints" on page 1784).</p> <p>For more information on accessibility checkpoints, see the section on testing Web objects in the <i>HP QuickTest Professional Add-ins Guide</i>.</p>
Page Checkpoint	<p>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.</p> <p>You create a page checkpoint by inserting a standard checkpoint on a page object.</p> <p>Page checkpoints are supported for the Web add-in environment (see "Supported Checkpoints" on page 1784).</p> <p>For more information on page checkpoints, see the section on testing Web objects in the <i>HP QuickTest Professional Add-ins Guide</i>.</p>

Checkpoint Type	Description
Database Checkpoint	<p>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.</p> <p>Database checkpoints are supported for all add-in environments (see "Supported Checkpoints" on page 1784).</p> <p>For more information on database checkpoints, see Chapter 20, "Database Checkpoints."</p>
XML Checkpoint	<p>Checks the data content of XML documents in XML files or XML documents in Web pages and frames.</p> <p>For more information on XML checkpoints, see Chapter 21, "XML Checkpoints."</p> <p>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 1784).</p>

Tasks

How to Insert a Checkpoint Step in a Test

This task describes how to insert a checkpoint step while recording or editing your test. You can also add an existing checkpoint to your test. It is generally more convenient to define checkpoints after creating the initial test.

Note: This task describes the general process of inserting a new checkpoint step to your test. However, certain checkpoint types may require additional prerequisites or steps.

Prerequisites

- Make sure to review and note all relevant information and prerequisites for the type of checkpoint you are adding. For details, see the prerequisites for the relevant checkpoint type.
- To insert a checkpoint while recording, start a recording session before proceeding to the next step.
- 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 "Active Screen Pane (Options Dialog Box)" on page 1434.
- To insert or modify an existing checkpoint step while editing, you may need to open the application and display the relevant object before proceeding to the next step. This depends on the environment and the object type you are checking. For details, see the prerequisites for the checkpoint type.

Insert a new checkpoint step while recording your test



- 1 Select **Insert > Checkpoint** menu, or click the **Insert Checkpoint or Output Value** button in the toolbar. This displays a menu of checkpoint options that are relevant to the selected step.
- 2 Select the type of checkpoint. QuickTest is hidden, and the pointer changes to a pointing hand. In your application, click the object that you want to check.

For details on using the pointing hand, see "Tips for Using the Pointing Hand" on page 157.

Note: If the object in your application is associated with more than one location, The Object Selection dialog box opens. This dialog box enables you to select an object to check from the object tree. The objects in the tree are displayed in hierarchical order, based on the location you clicked in the Active Screen or application. For more information, see "Object Selection Dialog Box" on page 155.

- 3 Define the properties and expected values of the checkpoint. For details, refer to the user-interface description in the relevant checkpoint chapter.

Insert a new or existing checkpoint while editing your test

Select the step where you want to add the checkpoint and do one of the following:

- Select **Insert > Checkpoint**, and then select the relevant checkpoint option.
- Select **Insert > Checkpoint > Existing Checkpoint**. The Add Existing Checkpoint Dialog Box (described on page 601) opens.
- Right-click any object in the Active Screen and select the relevant checkpoint. You can create checkpoints for any object in the Active Screen even if the object is not part of any step in the Keyword View. (If the Active Screen is not in view, select the **Active Screen** toolbar button.)



Note: If the object in your application is associated with more than one location, The Object Selection dialog box opens. This dialog box enables you to select an object to check from the object tree. The objects in the tree are displayed in hierarchical order, based on the location you clicked in the Active Screen or application. For more information, see "Object Selection Dialog Box" on page 155.

Move checkpoint objects from the local object repository to a shared object repository - Optional

After you insert a checkpoint step, the checkpoint object is added to the local object repository. If you are using shared object repositories, you can move the new checkpoint/output value object to your shared object repository. For details, see "How to Export Local Objects to a Shared Object Repository" on page 226, and "How to Update a Shared Object Repository From a Local Object Repository" on page 347.

Tips and considerations for inserting checkpoints

- 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"))
```

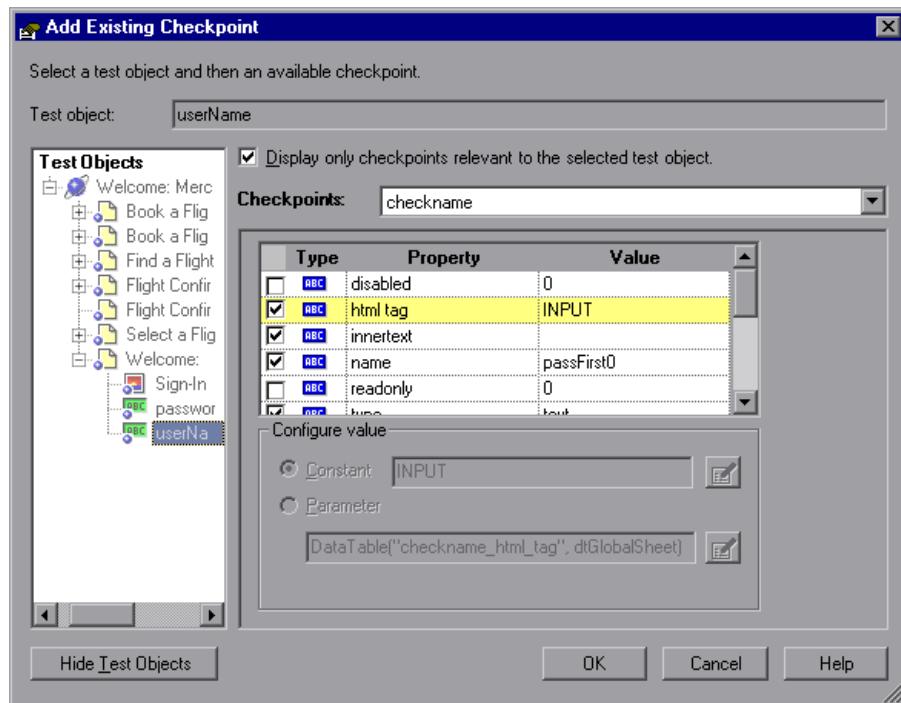
For more information on Expert View syntax, see "Basic VBScript Syntax" on page 970.

- 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*.

Reference

Add Existing Checkpoint Dialog Box

This dialog box enables you to add an existing checkpoint to your test while editing.



To access	Select Insert > Checkpoint > Existing Checkpoint .
Important information	<ul style="list-style-type: none"> ➤ 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.
See also	"Adding Existing Checkpoints to a Test" on page 592

User interface elements are described below (unlabeled elements are shown in angle brackets):

UI Elements	Description
Test object	The name of the test object for which you are adding a checkpoint.
Test Objects tree	All objects in the current action.
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, open your application and display the selected object to enable QuickTest to accurately determine all of the checkpoints that can apply to that object.</p>

UI Elements	Description
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>
<checkpoint details area>	Displays the settings of the selected checkpoint in read-only format.
Configure value	The value for the selected checkpoint in read-only mode. For more information, see Chapter 25, "Value Configuration and Regular Expressions".

Troubleshooting and Limitations - Creating Checkpoints

This section describes troubleshooting and limitations for creating checkpoints.

- Checkpoints that contain more than 64K of data may run slowly.

For more troubleshooting and limitations, see the section for the relevant add-in environment.

16

Standard Checkpoints

This chapter includes:

Concepts

- Standard Checkpoints Overview on page 606

Tasks

- How to Create or Modify a Standard Checkpoint Step on page 607

Reference

- Checkpoint Properties Dialog Box on page 609

- Image Checkpoint Properties Dialog Box on page 615

Concepts

Standard Checkpoints Overview

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 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.

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

Tasks

How to Create or Modify a Standard Checkpoint Step

If you are modifying an existing standard checkpoint step, see the "Checkpoint Properties Dialog Box" on page 609.

This task includes the following steps:

- "Prerequisites and considerations for inserting a new standard checkpoint step" on page 607
- "Insert a new standard checkpoint step" on page 607
- "Set the options for the standard checkpoint object" on page 608

Prerequisites and considerations for inserting a new standard checkpoint step

Before inserting a new standard checkpoint step, make sure to review all relevant information and perform all prerequisites:

Prerequisites	
Object visibility in application	During an editing session, make sure the object is visible in your application before inserting a standard checkpoint on it.
Considerations	
Availability	<ul style="list-style-type: none"> ➤ Recording sessions ➤ Editing sessions ➤ Active Screen
Supported environments	"Supported Checkpoints and Output Values Per Add-in" on page 1783

Insert a new standard checkpoint step

Insert a new checkpoint step, as described in "How to Insert a Checkpoint Step in a Test" on page 598. The Standard Checkpoint Properties dialog box opens.

Set the options for the standard checkpoint object

In the Standard Checkpoint Properties dialog box, specify the settings for the checkpoint object. For details, see "Checkpoint Properties Dialog Box" on page 609.

Note: If you are checking a Web image, you define the settings for the checkpoint in the Image Checkpoint Properties Dialog Box (described on page 615).

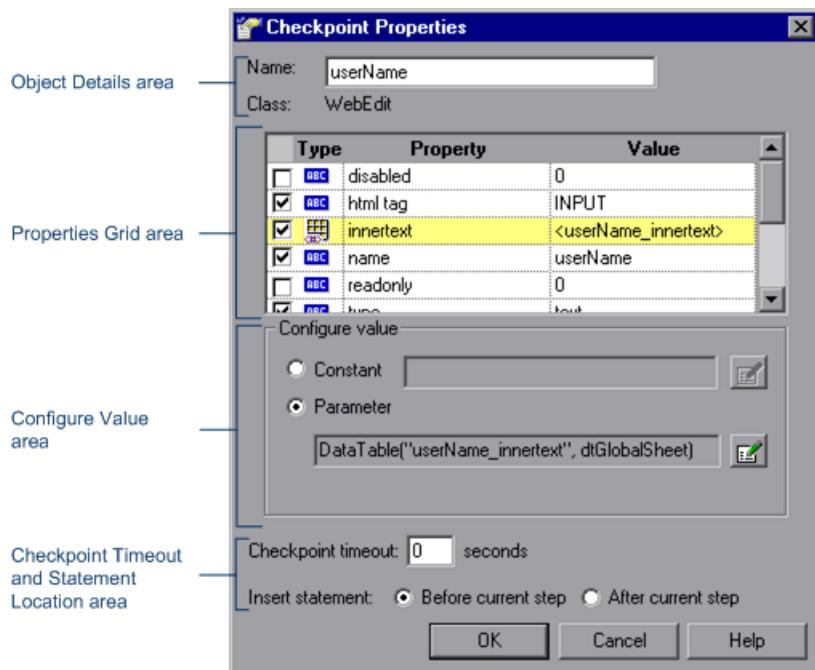
Reference

Checkpoint Properties Dialog Box

This dialog box enables you to specify which properties of the object to check, and edit the values of these properties.

The following image shows the Checkpoint Properties dialog box when a WebEdit object is selected. This image shows the dialog box that opens when adding a checkpoint to an existing test during an editing session.

The specific elements differ slightly depending on the type of object you are checking, and when adding a checkpoint during a recording session or editing an existing checkpoint.



To access	<p>Do one of the following:</p> <ul style="list-style-type: none"> ▶ Insert a new checkpoint step and select an object from your application. For details, see "How to Create or Modify a Standard Checkpoint Step" on page 607. ▶ In the Keyword View, right-click an existing checkpoint step and select Checkpoint Properties. ▶ In the local or shared object repository, click an existing checkpoint object. The Checkpoint details are displayed on the right side of the object repository dialog box, in the Object Details area.
Relevant tasks	"How to Create or Modify a Standard Checkpoint Step" on page 607
See also	"Checkpoint Types" on page 593

This dialog box contains the following key areas:

Object Details Area

The following image shows the Object Details area when a WebEdit object is selected for verification.



User interface elements are described below:

UI Element	Description
Name	<p>The name that QuickTest assigns to the checkpoint object. By default, the name is the same as the name of the object on which the checkpoint step is being performed. You can specify a different name for the checkpoint object or accept the default name.</p> <p>For a list of naming conventions, see "Naming Conventions" on page 1779.</p>

UI Element	Description
Class	The type of object (read-only).
	Find in Repository. Displays the checkpoint object in its object repository. Available only when editing an existing checkpoint step. It is not available when creating a new checkpoint step.

Properties Grid Area

The following image shows the Properties Grid area when a WebEdit object is selected for verification. Specific properties may vary depending on the type of object you are checking.

Type	Property	Value
<input type="checkbox"/> ABC	disabled	0
<input checked="" type="checkbox"/> ABC	html tag	INPUT
<input checked="" type="checkbox"/> ABC	innertext	<userName_innertext>
<input checked="" type="checkbox"/> ABC	name	userName
<input type="checkbox"/> ABC	readonly	0
<input checked="" type="checkbox"/> ABC	type	text

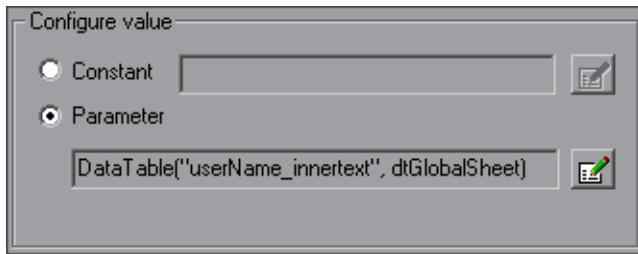
User interface elements are described below:

UI Elements	Description
Check box	<p>For each object class, QuickTest recommends default property checks. You can accept the default checks or modify them accordingly.</p> <p>To check a property, select the corresponding check box.</p> <p>To exclude a property check, clear the corresponding check box.</p>

UI Elements	Description
Type	<p>The  icon indicates that the value of the property is currently a constant.</p> <p>The  icon indicates that the value of the property is currently a test 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 "Configure Value Area" on page 867.

Configure Value Area

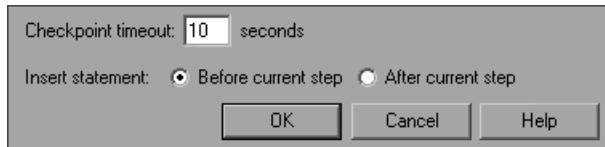
The following image shows the Configure Value area when a data table parameter is used in the checkpoint.



For a user interface description of this area, see "Configure Value Area" on page 867.

Checkpoint Timeout and Statement Location Area

The following image shows the timeout and statement location area when inserting a new checkpoint step during an editing session. When inserting a new checkpoint step during a recording session, the **Insert statement** option is not available.



User interface elements are described below:

UI Element	Description
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 it passes or until the timeout occurs. If the checkpoint does not pass before the timeout occurs, the checkpoint fails.</p> <p>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.</p> <p>If you specify a checkpoint timeout other than 0, and the checkpoint fails, the Run Results Viewer displays information on the checkpoint timeout.</p>
Insert statement	<p>Specifies whether to insert the checkpoint step before or after the currently selected step. The default value is Before current step.</p> <p>Note: Available only when inserting a new checkpoint step during an editing session. During a recording session, the checkpoint step is always inserted as the next step.</p>

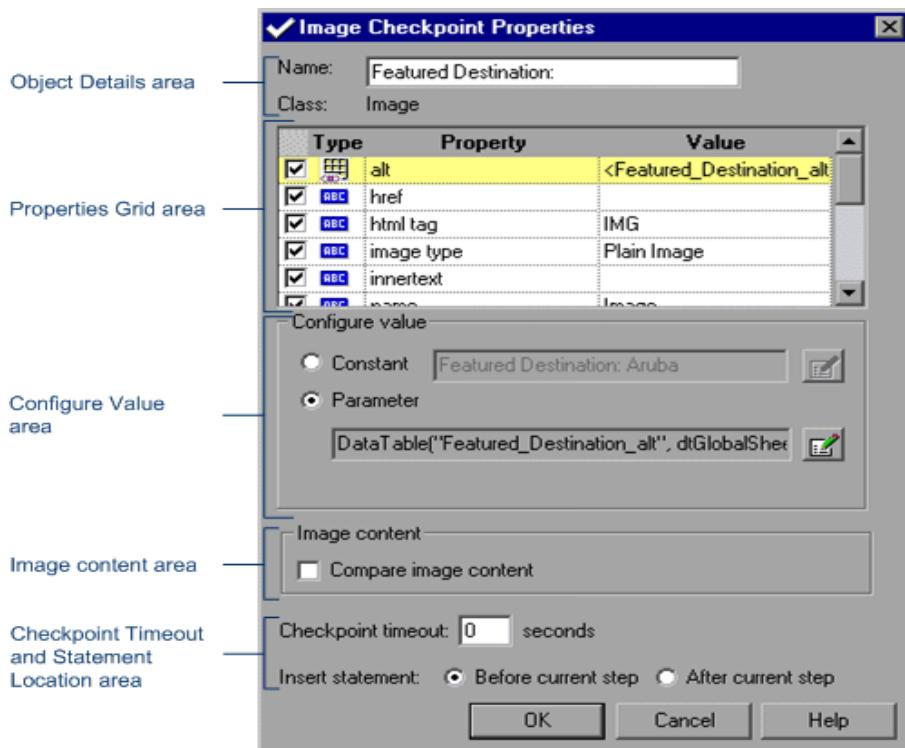
For a description of user interface elements, see "Configure Value Area" on page 867.

Image Checkpoint Properties Dialog Box

Note: This dialog box opens when checking only Web images, and it is almost identical to the Checkpoint Properties Dialog Box.

This dialog box enables you to specify which properties of the image to check and edit the values of those properties.

The following image shows an example of the Image Checkpoint Properties dialog box when an Image object is selected. This image shows the dialog box that opens when adding a checkpoint to an existing test during an editing session. The dialog box options differ slightly during a recording session or when editing an existing checkpoint.



To access	<p>Do one of the following:</p> <ul style="list-style-type: none"> ➤ Insert a new checkpoint step and select an object from your application. For details, see "How to Create or Modify a Standard Checkpoint Step" on page 607. ➤ In the Keyword View, right-click an existing checkpoint step and select Checkpoint Properties. ➤ In the local or shared object repository, click an existing checkpoint object. The Checkpoint details are displayed on the right side of the object repository dialog box, in the Object Details area.
Important information	This dialog box is similar to the standard Checkpoint Properties dialog box, except that it contains the Compare image content option.

This dialog box contains the following key elements:

Object Details Area

For a user interface description of this area, see "Object Details Area" on page 611.

Properties Area

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.

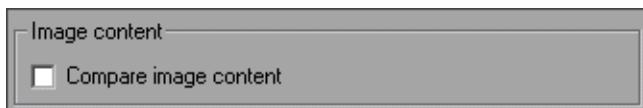
For a user interface description, see "Properties Grid Area" on page 612.

Configure Value Area

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 "Configure Value Area" on page 867.

Image Content Area



User interface elements are described below:

UI Element	Description
Compare image content	Enables you to compare the expected image source file with the actual image source file. If the expected and actual images are different, QuickTest displays them both in the Run Results. If the images are identical, only one graphic is displayed.

Checkpoint Timeout and Statement Location Area

For a user interface description of this area, see "Checkpoint Timeout and Statement Location Area" on page 613.

17

Bitmap Checkpoints

This chapter includes:

Concepts

- [Bitmap Checkpoints Overview on page 620](#)
- [Fine-Tuning the Bitmap Comparison on page 621](#)

Tasks

- [How to Create or Modify a Bitmap Checkpoint Step on page 624](#)

Reference

- [Bitmap Checkpoint Properties Dialog Box on page 626](#)

Concepts

Bitmap Checkpoints Overview

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.

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.

How QuickTest Checks Bitmaps

When you create a bitmap checkpoint, QuickTest captures the **visible** part of the specified object as a bitmap and inserts a checkpoint in the test. (QuickTest does not capture any part that is scrolled off the screen, or hidden by another object, for example.)

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 Run Results Viewer. In the Run Results Viewer 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 use the **Save still image captures to results** option in the **Run > Screen Capture** pane in the Options dialog box to configure when QuickTest saves the bitmaps in the run results. For details, see "Screen Capture Pane (Options Dialog Box)" on page 1450.

For details on run results of a checkpoint, see "Checkpoint and Output Value Results" on page 1174.

Tip: By default, QuickTest displays expected, actual, and difference bitmaps in the Run Results only for checkpoints that fail. If a bitmap checkpoint is configured such that the checkpoint may pass even if the expected and actual bitmaps are not identical, you might want to see the captured actual, expected and difference bitmaps in the run results even for bitmap checkpoints that pass. To do this, set the **Save still image captures to results** option to **Always**.



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 626) 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 can develop to run the bitmap comparison in the checkpoint according to a more specific algorithm. For details on using the **Comparer** option, see "Custom Comparers" on page 623.

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.

Custom Comparers

A custom comparer is a COM object that you or a third party can develop 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.

The **Comparer** 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 of when it can be useful to create a custom comparer, see "Custom Comparer for Images Whose Location Changes in the Application - Use-Case Scenario" on page 1821. For details on developing or installing custom comparers, see Appendix E, "Bitmap Checkpoint Customization."

If you select a custom comparer, some of the options in the Bitmap Checkpoint Properties dialog box are different. For details, see "Bitmap Checkpoint Properties Dialog Box" on page 626.

Tasks

How to Create or Modify a Bitmap Checkpoint Step

If you are modifying an existing bitmap checkpoint step proceed to the Set the options for the bitmap checkpoint object step.

This task includes the following steps:

- "Prerequisites and considerations for inserting a new bitmap checkpoint step" on page 624
- "Insert a new bitmap checkpoint step" on page 625
- "Set the options for the bitmap checkpoint object" on page 625

Prerequisites and considerations for inserting a new bitmap checkpoint step

Before inserting a new bitmap checkpoint step, make sure to review all relevant information and perform all prerequisites:

Prerequisites	
Object visibility in application	<ul style="list-style-type: none"> ➤ Make sure the object is fully visible in your application before inserting a bitmap checkpoint on it. If another application is overlapping the object, it is also captured. ➤ During a run session, 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 <code>MakeVisible</code> statement (for relevant environments) prior to your bitmap checkpoint step. For details on the <code>MakeVisible</code> method, see the <i>QuickTest Object Model Reference</i>.
Considerations	

Availability	► Recording sessions ► Active Screen
Supported environments	"Supported Checkpoints and Output Values Per Add-in" on page 1783
General considerations	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.

Insert a new bitmap checkpoint step

Insert a new bitmap checkpoint step, as described in "How to Insert a Checkpoint Step in a Test" on page 598. The Bitmap Checkpoint Properties dialog box opens.

Set the options for the bitmap checkpoint object

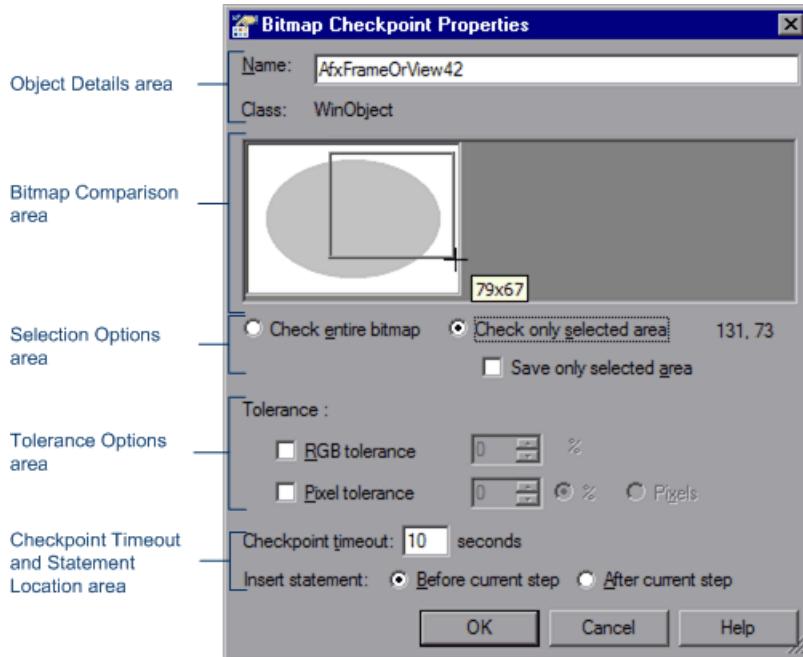
In the Bitmap Checkpoint Properties dialog box, specify the settings for the checkpoint object. For details, see "Bitmap Checkpoint Properties Dialog Box" on page 626.

Reference

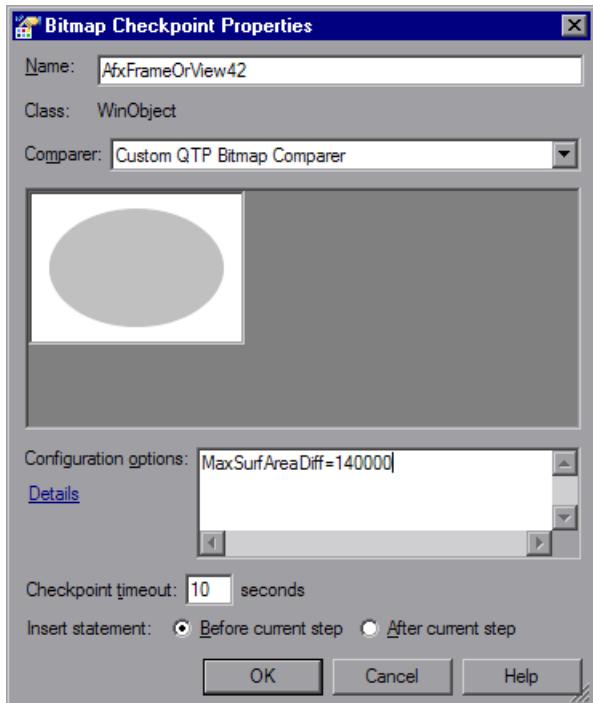
(Bitmap Checkpoint Properties Dialog Box)

This dialog box enables you to define and modify properties for a bitmap checkpoint.

The following image shows an example of the Bitmap Checkpoint Properties dialog box when inserting a new checkpoint during an editing session. The dialog box options differ slightly during a recording session or when editing an existing checkpoint. In addition, specific areas may vary depending on the type of comparison you are performing.



The following image shows the Bitmap Checkpoint Properties dialog box when a custom comparer is selected from the **Comparer** list. (The **Comparer** option is available because a custom comparer is installed and registered on the QuickTest computer.)

**To access**

Use one of the following:

- ▶ Insert a new checkpoint step and select a bitmap object from your application. For details, see "How to Create or Modify a Bitmap Checkpoint Step" on page 624.
- ▶ In the Keyword View, right-click an existing bitmap checkpoint step and select **Checkpoint Properties**.
- ▶ In the local or shared object repository, click an existing bitmap checkpoint object. The checkpoint details are displayed on the right side of the object repository dialog box, in the **Object Details** area.

Important information	<ul style="list-style-type: none"> ➤ If one or more custom comparers are installed and registered on the QuickTest computer, the Bitmap Checkpoint Properties dialog box includes a Comparer option. <p>A custom comparer is a COM object that you or a third party develop to run the bitmap comparison in the checkpoint according to a more specific algorithm.</p> <ul style="list-style-type: none"> ➤ If you select a custom comparer from the Comparer list, this dialog box may display different areas or options.
Relevant tasks	"How to Create or Modify a Bitmap Checkpoint Step" on page 624
See also	"Fine-Tuning the Bitmap Comparison" on page 621

Object Details Area

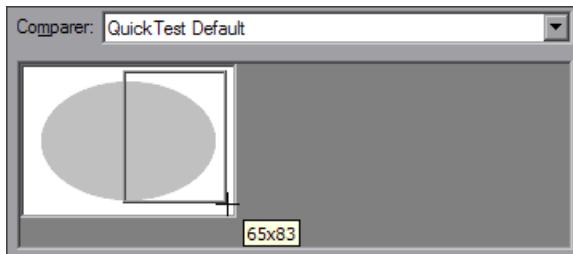


User interface elements are described below:

UI Element	Description
Name	The name that QuickTest assigns to the checkpoint object. By default, the name is the same as the name of the object on which the checkpoint step is being performed. You can specify a different name for the checkpoint object or accept the default name. For a list of naming conventions, see "Naming Conventions" on page 1779.
Class	The type of object (read-only).
	Find in Repository. Displays the checkpoint object in its object repository. Available only when editing an existing checkpoint step. It is not available when creating a new checkpoint step.

Bitmap Comparison Area

The following image shows an example of the Bitmap Comparison area when only a part of the bitmap is selected for verification. (The **Comparer** option is available because a custom comparer is installed and registered on the QuickTest computer.)



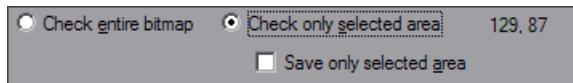
User interface elements are described below (unlabeled elements are shown in angle brackets):

UI Element	Description
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. Available only if one or more custom comparers are installed and registered on the QuickTest computer. For details, see "Fine-Tuning the Bitmap Comparison" on page 621.
<Bitmap display area>	Displays a bitmap of the object you selected.

Selection Options Area

Important information	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 Run Results Viewer.
------------------------------	---

The following image shows an example of the Selection Options area when only a part of the bitmap is selected for verification.



User interface elements are described below (unlabeled elements are shown in angle brackets):

UI Element	Description
Check entire bitmap / Check only selected area	<p>Enables you to specify whether the checkpoint compares the entire bitmap or only a specific area of the bitmap.</p> <p>Note: 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.</p>
Save only selected area	<p>Enables you to save only the selected area of the object with your test (to save disk space). Available only after you select Check only selected area and draw the rectangle that specifies the area.</p> <p>Note:</p> <ul style="list-style-type: none"> ▶ The bitmap stored in the checkpoint is cropped when you click OK. The Run Results Viewer displays only the selected area of the bitmap. ▶ 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.

UI Element	Description
<pointer position>	<p>Displays the coordinates of the pointer's current position (relative to the top left corner of the bitmap). Available only while the crosshairs pointer is visible.</p> <p>Note: As you draw the rectangle using the crosshairs, QuickTest displays a tooltip with the current selected area size near the crosshairs pointer.</p>

Tolerance Options Area

See also	"Fine-Tuning the Bitmap Comparison" on page 621
----------	---

The following image shows an example of the Tolerance Options area when no checkboxes are selected.



User interface elements are described below:

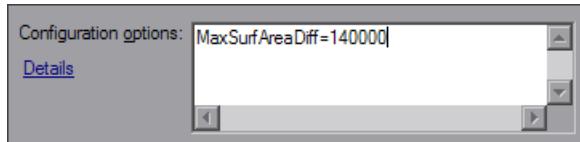
UI Element	Description
RGB tolerance	<p>Enables you to define the percent by which the RGB values of the pixels in the actual bitmap can differ from those of the expected bitmap and allow the checkpoint to pass. Available only for bitmaps with a color depth of 24 bits.</p> <p>You can modify the percentage manually or by using the up and down arrows.</p>

UI Element	Description
Pixel tolerance	<p>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.</p> <p>You can 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.)</p>

Custom Comparer Area

Important information	<p>If you select a custom comparer to run the bitmap comparison, the options in this area are displayed instead of the options for selecting an area of the bitmap and for setting tolerance levels.</p> <p>Available only if one or more custom comparers are installed and registered on the QuickTest computer. For details, see "Fine-Tuning the Bitmap Comparison" on page 621.</p>
See also	"Fine-Tuning the Bitmap Comparison" on page 621

The following image shows an example of the Custom Comparer area when a custom comparer is selected in the **Comparer** list.



User interface elements are described below:

UI Element	Description
Configuration options	Enables you to provide input (in string format) to the custom comparer, for any configuration options it supports. By default, this box displays a configuration string provided by the custom comparer (if available). Example: You might be able to specify tolerance levels, an acceptable deviation in size or location of the bitmap, and so on.
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.

Checkpoint Timeout and Statement Location Area

The following image shows the Checkpoint Timeout and Statement Location area when inserting a new checkpoint step during an editing session. When inserting a new checkpoint step during a recording session, the **Insert statement** option is not available.



A screenshot showing the 'Checkpoint timeout' and 'Statement location' settings. The 'Checkpoint timeout' field contains '0' seconds. The 'Insert statement' section shows two radio button options: 'Before current step' (selected) and 'After current step'.

Checkpoint timeout:	0	seconds
Insert statement:	<input checked="" type="radio"/> Before current step	<input type="radio"/> After current step

User interface elements are described below:

UI Element	Description
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 it passes or until the timeout occurs. If the checkpoint does not pass before the timeout occurs, the checkpoint fails.</p> <p>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.</p> <p>If you specify a checkpoint timeout other than 0, and the checkpoint fails, the Run Results Viewer displays information on the checkpoint timeout.</p>
Insert statement	<p>Specifies whether to insert the checkpoint step before or after the currently selected step. The default value is Before current step.</p> <p>Note: Available only when inserting a new checkpoint step during an editing session. During a recording session, the checkpoint step is always inserted as the next step.</p>

18

Table Checkpoints

This chapter includes:

Concepts

- ▶ [Table Checkpoints Overview on page 636](#)

Tasks

- ▶ [How to Create or Modify a Table Checkpoint Step on page 637](#)

Reference

- ▶ [Table Checkpoint Properties Dialog Box \(Table Content Tab\) on page 639](#)
- ▶ [Table Checkpoint Properties Dialog Box \(Properties Tab\) on page 649](#)
- ▶ [Define/Modify Row Range Dialog Box on page 652](#)

Concepts

Table Checkpoints Overview

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 property values 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 Run Results Viewer. For details, see Chapter 31, "Run Results Viewer."

Different environments support different checkpoints. For details on supported checkpoints, see "Supported Checkpoints" on page 1784.

Row Range Selection

The tables in your application may be very large. A table checkpoint on a large table may take a long time to create and a long time to run. You can choose to include all rows in your table checkpoint or you can specify a smaller row range.

For some QuickTest add-ins, when creating a new table checkpoint object, you can specify the range of rows you want to include using the "Define/Modify Row Range Dialog Box" on page 652.

Tasks

How to Create or Modify a Table Checkpoint Step

This task includes the following steps:

- "Prerequisites and considerations for inserting a new table checkpoint step" on page 637
- "Insert a new table checkpoint step" on page 637
- "Set the options for the table checkpoint object" on page 638

Prerequisites and considerations for inserting a new table checkpoint step

Before inserting a new table checkpoint step, make sure to review all relevant information and perform all prerequisites:

Prerequisites	
Object visibility in application	During an editing session, make sure the object is visible in your application before inserting a table checkpoint step on it.
Considerations	
Availability	<ul style="list-style-type: none"> ➤ Recording sessions ➤ Editing sessions ➤ Active Screen
Supported environments	"Supported Checkpoints and Output Values Per Add-in" on page 1783

Insert a new table checkpoint step

- 1 Insert a new table checkpoint step, as described in "How to Insert a Checkpoint Step in a Test" on page 598.
- 2 (Optional) If the Object Selection dialog box opens, select a table object from the displayed object tree and click **OK**. For details, see "Object Selection Dialog Box" on page 155.

- 3** (Optional) For certain objects in certain environments, before the Table Output Value Properties dialog box opens, the Define Row Range dialog box opens. Select the row range to check, as described in "Define/Modify Row Range Dialog Box" on page 652.

Set the options for the table checkpoint object

- In the Table Checkpoint Properties dialog box, specify the settings for the checkpoint object. For details, see "Table Checkpoint Properties Dialog Box (Table Content Tab)" on page 639.
- (Optional) Define cell selection in the Grid area of the Table Checkpoint Properties dialog box, as follows:

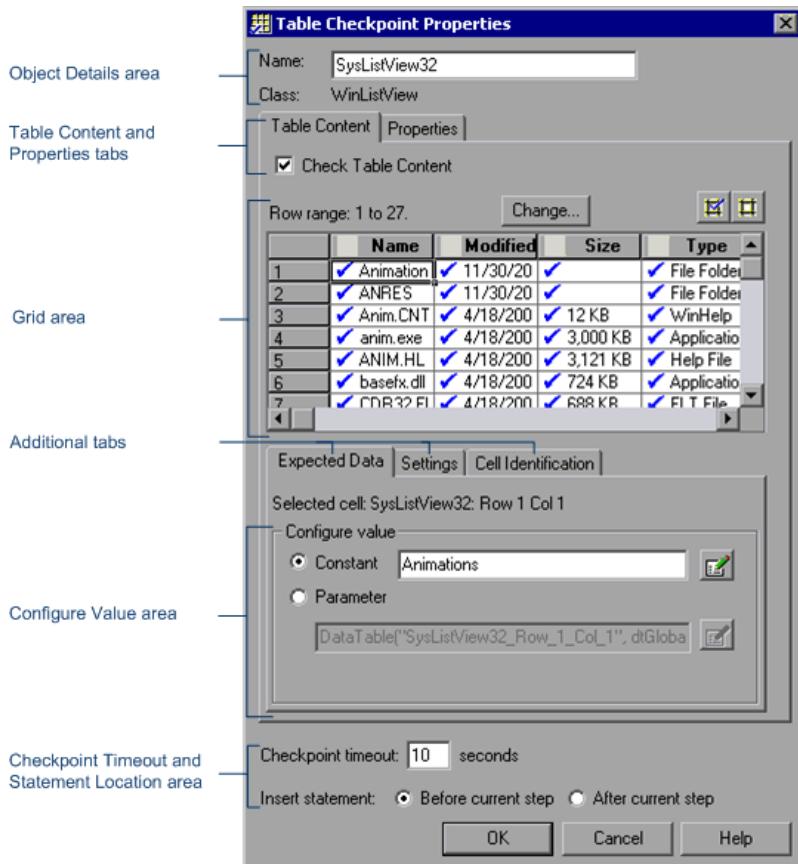
To:	Do this:
Add a single cell to or remove it from the check	Double-click the cell
Add an entire row to or remove it from the check	Double-click the row header
Add an entire column to or remove it from the check	Double-click the column header
Add all cells to or remove all cells from the check	Double-click 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 

Reference

Table Checkpoint Properties Dialog Box (Table Content Tab)

This dialog box enables you to define and modify content for a table checkpoint object. In some environments, this dialog box also enables you to check the properties of the object.

The following image shows an example of the Table Checkpoint Properties dialog box when a WinListView object is selected for verification.

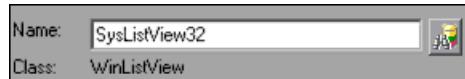


The image above shows the dialog box that opens when adding a table checkpoint to an existing test during an editing session. The dialog box options differ slightly during a recording session or when editing an existing checkpoint.

To access	Do one of the following: <ul style="list-style-type: none"> ➤ Insert a new checkpoint step and select a table object from your application. For details, see "How to Create or Modify a Table Checkpoint Step" on page 637. ➤ In the Keyword View, right-click an existing checkpoint step and select Checkpoint Properties. ➤ In the local or shared object repository, click an existing checkpoint object. The Checkpoint details are displayed on the right side of the object repository dialog box, in the Object Details area.
Important information	<ul style="list-style-type: none"> ➤ Most of the elements described in this section are available regardless of whether the Table Checkpoint Properties dialog box contains tabs. ➤ If the Table Checkpoint Properties dialog box contains tabs, you use the Table Content tab to check table content.
Relevant tasks	"How to Create or Modify a Table Checkpoint Step" on page 637
See also	<ul style="list-style-type: none"> ➤ "Table Checkpoints Overview" on page 636 ➤ "Table Checkpoint Properties Dialog Box (Properties Tab)" on page 649

Object Details Area

The following image shows the Object Details area when a ListView object is selected during an editing session.



User interface elements are described below:

UI Elements	Description
Name	<p>The name that QuickTest assigns to the checkpoint object. By default, the name is the same as the name of the object on which the checkpoint step is being performed. You can specify a different name for the checkpoint object or accept the default name.</p> <p>For a list of naming conventions, see "Naming Conventions" on page 1779.</p>
Class	The type of object (read-only).
	<p>Find in Repository. Displays the checkpoint object in its object repository. Available only when editing an existing checkpoint step. It is not available when creating a new checkpoint step.</p>

Table Content and Properties Tabs

Important information	<ul style="list-style-type: none"> ► The 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.
------------------------------	--

The following image shows the Table Content and Properties tabs area when each tab is selected.



User interface elements are described below:

UI Elements	Description
Table Content	<p>Displays the table content areas, which include:</p> <ul style="list-style-type: none"> ➤ "Grid Area" on page 642 ➤ "Expected Data Tab" on page 643 ➤ "Settings Tab" on page 644 ➤ "Cell Identification Tab" on page 646
Check Table Content	<p>Instructs QuickTest to check the content of the table object. (Selected by default.)</p>
Properties	<p>Displays the object properties. For details, see "Table Checkpoint Properties Dialog Box (Properties Tab)" on page 649.</p>
Check Properties	<p>Instructs QuickTest to check the properties of the table object. (Cleared by default.)</p>

Grid Area

Important information	<ul style="list-style-type: none"> ➤ The column header names are captured from the table you selected for your checkpoint. ➤ 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. ➤ You can change the column widths and row heights of the grid by dragging the column and row header dividers. ➤ If row range selection is supported, the row range you specify when creating the checkpoint is displayed above the grid
See also	"Considerations for Modifying Row Ranges" on page 653

The following image shows the Grid area when all cells are marked for verification.

	Name	Modified	Size	Type
1	Animation	✓ 11/30/20	✓	✓ File Folder
2	ANRES	✓ 11/30/20	✓	✓ File Folder
3	Anim.CNT	✓ 4/18/200	✓ 12 KB	✓ WinHelp
4	anim.exe	✓ 4/18/200	✓ 3,000 KB	✓ Application
5	ANIM.HL	✓ 4/18/200	✓ 3,121 KB	✓ Help File
6	basefx.dll	✓ 4/18/200	✓ 724 KB	✓ Application
7	FDR32.FI	✓ 4/18/200	✓ 688 KB	✓ F1 T File

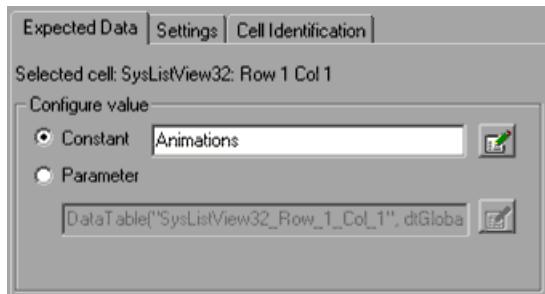
User interface elements are described below:

<grid area>	The grid area displays the captured/expected values of all the cells in the table. Only the ones with blue check marks are used (checked) by the checkpoint. You can instruct QuickTest to check the entire table, specific rows, specific columns, or specific cells. Note: QuickTest checks only cells containing a check mark.
Change	Enables you to define or modify the row range to check by opening the "Define/Modify Row Range Dialog Box" on page 652.
	Add/Remove from check. Add or remove the selected cells from the check.

Expected Data Tab

Important information	When more than one cell is selected (highlighted) in the grid area, the options in the Expected Data tab are disabled.
------------------------------	--

The following image shows the Expected Data area when the constant value Animations is displayed.



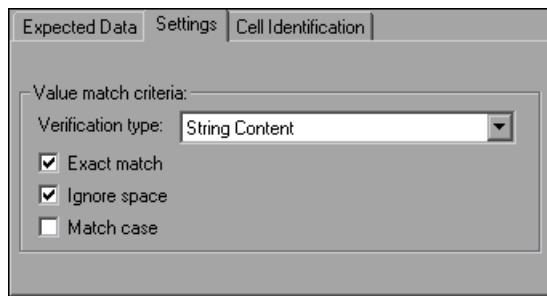
User interface elements are described below:

UI Elements	Description
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 details on modifying values, see "Configure Value Area" on page 867.

Settings Tab

Important information	<ul style="list-style-type: none">► The settings in this tab apply to all cells marked for verification.► The default setting is to treat cell values as strings and to check for the exact text, while ignoring spaces.
------------------------------	---

The following image shows the Settings area when the default values are selected.



User interface elements are described below:

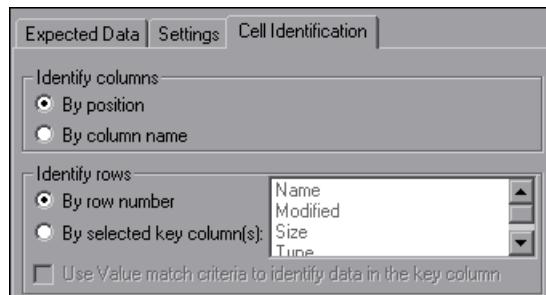
Option	Description
Verification type	<p>Specifies how cell contents are compared:</p> <ul style="list-style-type: none"> ► String Content. (Default) Evaluates the content of the cell as a string. For example, 2 and 2.00 are not recognized as the same string. ► Numeric Content. Evaluates the content of 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.
Exact match	<p>(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.</p> <p>Available only when String Content is selected as the Verification type.</p>

Option	Description
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. Available only when String Content is selected as the Verification type .
Match case	Conducts a case sensitive search. Available 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. Available only when Numeric Range is selected as the Verification type .

Cell Identification Tab

Important information	The settings in this tab apply to all cells marked for verification.
------------------------------	--

The following image shows the Cell Identification area when the default settings are displayed.



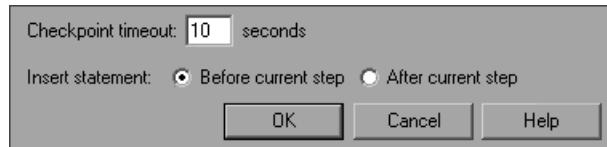
User interface elements are described below:

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>

Timeout and Statement Location Area

Important information	The Insert statement option is not available when inserting a new checkpoint step during a recording session.
------------------------------	--

The following image shows the Timeout and Statement Location area when inserting a new checkpoint step during an editing session. When inserting a new checkpoint step during a recording session, the **Insert statement** option is not available.



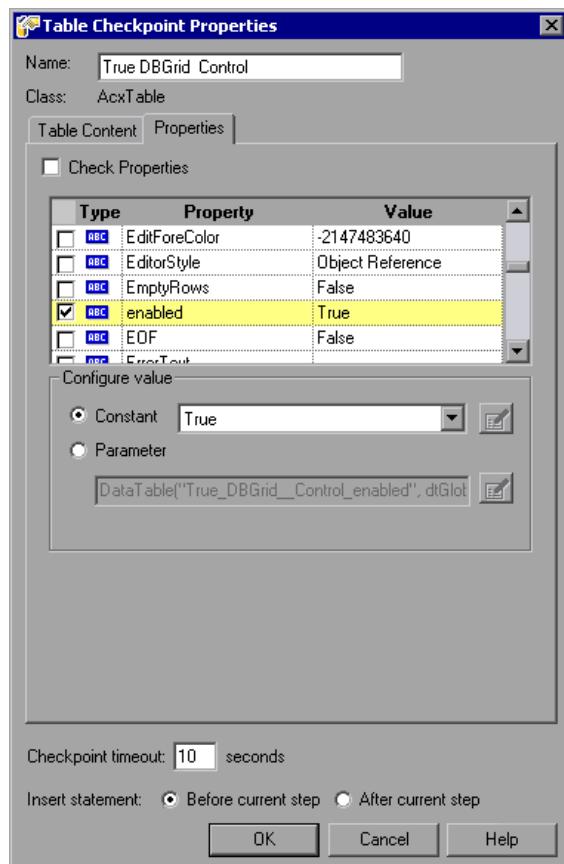
User interface elements are described below:

UI Element	Description
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 it passes or until the timeout occurs. If the checkpoint does not pass before the timeout occurs, the checkpoint fails.</p> <p>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.</p> <p>If you specify a checkpoint timeout other than 0, and the checkpoint fails, the Run Results Viewer displays information on the checkpoint timeout.</p>
Insert statement	<p>Specifies whether to insert the checkpoint step before or after the currently selected step. The default value is Before current step.</p> <p>Note: Available only when inserting a new checkpoint step during an editing session. During a recording session, the checkpoint step is always inserted as the next step.</p>

Table Checkpoint Properties Dialog Box (Properties Tab)

This tab enables you to specify which table (or grid) properties you want to check.

The following image shows an example of the Properties tab of the table checkpoint properties dialog box when a table object is selected for verification.



To access	<p>1 Open the Table Checkpoint Properties dialog box by using one of the following:</p> <ul style="list-style-type: none"> ➤ Insert a new checkpoint step and select a table object from your application. For details, see "How to Create or Modify a Table Checkpoint Step" on page 637. ➤ In the Keyword View or Expert View, right-click an existing checkpoint step and select Checkpoint Properties. ➤ In the local or shared object repository, click an existing checkpoint object. The Checkpoint details are displayed on the right side of the object repository dialog box, in the Object Details area. <p>2 In the Table Checkpoint Properties dialog box, select the Properties tab.</p>
Important information	<ul style="list-style-type: none"> ➤ This tab is available only for certain objects in certain environments. ➤ Selecting the Check Properties check box instructs QuickTest to check the properties of the table object. (Cleared by default.) ➤ 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. ➤ For details on general table checkpoint options located outside the Properties tab, such as Name and Checkpoint timeout, see "Table Checkpoint Properties Dialog Box (Table Content Tab)" on page 639.
Relevant tasks	<p>"How to Create or Modify a Table Checkpoint Step" on page 637</p>
See also	<ul style="list-style-type: none"> ➤ "Table Checkpoints Overview" on page 636 ➤ "Table Checkpoint Properties Dialog Box (Table Content Tab)" on page 639

The Properties tab contains the following key areas:

Object Details Area

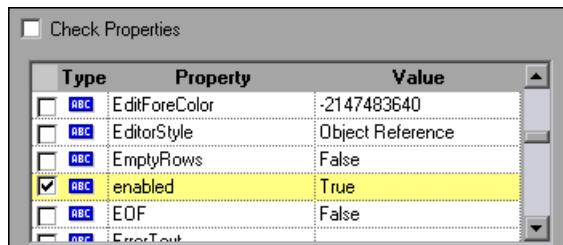
For a user interface description of this area, see "Table Checkpoint Properties Dialog Box (Table Content Tab)" on page 639.

Table Content and Properties Tabs

For a user interface description of this area, see "Table Checkpoint Properties Dialog Box (Table Content Tab)" on page 639.

Properties Grid Area

The Properties grid displays the table object's default properties, including the properties, their values, and their types.



For user interface description, see "Properties Grid Area" on page 612.

Configure Value Area

The **Configure value** area enables you to define the expected value of the property as a **Constant** or a **Parameter**.



For user interface description, see "Configure Value Area" on page 867.

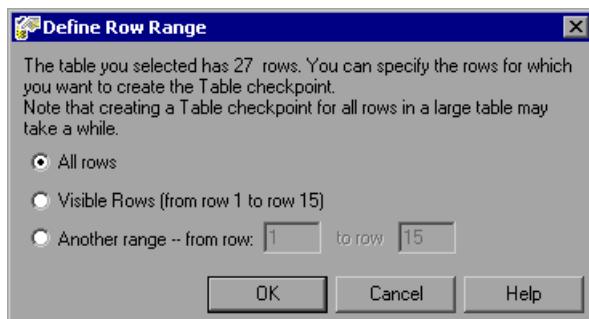
Checkpoint Timeout and Statement Location Area

For a user interface description of this area, see "Table Checkpoint Properties Dialog Box (Table Content Tab)" on page 639.

Define/Modify Row Range Dialog Box

This dialog box enables you to define or modify the number of rows included in an existing table or database checkpoint.

The following image shows this dialog box as it appears for a table checkpoint in define mode. Depending on the mode of the dialog box, the actual title bar of the image may differ. This dialog box may also differ when viewed for a table output value object.



To access	In the Table Checkpoint Properties dialog box or Table Output Value Properties dialog box, click the Change button.
Important information	This dialog box is also available for certain list view objects. This enables you to define a table checkpoint on specific rows within a list view object.
Relevant tasks	"How to Create or Modify a Table Checkpoint Step" on page 637
See also	<ul style="list-style-type: none"> ➤ "Table Checkpoint Properties Dialog Box (Table Content Tab)" on page 639 ➤ "Considerations for Modifying Row Ranges" on page 653 ➤ "How to Create or Modify a Table Output Value Step" on page 793

User interface elements are described below:

UI Elements	Description
All rows	<p>Includes all rows in the checkpoint or output value.</p> <p>Note: Capturing all of the data for large table or list view objects may take some time.</p>
Visible Rows	<p>Includes only the rows that are currently visible in your application in the checkpoint or output value.</p> <p>Available only for some environments or object types.</p>
Another range	<p>Includes the rows you specify in the checkpoint or output value.</p> <p>You can specify any row range between 1 and the number of rows listed in the dialog box.</p>

Considerations for Modifying Row Ranges

- 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 details, see "Update Options Tab (Update Run Dialog Box)" on page 1271.

- 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.

19

Text Checkpoints

This chapter includes:

Concepts

- Checking Text Overview on page 656

Tasks

- How to Create or Modify a Text or Text Area Checkpoint Step on page 657

Reference

- Text / Text Area Checkpoint Properties Dialog Box on page 660

Concepts

Checking Text Overview

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 details on standard checkpoints, see "How to Create or Modify a Standard Checkpoint Step" on page 607.
- **Text Area Checkpoint.** Enables you to check that a text string appears within a defined area in a Windows application, according to specified criteria. 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 (described on page 660) to configure the relative position of the Checked Text within the captured area. When QuickTest runs the test, it checks for the selected text within the defined area, according to the settings you configured.
- **Text Checkpoint.** Enables you to check that the text is displayed in a screen, window, or Web page, according to specified criteria. 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.

By default, 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.

For task details, see "How to Create or Modify a Text or Text Area Checkpoint Step" on page 657.

Tasks

How to Create or Modify a Text or Text Area Checkpoint Step

If you are modifying an existing text or text area checkpoint step proceed to Set the options for the text or text area checkpoint object.

This task includes the following steps:

- "Prerequisites and considerations for inserting a new text or text area checkpoint step" on page 657
- "Insert a new text or text area checkpoint step" on page 658
- "Set the options for the text or text area checkpoint object" on page 659

Prerequisites and considerations for inserting a new text or text area checkpoint step

Before inserting a new text checkpoint step, make sure to review all relevant information and perform all prerequisites:

Prerequisites	
Object visibility in application	During an editing session, make sure the object is visible in your application before inserting a text checkpoint on it.
Considerations	
Availability	<ul style="list-style-type: none"> ➤ Recording sessions ➤ Editing sessions ➤ Active Screen
Supported environments	"Supported Checkpoints and Output Values Per Add-in" on page 1783

Text recognition	Before you create a text or text area checkpoint for a Windows-based application, make sure you configure the required capture settings in the General > Text Recognition pane (Tools > Options > Text Recognition node). For more information, see "Text Recognition Pane (Options Dialog Box)" on page 1426 and "Text Recognition for Windows-Based Objects Overview" on page 848.
Checkpoints on the Text property	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. For details, see "How to Create or Modify a Standard Checkpoint Step" on page 607.

Insert a new text or text area checkpoint step

Insert a new text checkpoint step, as described in "How to Insert a Checkpoint Step in a Test" on page 598. The Text / Text Area Checkpoint Properties Dialog Box opens (as described on page 660).

- To create a text checkpoint while editing, you first highlight a text string in the Active Screen then right-click the string, and select **Insert Text Checkpoint**.
- When you create a text area checkpoint, you first define the area containing the text you want QuickTest to check. When you select Text Area Checkpoint, the mouse turns into a crosshairs pointer. Click and drag the crosshairs pointer to define this area. Release the mouse button after outlining the area required. For more details, see the section on text area checkpoints in "Checking Text Overview" on page 656, and the important information in "Text / Text Area Checkpoint Properties Dialog Box" on page 660.

Tip: Hold down the left mouse button and use the arrow keys to make precise adjustments to the defined area.

Set the options for the text or text area checkpoint object

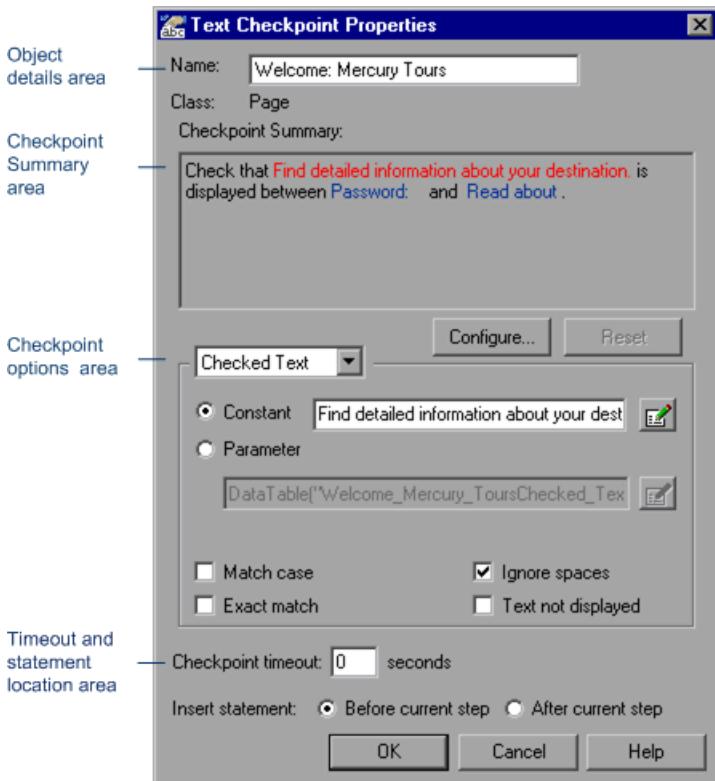
In the Text / Text Area Checkpoint Properties Dialog Box, specify the settings for the text checkpoint object. For details, see "Text / Text Area Checkpoint Properties Dialog Box" on page 660.

Reference

Text / Text Area Checkpoint Properties Dialog Box

This dialog box enables you to specify the text to be checked, as well as specify which text is displayed before and after the checked text.

These configuration options are particularly helpful when the text string you want to check appears several times or when it could change in a predictable way during run sessions.

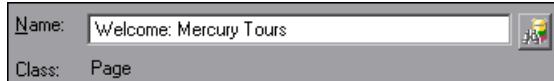


The image above shows an example of the Text Checkpoint Properties dialog box when adding a text checkpoint to an existing test during an editing session. The Text Checkpoint Properties dialog box options differ slightly during a recording session or when editing an existing checkpoint. The Text Area Checkpoint Properties dialog box is similar to the Text Checkpoint Properties dialog box.

To access	<p>Use one of the following:</p> <ul style="list-style-type: none"> ▶ Insert a new checkpoint step and select an object or text area from your application. For details, see "How to Create or Modify a Text or Text Area Checkpoint Step" on page 657. ▶ In the Keyword View, right-click an existing checkpoint step and select Checkpoint Properties. ▶ In the local or shared object repository, click an existing checkpoint object. The Checkpoint details are displayed on the right side of the object repository dialog box, in the Object Details area.
Important information	<p>Consider the following when defining the area for a text area checkpoint:</p> <ul style="list-style-type: none"> ▶ 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.
Relevant tasks	"How to Create or Modify a Text or Text Area Checkpoint Step" on page 657
See also	"Checking Text Overview" on page 656

This dialog box contains the following key areas:

Object Details Area

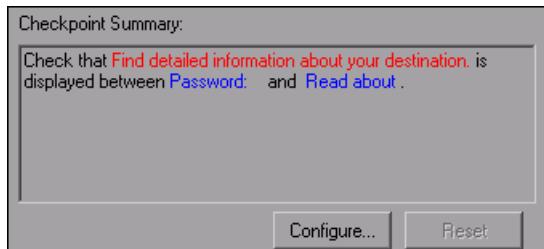


User interface elements are described below:

UI Element	Description
Name	The name that QuickTest assigns to the checkpoint object. By default, the name is the same as the name of the object on which the checkpoint step is being performed. You can specify a different name for the checkpoint object or accept the default name. For a list of naming conventions, see "Naming Conventions" on page 1779.
Class	The type of object (read-only).
	Find in Repository. Displays the checkpoint object in its object repository. Available only when editing an existing checkpoint step. It is not available when creating a new checkpoint step.

Checkpoint Summary Area

The following image shows an example of the Checkpoint Summary area.

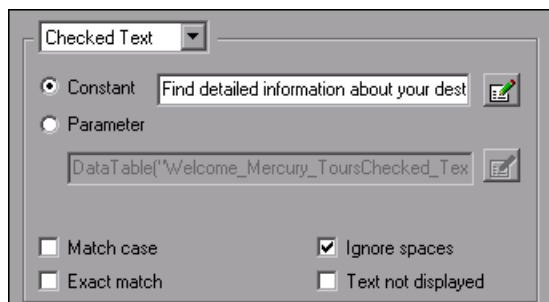


User interface elements are described below:

UI Element	Description
Checkpoint Summary	<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 click Configure to view and manipulate the actual selected text for the checkpoint.</p>
Configure	<p>Opens the Configure Text Selection Dialog Box (described on page 669), where you can specify the checked text, the text before (if any), and the text after (if any).</p>
Reset	<p>Resets the text selection to the previous configuration.</p>

Checkpoint Options Area

The following image shows an example of the checkpoint options area with **Checked Text** selected in the drop-down list.



User interface elements are described below:

UI Element	Description
<Text Area list>	<p>The checkpoint options area contains a drop-down box with Checked Text as the default selection.</p> <p>Possible options:</p> <ul style="list-style-type: none"> ➤ Checked Text. For details, see Checked Text Options. ➤ Text Before. For details, see "Text Before Option Area (Text / Text Area Checkpoint Properties Dialog Box)" on page 666. ➤ Text After. For details, see "Text After Option Area (Text / Text Area Checkpoint Properties Dialog Box)" on page 668. <p>You can set parameterization and other preferences for each of the string elements in your checkpoint by selecting one of these string element types from the drop-down and selecting your preferences.</p>

Checked Text Options

UI Element	Description
Constant	<p>(Default) Sets the expected value of the checked text as a constant. For information on modifying values, see "Configure Value Area" on page 867.</p> <p>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.</p>
Parameter	<p>Sets the expected value of the checked text as a parameter. For information on modifying values, see "Configure Value Area" on page 867.</p>
Match case	<p>Conducts a case-sensitive check.</p>
Exact match	<p>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.</p>

UI Element	Description
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

Timeout and Statement Location Area

Checkpoint timeout:	<input type="text" value="0"/> seconds
Insert statement:	<input checked="" type="radio"/> Before current step <input type="radio"/> After current step

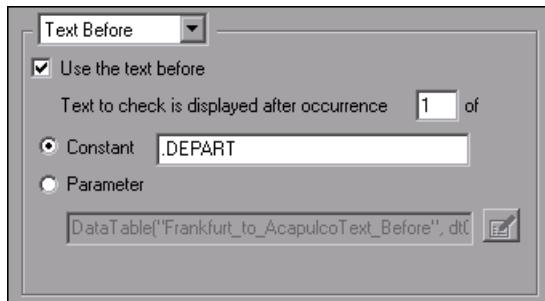
User interface elements are described below:

UI Element	Description
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 it passes or until the timeout occurs. If the checkpoint does not pass before the timeout occurs, the checkpoint fails.</p> <p>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.</p> <p>If you specify a checkpoint timeout other than 0, and the checkpoint fails, the Run Results Viewer displays information on the checkpoint timeout.</p>

UI Element	Description
Insert statement	Specifies whether to insert the checkpoint step before or after the currently selected step. The default value is Before current step . Note: Available only when inserting a new checkpoint step during an editing session. During a recording session, the checkpoint step is always inserted as the next step.

Text Before Option Area (Text / Text Area Checkpoint Properties Dialog Box)

The following image shows an example of the checkpoint options area with **Text Before** selected in the drop-down list.



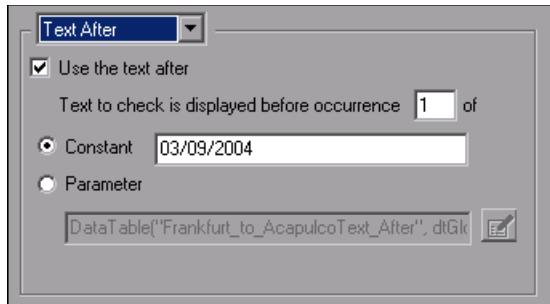
User interface elements are described below:

UI Element	Description
Use the text before	Checks the text before the checked text. To ignore this text, clear this check box.

UI Element	Description
Text to check is displayed after occurrence _ of	<p>Checks that the checked text is displayed after the specified text.</p> <p>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.</p> <p>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.</p> <p>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.</p>
Constant	<p>(Default) Sets the expected value of the text before the checked text as a constant. For information on modifying values, see "Configure Value Area" on page 867.</p> <p>If you modify the text, whenever possible, use a string that is unique within the object so that the occurrence number is 1.</p> <p>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" on page 669.</p>
Parameter	<p>Sets the expected value of the text before the checked text as a parameter. For information on modifying values, see "Configure Value Area" on page 867.</p>

Text After Option Area (Text / Text Area Checkpoint Properties Dialog Box)

The following image shows an example of the checkpoint options area with **Text After** selected in the drop-down list.



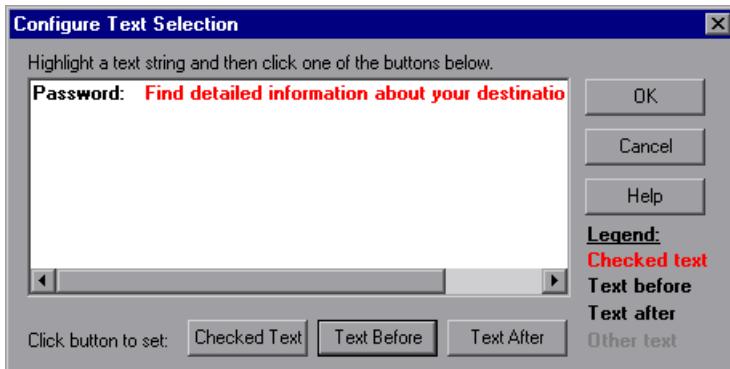
User interface elements are described below:

UI Element	Description
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 of	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.

UI Element	Description
Constant	<p>(Default) Sets the expected value of the text after the checked text as a constant. For information on modifying values, see "Configure Value Area" on page 867.</p> <p>If you modify the text, whenever possible, use a string that is unique within the object so that the occurrence number is 1.</p> <p>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" on page 669.</p>
Parameter	<p>Sets the expected value of the text after the checked text as a parameter. For information on modifying values, see "Configure Value Area" on page 867.</p>

Configure Text Selection Dialog Box

This dialog box enables you to specify the checked text, the text before, and the text after, for a text checkpoint.



To access	In the Text / Text Area Checkpoint Properties Dialog Box (described on page 660), click the Configure button in the Checkpoint Summary area.
------------------	--

Important information	<ul style="list-style-type: none"> ➤ 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 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. <p>Example: In the sample image above, suppose 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:</p> <ul style="list-style-type: none"> ➤ 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.
Relevant tasks	"How to Create or Modify a Text or Text Area Checkpoint Step" on page 657

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:

UI Elements	Description
Checked Text	Sets the highlighted text as the checked text. QuickTest displays this text in red and the remainder in black.

UI Elements	Description
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.

20

Database Checkpoints

This chapter includes:

Concepts

- Database Checkpoints Overview on page 674

Tasks

- How to Create or Modify a Database Checkpoint Step on page 676

Reference

- Database Checkpoint Properties Dialog Box on page 679
- Connect to Database Using ODBC Page (Database Query Wizard) on page 689
- Specify SQL Statement Page (Database Query Wizard) on page 691

Troubleshooting and Limitations - Database Checkpoints on page 692

Concepts

Database Checkpoints Overview

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.

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.CheckCheckPoint` statement and as a step in the Keyword View.

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

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.

You can also specify the way QuickTest identifies the selected cells. 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**.

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.

Note:

- 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.
 - You can view the results of the checkpoint in the Run Results Viewer. For details, see Chapter 31, "Run Results Viewer."
 - For details on creating database checkpoints, see "How to Create or Modify a Database Checkpoint Step" on page 676.
-

Tasks

How to Create or Modify a Database Checkpoint Step

If you are modifying an existing database checkpoint step proceed to the Set the options for the database checkpoint object step.

This task includes the following steps:

- "Prerequisites and considerations for inserting a new database checkpoint step" on page 677
- "Insert a new database checkpoint step" on page 677
- "Set the options for the database checkpoint object" on page 678

Prerequisites and considerations for inserting a new database checkpoint step

Before inserting a new database checkpoint step, make sure to review all relevant information and perform all prerequisites:

Prerequisites	
Object visibility in application	During an editing session, make sure the object is visible in your application before inserting a database checkpoint on it.
Considerations	
Availability	<ul style="list-style-type: none"> ▶ Recording sessions ▶ Editing sessions ▶ Active Screen
Supported environments	"Supported Checkpoints and Output Values Per Add-in" on page 1783

Insert a new database checkpoint step

- 1 Insert a new checkpoint step, as described in "How to Insert a Checkpoint Step in a Test" on page 598.
- 2 In the Database Query Wizard, define the query for your checkpoint using Microsoft Query or by manually entering a database connection and SQL statement. For details, see "Connect to Database Using ODBC Page (Database Query Wizard)" on page 689.
- 3 If you selected Microsoft Query as your data source, Microsoft Query opens, enabling you to define a query. When you are done, in the Finish page of the Query Wizard, use one of the following:
 - ▶ **Exit and return to QuickTest Professional.** Exits Microsoft Query.
 - ▶ **View data or edit query in Microsoft Query.** View or edit the query prior to exiting Microsoft Query.
 For details on working with Microsoft Query, see your Microsoft Query documentation.

- 4 If you selected **Specify SQL statement manually**, the Specify SQL statement page opens, enabling you to specify the connection string and the SQL statement. For details, see "Specify SQL Statement Page (Database Query Wizard)" on page 691.
- 5 After defining the query, the Database Checkpoint Properties dialog box opens, enabling you to specify the settings for the checkpoint object. For details, see "Database Checkpoint Properties Dialog Box" on page 679.

Set the options for the database checkpoint object

- In the Database Checkpoint Properties dialog box, specify the settings for the checkpoint object. For details, see "Database Checkpoint Properties Dialog Box" on page 679.
- Define the cell selection in the Grid area of the Database Checkpoint Properties dialog box, as follows:

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 

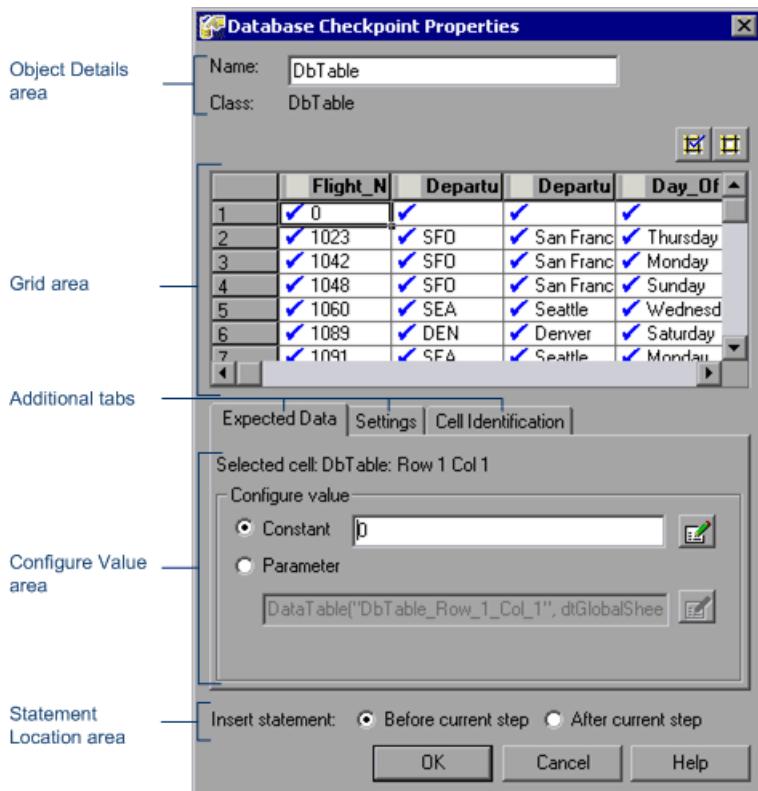
- To modify the SQL query definition, in the Keyword View or Expert View, right-click the database object that you want to modify and select **Object Properties**.
- To modify the expected data of the database checkpoint, open the Database Checkpoint Properties Dialog Box (described on page 679).

Reference

Database Checkpoint Properties Dialog Box

This 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 following image shows an example of the Database Checkpoint Properties dialog box when adding a database checkpoint to an existing test during an editing session. The dialog box options differ slightly during a recording session or when editing an existing checkpoint.



To access	<p>Use one of the following:</p> <ul style="list-style-type: none"> ➤ Insert a new checkpoint step and select an object from your application. For details, see "How to Create or Modify a Database Checkpoint Step" on page 676. ➤ In the Keyword View or Expert View, right-click an existing checkpoint step and select Checkpoint Properties. ➤ In the local or shared object repository, click an existing checkpoint object. The Checkpoint details are displayed on the right side of the object repository dialog box, in the Object Details area.
Important information	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.
Relevant tasks	"How to Create or Modify a Database Checkpoint Step" on page 676
See also	<ul style="list-style-type: none"> ➤ "Database Checkpoints Overview" on page 674 ➤ "Connect to Database Using ODBC Page (Database Query Wizard)" on page 689 ➤ "Specify SQL Statement Page (Database Query Wizard)" on page 691

This dialog box contains the following key areas:

Object Details Area

The following image shows the Object Details area when a database table object is selected for verification.



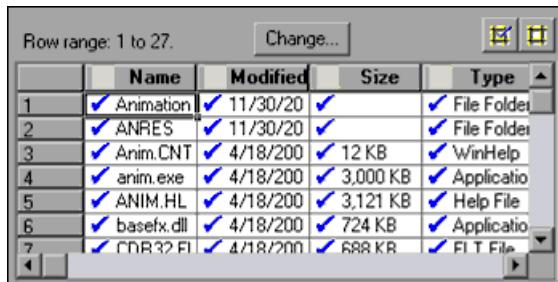
User interface elements are described below:

UI Element	Description
Name	<p>The name that QuickTest assigns to the checkpoint object. By default, the name is the same as the name of the object on which the checkpoint step is being performed. You can specify a different name for the checkpoint object or accept the default name.</p> <p>For a list of naming conventions, see "Naming Conventions" on page 1779.</p>
Class	The type of object (read-only).
	Find in Repository. Displays the checkpoint object in its object repository. Available only when editing an existing checkpoint step. It is not available when creating a new checkpoint step.

Grid Area

Important information	<ul style="list-style-type: none"> ➤ The column header names are captured from the database you selected for your checkpoint. ➤ 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. ➤ You can change the column widths and row heights of the grid by dragging the column and row header dividers.
------------------------------	---

The following image shows the Grid area when all cells are marked for verification.



A screenshot of the QuickTest Define/Modify Row Range dialog box. The grid displays the following data:

	Name	Modified	Size	Type
1	✓ Animation	✓ 11/30/20	✓	✓ File Folder
2	✓ ANRES	✓ 11/30/20	✓	✓ File Folder
3	✓ Anim.CNT	✓ 4/18/200	✓ 12 KB	✓ WinHelp
4	✓ anim.exe	✓ 4/18/200	✓ 3,000 KB	✓ Application
5	✓ ANIM.HL	✓ 4/18/200	✓ 3,121 KB	✓ Help File
6	✓ basefx.dll	✓ 4/18/200	✓ 724 KB	✓ Application
7	✓ CDR32.FI	✓ 4/18/200	✓ 688 KB	✓ FIT File

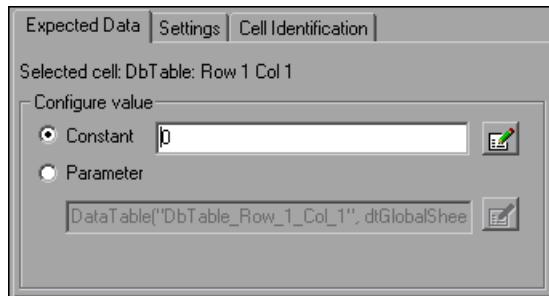
User interface elements are described below:

<grid area>	The grid area displays the captured/expected values of all the cells in the table. Only the ones with blue check marks are used (checked) by the checkpoint. You can instruct QuickTest to check the entire table, specific rows, specific columns, or specific cells. Note: QuickTest checks only cells containing a check mark.
Change	Modifies the row range. For more details, see "Define/Modify Row Range Dialog Box" on page 652.
	Add/Remove from check. Add or remove the selected cells from the check.

Expected Data Tab

Important information	When more than one cell is selected (highlighted) in the grid area, the options in the Expected Data tab are disabled.
------------------------------	--

The following image shows the Expected Data area when the constant value 0 is displayed.



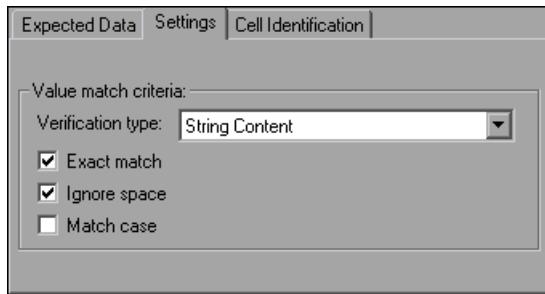
User interface elements are described below:

UI Element	Description
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 details on modifying values, see "Configure Value Area" on page 867.

Settings Tab

Important information	<ul style="list-style-type: none"> ➤ The settings in this tab apply to all cells marked for verification. ➤ The default setting is to treat cell values as strings and to check for the exact text, while ignoring spaces.
------------------------------	--

The following image shows the Settings area when the default values are selected.



User interface elements are described below:

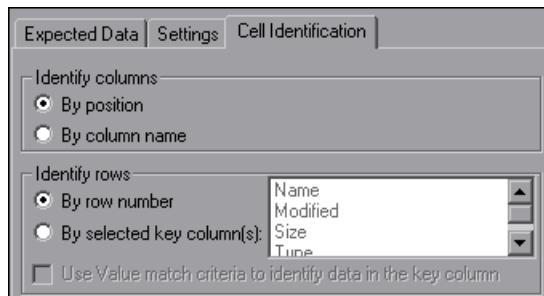
UI Elements	Description
Verification type	<p>Specifies how cell contents are compared:</p> <ul style="list-style-type: none"> ➤ String Content. (Default) Evaluates the content of the cell as a string. For example, 2 and 2.00 are not recognized as the same string. ➤ Numeric Content. Evaluates the content of the cell according to numeric values. For example, 2 and 2.00 are recognized as the same number. ➤ Numeric Range. Compares the content of the cell against a numeric range, where the minimum and maximum values are any real number that you specify. This comparison differs from string and numeric content verification in that the actual result set data is compared against the range that you defined and not against a specific expected value.
Exact match	<p>(Default) Checks that the exact text, and no other text, is displayed in the cell. Clear this box if you want to check that a value is displayed in a cell as part of the contents of the cell.</p> <p>Note: QuickTest displays this option only when String Content is selected as the Verification type.</p>

UI Elements	Description
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 .

Cell Identification Tab

Important information	The settings in this tab apply to all cells marked for verification.
------------------------------	--

The following image shows the Cell Identification area when the default settings are displayed.



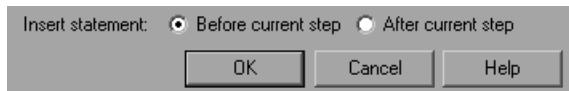
User interface elements are described below:

Identify columns	<p>Specifies how to locate the column in your database that contains the cells whose data you want to compare with the expected data.</p> <p>► By position. Instructs QuickTest to locate cells according to the column position. For example, QuickTest compares cells in the second column in the Database Checkpoint Properties dialog box with cells in the second column in the database. Note: A shift in the position of the columns within the database results in a mismatch.</p> <p>► By column name. (Default) Instructs QuickTest to locate cells according to the column name. For example, QuickTest compares cells in a column named <code>index</code> in the Database Checkpoint Properties dialog box with cells in a column named <code>index</code> in the database. Note: A shift in the position of the columns within the database does not result in a mismatch.</p>
-------------------------	--

Identify rows	<p>Specifies how to locate the row in your database that contains the cells whose data you want to compare with the expected data.</p> <ul style="list-style-type: none"> ➤ By row number. (Default) Instructs QuickTest to locate cells according to the row position. For example, QuickTest compares cells in the second row in the Database Checkpoint Properties dialog box with cells in the second row in the database. Note: A shift in the position of the rows within the database results in a mismatch. ➤ By selected key column(s). Instructs QuickTest to locate the relevant rows by matching the value in a column selected as a key column. For example, if the Flight_Number column is selected as a key column, then QuickTest compares cells for the checkpoint only in rows where the value in the Flight_Number column in the database is the same as the value in the Flight_Number column in the Database Checkpoint Properties dialog box. When you select this option, select one or more key columns from the list of available columns. <p>Notes:</p> <ul style="list-style-type: none"> ➤ A shift in the position of the rows within the database 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. ➤ A key symbol  is displayed in the header of selected key columns.
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. Available only when you select to identify rows By selected key column(s).</p>

Statement Location Area

The following image shows the Statement Location area when inserting a new checkpoint step during an editing session. When inserting a new checkpoint step during a recording session, the **Insert statement** option is not available.



User interface elements are described below:

UI Element	Description
Insert statement	Specifies whether to insert the checkpoint step before or after the currently selected step. The default value is Before current step . Note: Available only when inserting a new checkpoint step during an editing session. During a recording session, the checkpoint step is always inserted as the next step.

Connect to Database Using ODBC Page (Database Query Wizard)

This wizard enables you to define the query for your checkpoint using Microsoft Query or by manually entering a database connection and SQL statement.



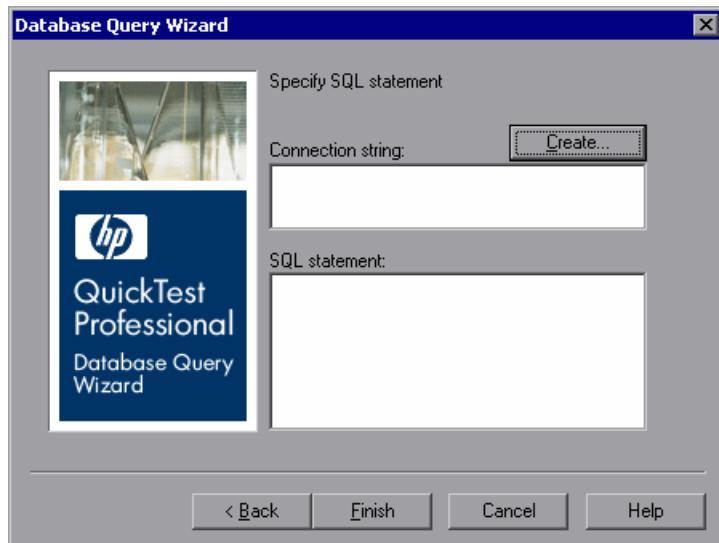
To access	Select Insert > Checkpoint > Database Checkpoint .
Relevant tasks	"How to Create or Modify a Database Checkpoint Step" on page 676
Wizard map	This wizard contains: Connect to Database Using ODBC > Specify SQL Statement (page 691)
See also	"Specify SQL Statement Page (Database Query Wizard)" on page 691

User interface elements are described below:

UI Elements	Description
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 page in the wizard, which enables you to specify the connection string and an SQL statement.
Maximum number of rows	When selected, enables you 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 page when you click Next before opening Microsoft Query. Available only when Create query using Microsoft Query is selected.

Specify SQL Statement Page (Database Query Wizard)

This page enables you to manually specify the database connection string and the SQL statement.



To access	Select Insert > Checkpoint > Database Checkpoint .
Important information	QuickTest takes several seconds to capture the database query and restore the QuickTest window.
Relevant tasks	"How to Create or Modify a Database Checkpoint Step" on page 676
Wizard map	This wizard contains: Connect to Database Using ODBC (page 689) > Specify SQL Statement
See also	"Connect to Database Using ODBC Page (Database Query Wizard)" on page 689

User interface elements are described below:

UI Elements	Description
Connection string	The connection string.
Create	Opens 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	The details of the SQL statement.

Troubleshooting and Limitations - Database Checkpoints

This section describes troubleshooting and limitations for database checkpoints.

- The format of captured values varies depending on the specific system settings. For example, date and time values may be set to different formats.
Workaround: If you are running the test on a different system than the one you used to record the test, confirm that the systems use the same format settings.
- When you create a database checkpoint on one machine and try to run it on different machine, you should have the same ODBC driver installed on both machines.

21

XML Checkpoints

This chapter includes:

Concepts

- XML Checkpoints - Overview on page 694
- XML Checkpoint Types on page 696
- Using XML Objects and Methods to Enhance Your Test on page 697

Tasks

- How to Create or Modify an XML Checkpoint Step on page 698
- How to Update the XML Hierarchy for XML Test Object Operation Checkpoints (WebService Test Objects Only) on page 700

Reference

- XML Checkpoint Properties Dialog Box on page 703
- Edit XML as Text Dialog Box on page 710
- XML Source Selection - Checkpoint / Output Value Properties Dialog Box on page 712
- Schema Validation Dialog Box on page 716

Troubleshooting and Limitations - XML Checkpoints on page 722

Concepts

XML Checkpoints - Overview

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.

XML (Extensible Markup Language) is a meta-markup language for text documents that is endorsed as a standard by the W3C (World Wide Web Consortium). 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 CSS (Cascading Style Sheets) or XSL-FO (XSL Formatting Objects) to present the data.

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 Run Results Viewer. You can also view detailed results by opening the XML Checkpoint Results window. For more information, see Chapter 31, "Run Results Viewer."

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/>

XML Checkpoint Types

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.

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 following methods to populate the XML tree:

- Update the XML Tree Manually
- Import an XML Tree from a File
- Update 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.

For details, see "How to Update the XML Hierarchy for XML Test Object Operation Checkpoints (WebService Test Objects Only)" on page 700.



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 27, "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*.

Tasks

How to Create or Modify an XML Checkpoint Step

If you are modifying an existing XML checkpoint step proceed to the Set the options for the XML checkpoint object step.

This task includes the following steps:

- "Considerations for inserting a new XML checkpoint step" on page 698
- "Insert a new XML checkpoint step" on page 699
- "Set the options for the XML checkpoint object" on page 699
- "View XML checkpoint results" on page 699

Considerations for inserting a new XML checkpoint step

Before inserting a new XML checkpoint step, make sure to review all relevant information:

Considerations	
Availability	<ul style="list-style-type: none">➤ Recording sessions➤ Editing sessions (XML file and test object checkpoints only)
Supported environments	"Supported Checkpoints and Output Values Per Add-in" on page 1783
XML checkpoint types	<ul style="list-style-type: none">➤ XML checkpoints for Web pages and Frames➤ XML checkpoints to directly access and verify specific XML files in your system➤ XML test object checkpoints 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.

Insert a new XML checkpoint step

Insert a new XML checkpoint step, as described in "How to Insert a Checkpoint Step in a Test" on page 598. The XML Checkpoint Properties Dialog Box opens (as described on page 703). If you are inserting an XML checkpoint on an XML file or an XML test object, the XML Source Selection - Checkpoint / Output Value Properties Dialog Box opens first (described on page 712).

Set the options for the XML checkpoint object

In the XML Checkpoint Properties Dialog Box, specify the settings for the XML checkpoint. For details, see "XML Checkpoint Properties Dialog Box" on page 703.

View XML checkpoint results

You can view summary results of the XML checkpoint in the Run Results Viewer. You can view detailed results by opening the XML Checkpoint Results window. For more information on XML checkpoint results, see "XML Checkpoint Results" on page 1185.

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.

How to Update the XML Hierarchy for XML Test Object Operation Checkpoints (WebService Test Objects Only)

Note: This section is relevant only when working with XML checkpoints on WebService test object operations (with the QuickTest Professional Web Services Add-in).

QuickTest cannot generate the expected return values of an operation when you insert an XML checkpoint on a Web service operation. You therefore, need to update the elements, attributes and values in your XML tree.

You can use one of the steps below to update the XML hierarchy for XML Test Object Operation Checkpoints.

This task includes the following steps:

- "Update the XML Tree Manually" on page 700
- "Import an XML Tree from a File" on page 700
- "Update the XML Tree Using Update Run Mode" on page 701
- "View XML checkpoint results" on page 702

Update the XML Tree Manually

Use the options in the XML Checkpoint Properties Dialog Box (as described on page 703) to update the XML tree by adding elements, attributes, and values.

Import an XML Tree from a File



- 1 Use the **Import XML** button in the XML Checkpoint Properties Dialog Box (as described on page 703) to replace a node in the XML tree.
- 2 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 22, "Parameterizing Values."

Update the XML Tree Using Update Run Mode

- 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.

For details on using Update Run Mode, see "How to Update Test Object Descriptions, Checkpoints, Output Values, or Active Screens" on page 1252.



- If you want to confirm that QuickTest successfully updated your checkpoint, expand the tree in the Run Results Viewer and select the XML checkpoint. Then check that **Update done** is displayed in the pane on the right. (If the Run Results Viewer did not open automatically at the end of the run, click the **Results** button or select **Automation > Results**.)

View XML checkpoint results

You can view summary results of the XML checkpoint in the Run Results Viewer. You can view detailed results by opening the XML checkpoint Results window. For more information on XML checkpoint results, see "XML Checkpoint Results" on page 1185.

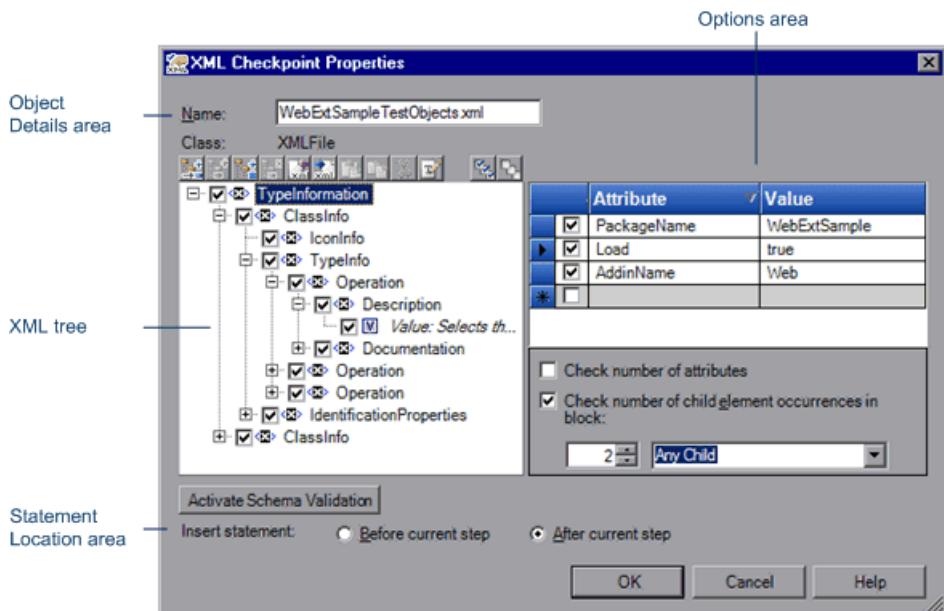
Note: XML checkpoints on Web service operations compare the expected values of the checkpoint to the actual values returned from the last native Web service operation performed on the test object. If a different Web service operation step is performed prior to the checkpoint, then the checkpoint fails.

Reference

XML Checkpoint Properties Dialog Box

This dialog box enables you to choose which elements, attributes, and/or values to check in an XML checkpoint. You can also add, modify and delete elements, attributes, and values in the XML tree.

The following image shows an example of the XML Checkpoint Properties dialog box when adding an XML checkpoint to an existing test during an editing session. The dialog box options differ slightly during a recording session or when editing an existing checkpoint.



To access	Use one of the following: <ul style="list-style-type: none"> ➤ Insert a new checkpoint step and select a Web page or frame that contains an XML document. For details, see "How to Create or Modify an XML Checkpoint Step" on page 698. ➤ In the Keyword View or Expert View, right-click an existing XML checkpoint step and select Checkpoint Properties. ➤ In the local or shared object repository, click an existing XML checkpoint object. The Checkpoint details are displayed on the right side of the object repository dialog box, in the Object Details area.
Important information	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 "Edit XML as Text Dialog Box" on page 710.
Relevant tasks	"How to Create or Modify an XML Checkpoint Step" on page 698

This dialog box contains the following key areas:

- "Object Details Area" on page 704
- "XML Tree" on page 705
- "Options Area" on page 708
- "Schema Validation" on page 709
- "Statement Location Area" on page 709

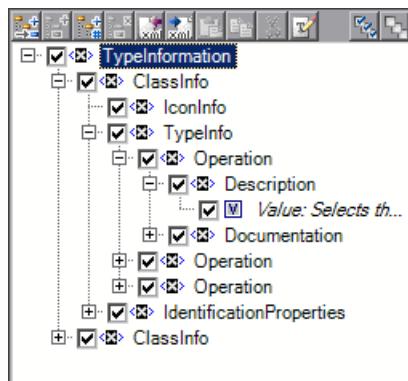
Object Details Area



User interface elements are described below:

UI Element	Description
Name	<p>The name that QuickTest assigns to the XML checkpoint object. By default, the name is the same as the name of the object on which the XML checkpoint step is being performed. You can specify a different name for the XML checkpoint object or accept the default name.</p> <p>For a list of naming conventions, see "Naming Conventions" on page 1779.</p>
Class	<p>The type of object (read-only).</p>
	Find in Repository. Displays the checkpoint in its object repository. Available only when editing an existing checkpoint step. It is not available when creating a new checkpoint.

XML Tree



Important information	<ul style="list-style-type: none"> ➤ 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 or value node in the XML tree to display, edit, or parameterize its expected attributes and values in the options area of the XML Checkpoint Properties dialog box. For details, see the section on the Options area, below. ➤ The XML tree pane is resizable.
------------------------------	---

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.

Command	Icon	Description
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 "Edit XML as Text Dialog Box" on page 710.
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. Available: Only from the context menu (right-click menu).

Options Area

	Attribute	Value
▶	<input checked="" type="checkbox"/> PackageName	\WebExtSample
▶	<input checked="" type="checkbox"/> Load	true
▶	<input checked="" type="checkbox"/> AddinName	\Web
*	<input type="checkbox"/>	

<input type="checkbox"/> Check number of attributes
<input checked="" type="checkbox"/> Check number of child element occurrences in block:
<input type="button" value="2"/> <input type="button" value="Any Child"/> <input type="button" value="▼"/>

Important information

The **Attribute** and **Value** columns are resizable.

User interface elements are described below:

UI Element	Description
Attribute	The list of attributes for the selected element or value node in the XML tree.
Value	The list of values for the selected element or value node in the XML tree.

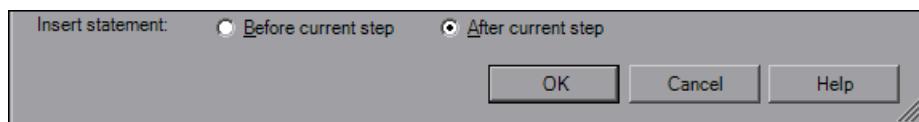
UI Element	Description
Check number of attributes	Checks the number of attributes that are attached to the element.
Check number of child element occurrences in block	<p>Displays the number of child elements associated with the selected parent element. If you select this option, QuickTest verifies that 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 details, see "Schema Validation Dialog Box" on page 716.

Statement Location Area

The following image shows the Statement Location area when inserting a new checkpoint step during an editing session. When inserting a new checkpoint step during a recording session, the **Insert statement** option is not available.

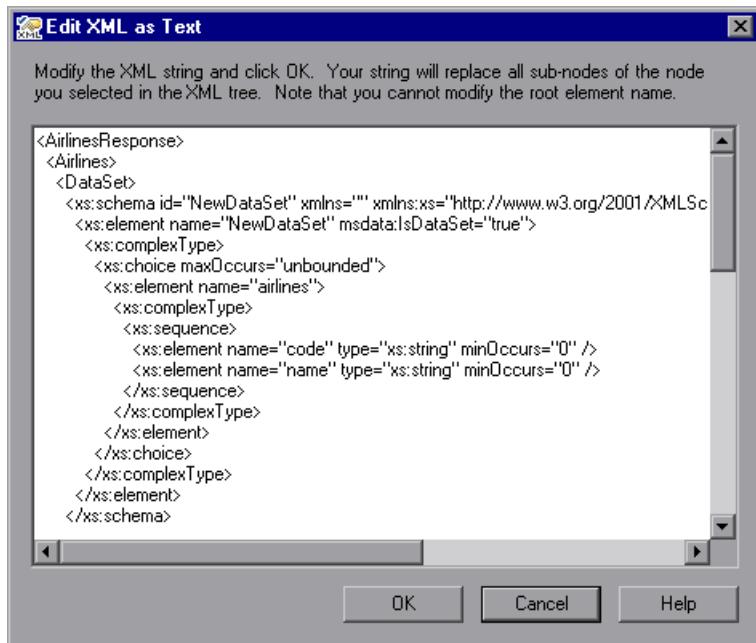


User interface elements are described below:

UI Element	Description
Insert statement	<p>Specifies whether to insert the checkpoint step before or after the currently selected step. The default value is Before current step.</p> <p>Available: Only when inserting a new checkpoint step during an editing session. During a recording session, the checkpoint step is always inserted as the next step.</p>

Edit XML as Text Dialog Box

This dialog box enables you to edit XML content from the XML tree in a text editor.

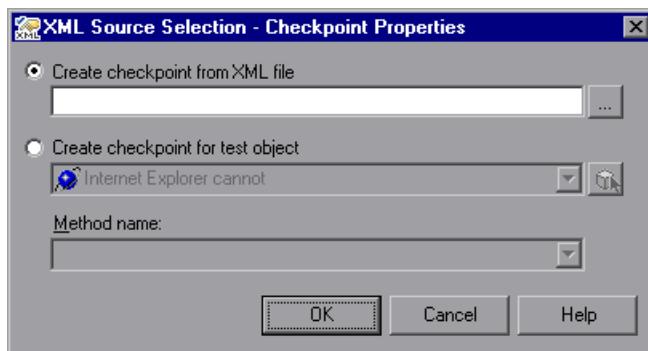


To access	Click the Edit XML as Text button  in the XML tree area of the "XML Checkpoint Properties Dialog Box" on page 703.
Important information	<ul style="list-style-type: none">▶ This dialog box is used mainly for constructing an entire XML segment from a string or for fixing syntax problems that 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.▶ You cannot modify the name of the root element displayed in the Edit XML as Text dialog box.
Relevant tasks	"How to Create or Modify an XML Checkpoint Step" on page 698

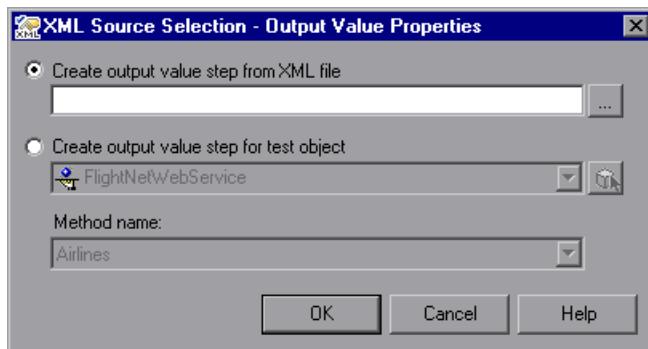
XML Source Selection - Checkpoint / Output Value Properties Dialog Box

This dialog box enables you to specify the XML file or test object for which you want to insert an XML checkpoint/output value step.

The following image shows an example of the XML Source Selection - Checkpoint Properties dialog box with the **Method name** box displayed. The **Method name** box is available only when the Web Services Add-in is installed and loaded.



The following image shows an example of the XML Source Selection - Output Value Properties dialog box with the **Method name** box displayed. The **Method name** box is available only when the Web Services Add-in is installed and loaded.



To access	<p>For checkpoints: Select Insert > Checkpoint > XML Checkpoint (From Resource)</p> <p>For output values: Select Insert > Output Value > XML Output Value (From Resource)</p>
Important information	<ul style="list-style-type: none">▶ The Method name box is available only when the Web Services Add-in is installed and loaded. The Method name box is enabled only when you select a WebService test object.▶ 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 "How to Update the XML Hierarchy for XML Test Object Operation Checkpoints (WebService Test Objects Only)" on page 700.
Relevant tasks	<ul style="list-style-type: none">▶ "How to Create or Modify an XML Checkpoint Step" on page 698▶ "How to Create or Modify an XML Output Value Step" on page 798

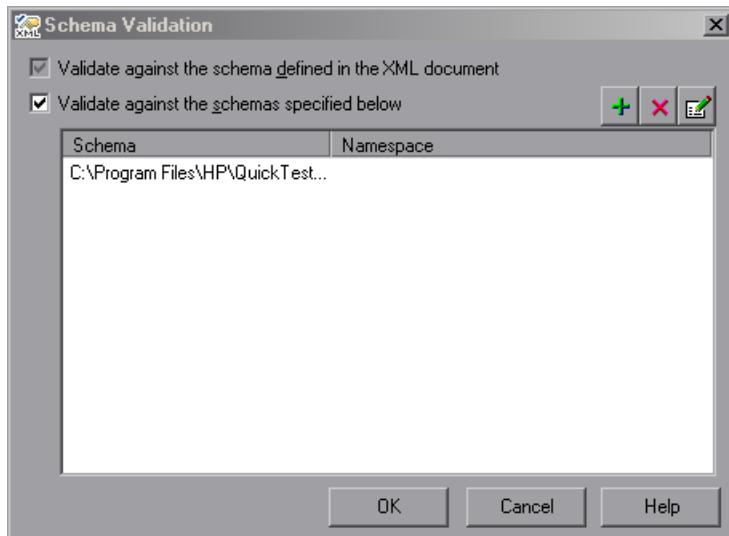
User interface elements are described below:

UI Elements	Description
Create checkpoint step from XML file / Create output value step from XML file	<p>Specifies the file for which you want to insert a checkpoint/output value. In the box, you can:</p> <ul style="list-style-type: none">➤ Enter the URL or the file path of the XML file.➤ Click the browse button to open the Open XML File dialog box and navigate to the XML file. You can specify an XML file either from your file system or from Quality Center. <p>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 "Folders Pane (Options Dialog Box)" on page 1431.</p>

UI Elements	Description
Create checkpoint step for test object / Create output value step for test object	<ul style="list-style-type: none"> ▶ Specifies an XML checkpoint/output value for a test object. From the list, select the test object. If you are creating the checkpoint/output value for a Web service operation return value, select the operation. ▶ 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. <p>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).</p> <p>Tips:</p> <ul style="list-style-type: none"> ▶ To select an object that is not displayed in the list, you can click Object from Repository . Then select a 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 "Select Test Object Dialog Box" on page 513. ▶ 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.
Method name (WebService test objects only)	The Web service operation whose return values you want to check.

Schema Validation Dialog Box

This 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.



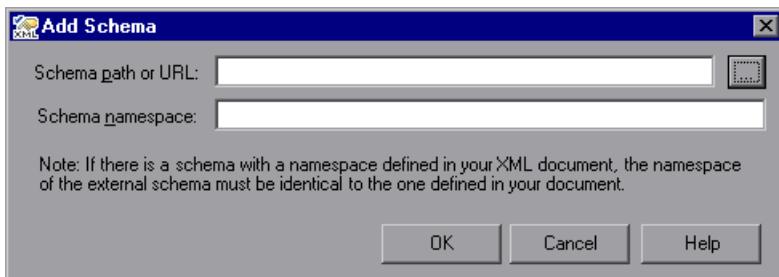
To access	XML Checkpoint Properties dialog box > Activate Schema Validation button
Important information	<p>► If you are validating an XML file using a schema defined in the XML file, the schema can be defined with an absolute or relative path. When you specify a relative path, QuickTest searches for the schema in the folders listed in the Folders pane of the Options dialog box. For more information, see "Folders Pane (Options Dialog Box)" on page 1431.</p> <p>► 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.</p> <p>► 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.</p> <p>► 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.</p>
Relevant tasks	"How to Create or Modify an XML Checkpoint Step" on page 698

User interface elements are described below:

UI Elements	Description
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	<p>Instructs QuickTest to use one or more external XML schema files to validate the hierarchy of your XML. Any schemas defined within your XML document are also checked.</p> <p>Note: If you select this option:</p> <ul style="list-style-type: none"> ➤ The Validate against the schema defined in the XML document option is automatically selected and is disabled. ➤ The Add Schema, Remove Schema, and Modify Schema buttons are enabled.
	Add Schema. Enables you to add an external schema file to the list. For more information, see "Add Schema Dialog Box" on page 719.
	Remove Schema. Enables you to remove the selected external schema file from the list.
	Modify Schema. Enables you to modify the details of the selected external schema file in the list. For more information, see "Edit Schema Dialog Box" on page 721.

Add Schema Dialog Box

This 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.



To access	XML Checkpoint Properties dialog box > Activate Schema Validation button > Add Schema button
Relevant tasks	"How to Create or Modify an XML Checkpoint Step" on page 698

User interface elements are described below:

UI Elements	Description
Schema path or URL	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.) The namespace of your schema file. 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.

Edit Schema Dialog Box

This 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.



To access	XML Checkpoint Properties dialog box > Activate Schema Validation button > Modify Schema button
Relevant tasks	"How to Create or Modify an XML Checkpoint Step" on page 698

User interface elements are described below:

UI Elements	Description
Schema path or URL	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.) The namespace of your schema file. 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.

Troubleshooting and Limitations - XML Checkpoints

This section describes troubleshooting and limitations for XML checkpoints.

- When executing an XML checkpoint on an XML file that contains > as a value, an error message may occur.
- When you add a new value node to an XML node, in some cases the new value may not be displayed.

Workaround: Close the Edit XML as Text dialog box and reopen it to display the new value node correctly.

- When inserting an XML file checkpoint on a file that cannot be loaded, or a file that is formatted incorrectly, you may receive an error message.
- Creating and running XML checkpoints for large XML documents may take a few minutes.

22

Parameterizing Values

This chapter includes:

Concepts

- ▶ Parameterizing Values Overview on page 725
- ▶ Test and Action Input Parameters on page 727
- ▶ Data Table Parameters on page 731
- ▶ Environment Variable Parameters on page 734
- ▶ When to Choose Global or Action Data Table Parameters on page 737
- ▶ Automatically Parameterizing Steps on page 738
- ▶ Data Driver on page 744
- ▶ Example of a Parameterized Test on page 744

Tasks

- ▶ How to Parameterize Values for Operations or Local Objects on page 750
- ▶ How to Parameterize a Checkpoint Property Value on page 751
- ▶ How to Use User-Defined External Environment Variables on page 753
- ▶ How to Create an External Environment Variables File on page 755

Reference

- ▶ Default Parameter Values on page 757
- ▶ Built in Environment Variables on page 758
- ▶ Parameter Options Dialog Box (Test/Action Parameter) on page 760
- ▶ Parameter Options Dialog Box (Data Table) on page 763
- ▶ Parameter Options Dialog Box (Environment) on page 766

- [Edit Complex Value Dialog Box on page 769](#)
- [Parameter Options Dialog Box \(Random Number\) on page 770](#)
- [Data Driver Dialog Box on page 773](#)
- [Data Driver Wizard - Select Parameterization Type Page on page 776](#)
- [Data Driver Wizard - Parameterize the Selected Step Page on page 778](#)

Concepts

Parameterizing Values Overview

You can enhance your test by parameterizing the values that it uses. A **parameter** is a variable that is assigned a value from an external data source or generator.

You can parameterize values of:

- Checkpoints.
- Object properties for a selected step.
- Operation arguments defined for a selected step.
- One or more properties of an object stored in the local object repository in the Object Properties dialog box or Object Repository window. For details on parameterizing a property value for an object in a shared object repository, see Chapter 6, "Shared Object Repositories."

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 details on the local object repository, see Chapter 4, "Managing Test Objects in Object Repositories."

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. For details, see "Test and Action Input Parameters" on page 727.
- **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 details, see "Data Table Parameters" on page 731.
- **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 details, see "Environment Variable Parameters" on page 734.
- **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 instruct QuickTest to generate a random number and insert it in a **number of tickets** edit box.

Tips:

- If you want to parameterize all the operation arguments in your test or in one or more actions of a test, consider using the Automatically parameterize steps option. For details, see "Automatically Parameterizing Steps" on page 738.
- If you want to parameterize the same value in several steps in your test, consider using the Data Driver rather than adding parameters manually. For details, see "Data Driver" on page 744.
- When you use the Step Generator to add new steps, you can parameterize the values for the operation you select. For details, see "How to Insert Steps Using the Step Generator" on page 900.
- You can also parameterize identification property values of test objects in the object repository using repository parameters. For details, see "Working with Repository Parameters" on page 252.

Test and Action Input Parameters

You can parameterize a value in a step using a test or action input parameter. 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.

You can parameterize a value using a test or action parameter only if the parameter has been defined for the test or action. For details on defining parameters, see "Parameters Pane (Test Settings Dialog Box)" on page 1479, "Parameters Tab (Action Properties Dialog Box)" on page 563, and "Action Call Properties Dialog Box" on page 550.

Example:

Suppose that Action3 is a nested action of Action1 (a top-level action), and you want to parameterize a value 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 value 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 details, see "Considerations for Setting Action Parameters" on page 729.

This section also includes:

- "How to Parameterize a Value" on page 728
- "Using Action Parameters in Steps in the Expert View" on page 728
- "Considerations for Setting Action Parameters" on page 729

How to Parameterize a Value

You parameterize values 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.

Alternatively, you can enter the parameter name in the Expert View using the **Parameter** utility object. For details, see "Using Action Parameters in Steps in the Expert View" on page 728.

For details, see "How to Parameterize Values for Operations or Local Objects" on page 750 and "How to Parameterize a Checkpoint Property Value" on page 751.

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 details, see "Storing Output Values" on page 787.

For details, see "Parameter Options Dialog Box (Test/Action Parameter)" on page 760.

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")` 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.

Example:

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

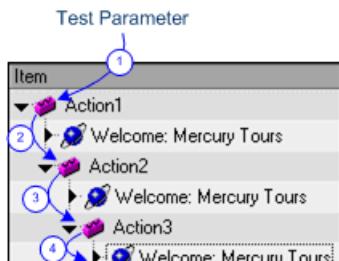
NumTableRows = Browser("Mercury").Page("List Of Sales").
    WebTable("Orders").RowCount

Parameter("TotalValue") = Browser("Mercury").Page("List Of Sales").
    WebTable("Orders").GetCellData(NumTableRows,"Total")
```

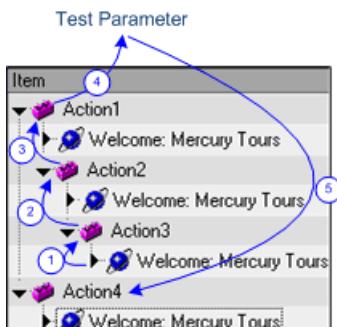
Considerations for Setting Action Parameters

- 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.
- 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). For example:

Test -> Action1 -> Action2 -> Action3 -> (Action3) Step 1



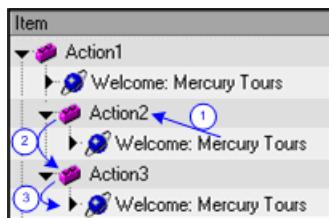
- Output action parameter values can be retrieved from a previous action at the same hierarchical level, from a parent action, or from the current action. You can use an action output value from one action within the step of another action if:
 - You pass the value from action to action up the test hierarchy 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.

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).

When you create a new data table parameter, a new column is added at the end of 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, the values in the column for the selected parameter are retained and are not overwritten by the current value of the parameter.

Note: If you parameterize a value that is defined as a variant value, then when QuickTest retrieves the value from the data table, it will retrieve it as a string. This will occur even if you enter a numeric value in the data table. For example, if you parameterize the argument of a step such as:
`WpfWindow("MyWindow").WpfComboBox("cb").Select 1`
and you enter the value 1 in the data table, then when the step runs, it will retrieve the value as a string: "1", and the step will fail.

A data table comprises:

- **Columns.** Each column in the table represents the list of values for a single data table parameter.
- **Column headers.** The data table parameter names.
- **Rows.** 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.
- **Sheets.** The data table contains two types of sheets—Global and action-specific. For details, see "Data Table Sheets" on page 1327.

For details on entering values in the data table, see Chapter 38, "Data Table Pane."

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 details, see Chapter 23, "Output Values."

Example 1:

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.

Example 2:

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.

Although 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.

A1		Acapulco						
		departure	arrival	C	D	E	F	G
1	Acapulco	New York						
2	New York	Paris						
3	London	Frankfurt						
4								
5								

Global Action1

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

A1		Acapulco						
		departure	arrival	C	D	E	F	G
1	Acapulco	New York						
2	New York	Paris						
3	London	Frankfurt						
4								
5								
6								
7								

Global Action1

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.

Example:

You can instruct QuickTest to 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.

Tip: Environment parameters are especially useful for localization testing, when you want to test an application where the user interface strings change according to 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 details, see Chapter 38, "Data Table Pane."

There are several types of environment variables:

User-Defined Internal Environment Variables

User-defined internal environment variables are defined 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, as described in "Environment Pane (Test Settings Dialog Box)" on page 1483.
- Parameter Options dialog box, as described in "Parameter Options Dialog Box (Environment)" on page 766.

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 details, see Chapter 23, "Output Values."

User-Defined External Environment Variables

User-defined external environment variables are predefined 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. External environment variable values are designated as read-only within the test.

External environment variable files are comprised of a list of variable-value pairs in **.xml** format. You select the active external environment variable file for a test in the Environment pane of the Test Settings dialog box (see "Environment Pane (Test Settings Dialog Box)" on page 1483). Then you can use the variables from the file as parameters.

You can set up your environment variable XML files manually, or you can define the variables as internal environment variables in the Environment pane of the Test Settings dialog box and use the **Export** button to create the **.xml** file with the correct structure.

For details on creating and using user-defined external environment variable files see, "How to Use User-Defined External Environment Variables" on page 753.

Note:

- You can also store environment variable files in Quality Center. For details, see "Environment Variable Files and Quality Center" on page 736.
 - 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.
-

Built-in Environment Variables

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 details, see "Environment Pane (Test Settings Dialog Box)" on page 1483.

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 details, see "Parameter Options Dialog Box (Environment)" on page 766.

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 details, see the section on using environment variables to specify the record and run details for your test in the *HP QuickTest Professional Add-ins Guide*.

Environment Variable Files and 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. 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.

When to Choose Global or Action Data Table Parameters

When you parameterize a step in a test using the data table, you must decide whether you want to make it a **global data table parameter** or a **local data table parameter**.

You use local data table parameters when you want the data to be used only for a single action. You use global data table parameters when you want the data to be available to other actions, and when you want subsequent iterations to use different data for a particular parameter (each time the test repeats or each time the action repeats within the test).

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.

► **Global data table parameters** take data from the Global sheet in the data table. The Global sheet contains the data that replaces global parameters in each iteration of the test.

By default, the test runs one iteration for each row in the Global sheet of the data table. You can also set the test to run only one iteration. You can also set the test 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.

For details on setting global iteration preferences, see "Run Pane (Test Settings Dialog Box)" on page 1471.

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 details, see Chapter 23, "Output Values."

- **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.

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 details on setting action iteration preferences, see "Run Tab (Action Call Properties Dialog Box)" on page 552.

After running a parameterized test, you can view the actual values taken from the data table in the run results run-time data table. For details, see "Data Table Pane (Run Results Viewer)" on page 1131.

Automatically Parameterizing Steps

You can instruct QuickTest to automatically parameterize the steps in your test actions at the end of a recording session.

This enables you to create actions that can be used for a variety of different purposes or scenarios by referencing different sets of data.

To activate this option, select the **Automatically parameterize steps** option in the General tab of the Options dialog box. You can set the option to use **Global Data Table Parameters** or **Test Parameters**.

When you stop a recording session while this option is selected, QuickTest replaces the constant values in the test object operation arguments of your steps with either Data Table parameters or action parameters, based on your selection in the Options dialog box.

QuickTest performs this automatic parameterization for all relevant steps in any action in your test, in which you recorded one or more steps during that recording session.

- When you select to use **Global Data Table Parameters**, the generated parameters are added to the Global sheet of the data table.

If you work with HP ALM, you can map these parameters to the column names of a data resource and then use different configurations in your test sets. For details on HP ALM configurations and mapping data table parameters to HP ALM data resources, see "Data Awareness in HP ALM" on page 1613.

- When you select to use **Test Parameters**, QuickTest parameterizes the step with a newly created action parameter. It also creates a corresponding test parameter.

For more details on how QuickTest creates these parameters, which types of operation arguments are included and excluded from the parameterization, and other important criteria to consider, see "Guidelines and Considerations for Automatically Parameterizing Steps" on page 740.

Guidelines and Considerations for Automatically Parameterizing Steps

Before you begin a recording session with the **Automatically parameterize steps** option activated, make sure you are aware of the following guidelines and considerations:

General Guidelines

- The automatic parameterization process runs on all relevant steps in all local actions in which you recorded one or more steps during the recording session. It will also parameterize the action that is displayed when you stop the recording session, even if you did not record any steps in that action.
- All relevant steps in any relevant action are parameterized, whether or not those steps were added in the current recording session. Therefore, you can perform automatic parameterization on an existing action by starting and stopping a recording session without adding any steps.
- Automatic parameterization is not performed on external actions that are called from your test, nor for actions called using the **LoadAndRunAction** statement.
- In general, simple, constant (string, number, boolean) test object and utility object operation arguments are parameterized. Therefore, if the following are contained in a method argument, they are not parameterized:
 - arguments that are already parameterized
 - variables
 - enumeration constants, such as **micLeftbtn**
 - expressions, such as: **x = 3**
 - Assignments of values, such as:
`Window("Notepad").WinMenu("Menu").ExpandMenu = True`
 - mathematical or other concatenation operations, such as "**Hello World**" & **micCtrl** & "**S**"

- VBScript statements, such as `msgbox "hello"`
- VBScript language statements such as `For`, `If`, `Do`, `While`
- steps inside called functions from function libraries or in functions or sub-routines defined directly within the action
- In addition to the above general rule, operation arguments in the following scenarios are also not parameterized:
 - **SAPGuiTable.Input**, **SAPGuiGrid.Input**, or **SAPGuiAPOGrid.Input** steps (inserted using the **Auto-parameterize table and grid controls** option). For details, see the **auto-parameterize table** topic in the SAP section of the *HP QuickTest Professional Add-ins Guide*.
 - Native methods and properties accessed by the **Object** property.
 - Steps for test objects that are stored in variables. For example, the "text" constant is not parameterized in:
`Set MyEditBox = Browser("x").Page("x").WebEdit("myedit")
MyEditBox.Set "text"`
 - Steps containing programmatic descriptions, such as:
`Browser("x").Page("x").WebEdit(MyDescription).Set "text"`
or
`Browser("x").Page("x").WebEdit("prop:=value", "prop2:=value2").Set "text".)`
 - User-defined functions that are registered to test objects using a **RegisterUserFunc** statement.

However, built-in QuickTest test object operations that were overridden using a **RegisterUserFunc** statement are parameterized.

- Operation arguments of type variant, when the value is a number.

For example, in the following statement, the variant argument of the `Select` method is not parameterized, because it is a number.

`WpfWindow("MyWindow").WpfComboBox("cb").Select 1`

However, in the following statement, the variant argument of the `Select` method is parameterized, because it is a string.

`WpfWindow("MyWindow").WpfComboBox("cb").Select "item1"`

- When working with Siebel applications, only operation arguments of **Sbl*** test objects, which represent standard interactivity (SI) objects are parameterized. The operation arguments of **Sieb*** test objects, which represent Siebel High Interactivity (HI) API test objects, are not parameterized. For details on these types of objects, see the **Siebel** section of the *HP QuickTest Professional Add-ins Guide*.
- The name of the parameter that QuickTest creates for each method argument is in the format **TestObjectName_ArgumentName**.
- If a parameter with this name already exists for the relevant parameter type, QuickTest appends an underscore and a number to the end of the new parameter to create a unique name.
- If the test object name contains characters that are not valid for parameter names or if it contains multi-byte characters, then the test object class is used instead of the test object name.
- The automatic parameterization option is relevant only for tests. QuickTest does not automatically parameterize steps after you record steps in a component.
- If you select the **Automatically generate With statements after recording** option in addition to the **Automatically parameterize steps** option, then when you stop a recording session, QuickTest converts the steps to **With** statements and then parameterizes the operation arguments within them.

Guidelines for Automatic Parameterization to Data Table Parameters

- When you use **Global Data Table Parameters** for the automatic parameterization option, the arguments are parameterized using Global Data Table parameters. The constant value that the parameter replaces is stored in the created column in the Global data sheet.
If the Global data sheet already contains more than one row, the value is entered in all rows of the created data table column.
- Data table sheets can contain 256 columns. At the end of a recording session, QuickTest parameterizes the relevant operation arguments until it reaches the 256th column. If QuickTest reaches the maximum number of data table parameters, it stops parameterizing at that point and a message informs you that not all relevant steps were parameterized.

- If the test uses an external data table, and the data table file is read-only, locked by another user, or checked in to Quality Center version control, the automatic parameterization will not be performed and a message will be displayed when you finish your recording session.

Guidelines for Automatic Parameterization to Test Parameters

- When you use **Test Parameters** for the automatic parameterization option:
 - Each relevant method argument is converted to a new action parameter.
 - A corresponding test parameter is also created.
 - The constant value that the parameter replaces is stored as the default value for the action parameter (in the Action Properties dialog box) as well as for the test parameter (in the Parameters pane of the Test Settings dialog box).

For **top-level actions**, QuickTest also promotes (parameterizes) the action parameter value (in the Action Call Properties dialog box) to use the corresponding test parameter for you.

For **nested actions**, QuickTest creates an action parameter and a corresponding test parameter. However, the action parameter value (in the Action Call Properties dialog box) is not promoted (parameterized) to use a parent action parameter. It is set to use the constant value that it replaced.

When working with nested actions, if you want to use the created test parameter in order to provide the actual values from an external application that calls your test, such as from HP Service Test or SAP eCATT, you must manually create corresponding action parameters for all actions that call your action, and then pass (promote) the action parameter values up to each parent action and then to the test parameter level. For details on passing values between action and test parameters, see "How to Use Action Parameters - Use-Case Scenario" on page 548.

Data Driver

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.

Note:

- ▶ 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.
-

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.

This example includes:

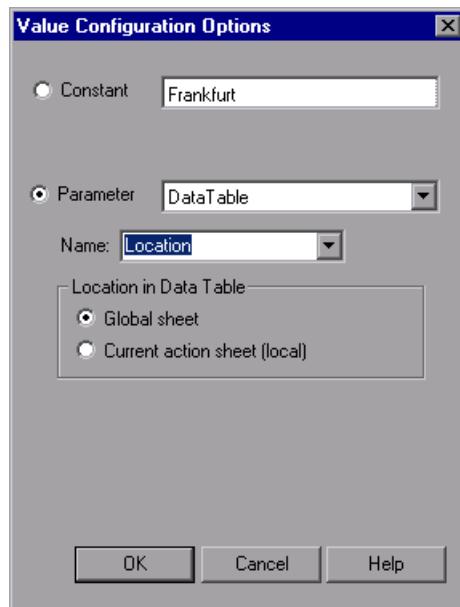
- "Step 1: Parameterize a Step" on page 746
- "Step 2: Parameterize a Checkpoint" on page 747
- "Step 3: Enter Data in the Data Table" on page 749
- "Step 4: View the Parameterized Test" on page 749

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 details on parameterizing a step, see "How to Parameterize Values for Operations or Local Objects" on page 750.

 **Step 2: Parameterize a Checkpoint**

Add a parameterized text checkpoint to check that the correct locations were selected before you book a flight. To do this:

Select the Select a Flight step. In the Active Screen, highlight the text Frankfurt to Acapulco, right-click and insert a text checkpoint:



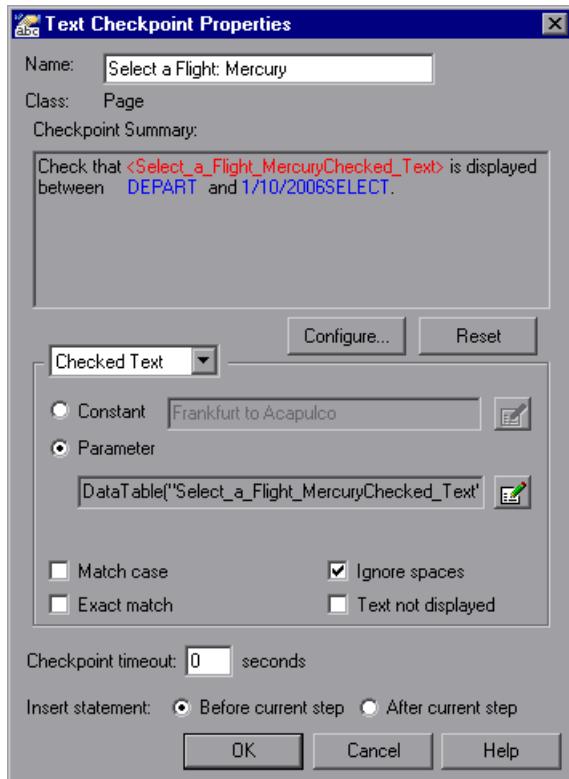
The screenshot shows a flight selection interface. At the top, a yellow bar contains the text "SELECT FLIGHT" and a small airplane icon. Below this, a message reads: "Select your departure and return flight from the selections below. Your total price will be higher than quoted if you elect to fly on a different airline for both legs of your travel." The main section is titled "DEPART" and shows a flight search for "Frankfurt to Acapulco" on "03/09/2004". A table lists five flight options:

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



In the Text Checkpoint Properties dialog box, select **Parameter** to parameterize the selected text. Select the **Parameter** radio button and click the **Parameter Options** button.

In the Parameter Options dialog box, rename the 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 details on parameterizing a checkpoint, see "How to Parameterize a Checkpoint Property Value" on page 751.

Step 3: Enter Data in the Data Table

Complete the data table, for example:

Data Table				
	F4			
	Location	Check_Locations_Text	C	D
1	Frankfurt	Frankfurt to Acapulco		
2	Acapulco	Acapulco to Frankfurt		
3				
4				
5				

For details on data tables, see Chapter 38, "Data Table Pane."

Step 4: View the Parameterized Test

The following example shows the test after parameterizing the step and creating a parameterized text checkpoint.

 Welcome: Mercury T...			
 userN ame	Set	"mercury"	Enter "mercury" in the "userName" edit box.
 passw ord	SetS...	"404ee5e998b25bdc6f116b031452..."	Enter the encrypted string "404ee5e998b25bdc6f116b031452..." in the "password" edit box.
 Sig n-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 specified value.
 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	"333665777"	Enter "333665777" 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.

Tasks

How to Parameterize Values for Operations or Local Objects

This task describes different ways to parameterize the values for operations in a step or objects in the local object repository.

This task includes:

- "Parameterize a value for an operation using the parameterization icon" on page 750
- "Use the Data Driver to parameterize some or all of the occurrences of a constant value in your test" on page 751
- "Enter input and output parameters as values in the Expert View" on page 751

Parameterize a value for an operation using the parameterization icon

- 1 To parameterize a value for an operation using the parameterization icon, in the Keyword View click in the **Value** column of the required step.
 - To parameterize property values for local objects, do one of the following:
 - 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 the parameterization icon

- To parameterize a value for an operation, in the Keyword View, click the parameterization icon for the value that you want to parameterize.
- To parameterize property values for local objects, in the Object Repository 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, showing the currently defined value.

3 Parameterize the value using the Value Configuration Options dialog box

For details, see "Value Configuration Options Dialog Box" on page 872.

Use the Data Driver to parameterize some or all of the occurrences of a constant value in your test

For details, see "Data Driver Dialog Box" on page 773.

Enter input and output parameters as values in the Expert View

You can enter input and output parameters as values in the Expert View using the Parameter utility object. For details, see "Using Action Parameters in Steps in the Expert View" on page 728.



How to Parameterize a Checkpoint Property Value

This task describes how to parameterize the values for properties in checkpoints. You can parameterize the values for properties in checkpoints in one of the following ways:

This task includes the following steps:

- "Parameterize a value for a property in a checkpoint using the Checkpoint Properties dialog box" on page 752
- "Use the Data Driver to parameterize some or all of the occurrences of a constant value in your test." on page 753

- "Enter input and output parameters as values in the Expert View" on page 753

Parameterize a value for a property in a checkpoint using the Checkpoint Properties dialog box

1 Open the dialog box for the checkpoint properties

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** window and select the checkpoint.



2 Use the options in the Configure Value area to parameterize the value

a Select the property whose value you want to parameterize from the table of properties.

b In the **Configure value** area, select **Parameter**.



c Click the **Parameter Options** button. Depending on the type of checkpoint, one of the following parameter options dialog boxes opens:

- "Parameter Options Dialog Box (Test/Action Parameter)" on page 760
- "Parameter Options Dialog Box (Data Table)" on page 763
- "Parameter Options Dialog Box (Environment)" on page 766
- "Parameter Options Dialog Box (Random Number)" on page 770

Use the Data Driver to parameterize some or all of the occurrences of a constant value in your test.

For details, see "Data Driver Dialog Box" on page 773.

Enter input and output parameters as values in the Expert View

You can enter input and output parameters as values in the Expert View using the Parameter utility object. For details, see "Using Action Parameters in Steps in the Expert View" on page 728.

**How to Use User-Defined External Environment Variables**

This task describes how to create an external variable file and use it in your test.

This task includes the following steps:

- "Create an external environment variables file" on page 754
- "(Quality Center users only) Upload the environment resource file to your Quality Center project" on page 754
- "Use environment variables in your test" on page 754
- "Results" on page 754

1 Create an external environment variables file

You can create an external environment variable file using the Test Settings dialog box or manually. For details, see "Create an external environment variable file manually" on page 755.

2 (Quality Center users only) Upload the environment resource file to your Quality Center project

- a In the Quality Center Test Resources module, create a new environment variable resource and then upload the .xml environment variable file you want to use with your test. For details, see the Quality Center or HP ALM user guide.
- b In QuickTest, connect to the Quality Center project. For details, see "HP ALM Connection Dialog Box" on page 1635.

3 Use environment variables in your test

- a Open the Environment pane in the Test Settings dialog box (**File > Settings > Environment** node). For details, see "Environment Pane (Test Settings Dialog Box)" on page 1483.
- b Select **User-defined** from the **Variable type** list.
- c Select the **Load variables and values from external file (reloaded each run session)** check box.
- d Use the browse button to the right of the **File** edit box, or enter the full path of the external environment variables file you want to use with your test. If your test is stored in Quality Center, you must select an file that is stored with your Quality Center project.

The variables defined in the selected file are displayed in the list of user-defined environment variables.

4 Results

You can now select the variables in the active file as external user-defined environment parameters in your test. For details, see "Parameter Options Dialog Box (Environment)" on page 766.

How to Create an External Environment Variables File

Note: This task is part of a higher-level task. For details, see "How to Use User-Defined External Environment Variables" on page 753.

You create an external environment variables file in one of the following ways:

- "Create an external environment variable file using the Test Settings dialog box" on page 755
- "Create an external environment variable file manually" on page 755

Create an external environment variable file using the Test Settings dialog box

- 1 Define the variables in the Environment pane of the Test Settings dialog box (**File > Settings > Environment** node).
- 2 Click Export to export the user-defined environment variables to an .xml file.

For details, see "Environment Pane (Test Settings Dialog Box)" on page 1483.

Create an external environment variable file manually

- 1 Create an .xml file using the editor of your choice.
You can use the QuickTest environment variable file schema in:
`<QuickTest Professional installation folder>\help\QTEnvironment.xsd` or
follow the formatting instructions below.
- 2 Enter `<Environment>` on the first line.

- 3 Enter each variable name-value pair in between <Variable> elements using 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 and not in QuickTest</Description>
</Variable>
```

- 4 Enter </Environment> on the last line.

Example:

```
<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.

Reference

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 details on data table sheets, see Chapter 38, "Data Table Pane."



Built in Environment Variables

The following built-in environment variables are available:

Name	Description
ActionIteration	The action iteration currently running.
ControllerHostName	The name of the controller's computer. This variable is relevant only when running as a GUI Vuser from the LoadRunner controller.
GroupName	The name of the group in the running scenario. This variable is relevant only when running as a GUI Vuser from the LoadRunner controller.
LocalHostName	The local host name.
OS	The operating system.
OSVersion	The operating system version.
ProductDir	The folder path where the product is installed.
ProductName	The product name.
ProductVer	The product version.
ResultDir	<p>The path of the folder in which the current run results are located.</p> <p>Note: You cannot use the ResultDir environment variable when running a test from Business Availability Center, LoadRunner, or the Silent Test Runner in QuickTest.</p>
Scenarioid	The identification number of the scenario. This variable is relevant only when running as a GUI Vuser from the LoadRunner controller.
SystemTempDir	The system temporary directory.
TestDir	The path of the folder in which the test is located.
TestIteration	The test iteration currently running.
TestName	The name of the test.

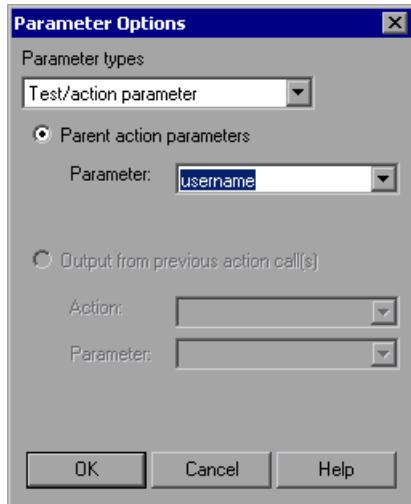
Name	Description
UpdatingActiveScreen	Indicates whether the Active Screen images and values are being updated during the update run process. For details, see "Update Options Tab (Update Run Dialog Box)" on page 1271.
UpdatingCheckpoints	Indicates whether checkpoints are being updated during the update run process. For details, see "Update Options Tab (Update Run Dialog Box)" on page 1271.
UpdatingTODescriptions	Indicates whether the set of properties used to identify test objects are being updated during the update run process. For details, see "Update Options Tab (Update Run Dialog Box)" on page 1271.
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.

For more details, see the section on built-in environment variables in "Environment Variable Parameters" on page 734.

Parameter Options Dialog Box (Test/Action Parameter)

This dialog box enables you to define settings for a test or action parameter.

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 value storage locations provide similar options.



To access	<ul style="list-style-type: none"> ▶ For argument values: In the keyword view, click the Configure the value button . ▶ For object property values: In the Object Repository window, select the object property value you want to parameterize and click the Configure the value button . ▶ For checkpoints: In the Configure value area of the Checkpoint Properties dialog box, select the Parameter radio button and click the Parameter Options button . ▶ For output value storage locations: In the Object Repository window, select the output object property value you want to parameterize and click the Modify button in the Configure value area.
Important information	<p>You can also use test or action parameter variables using parameterization objects and methods in the Expert View. For details, see the <i>HP QuickTest Professional Object Model Reference</i>.</p>
Relevant tasks	<ul style="list-style-type: none"> ▶ "How to Parameterize Values for Operations or Local Objects" on page 750 ▶ "How to Parameterize a Checkpoint Property Value" on page 751
See also	<ul style="list-style-type: none"> ▶ "Test and Action Input Parameters" on page 727 ▶ "Parameterizing Values" on page 723 <p>This dialog box is similar to:</p> <ul style="list-style-type: none"> ▶ "Parameter Options Dialog Box (Data Table)" on page 763 ▶ "Parameter Options Dialog Box (Environment)" on page 766 ▶ "Parameter Options Dialog Box (Random Number)" on page 770

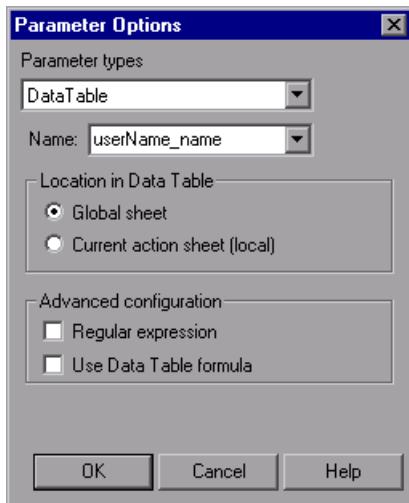
User interface elements are described below:

UI Elements	Description
Parameter types	<p>The type of parameter you want to use for the value. Make sure that Test/action parameter is selected.</p>
Parent action parameter/Test parameter	<p>Select this radio button if you want the parameter to take its value from an input parameter. The available radio button option depends on where you are defining the parameter:</p> <ul style="list-style-type: none"> ➤ Parent action parameter. Available for nested actions and all steps. ➤ Test parameter. Available for top-level actions only.
Parameter	<p>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.</p>
Output from previous action call(s)	<p>Select this radio button if you want the parameter to take its value from an output parameter. You can select from the output parameters of any previous action in the same hierarchical level as the current action, for which output parameters are defined.</p> <ul style="list-style-type: none"> ➤ 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).

Parameter Options Dialog Box (Data Table)

This dialog box enables you to define settings for a data table parameter.

The image 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.



To access	<ul style="list-style-type: none"> ➤ For argument values: In the Keyword View, click the Configure the value button . ➤ For object property values: In the Object Repository window, select the object property value you want to parameterize and click the Configure the value button . ➤ For checkpoints: In the Configure value area of the Checkpoint Properties dialog box, select the Parameter radio button and click the Parameter Options button . ➤ For output value storage locations: In the Object Repository window, select the output object property value you want to parameterize and click the Modify button in the Configure value area.
-----------	--

Important information	<ul style="list-style-type: none"> ▶ You cannot select Use Data Table formula if Regular expression is selected. ▶ The use of complex and/or nested formulas in the data table is not supported. ▶ You can also define data table parameters using parameterization objects and methods in the Expert View. For details, see the <i>HP QuickTest Professional Object Model Reference</i>.
Relevant tasks	<ul style="list-style-type: none"> ▶ "How to Parameterize Values for Operations or Local Objects" on page 750 ▶ "How to Parameterize a Checkpoint Property Value" on page 751
See also	<ul style="list-style-type: none"> ▶ "Formulas in Data Tables" on page 1332 ▶ "Default Parameter Values" on page 757 <p>This dialog box is similar to:</p> <ul style="list-style-type: none"> ▶ "Parameter Options Dialog Box (Test/Action Parameter)" on page 760 ▶ "Parameter Options Dialog Box (Environment)" on page 766 ▶ "Parameter Options Dialog Box (Random Number)" on page 770

User interface elements are described below:

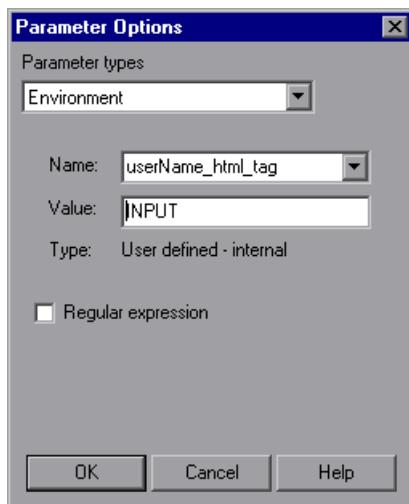
UI Elements	Description
Parameter types	The type of parameter you want to use for the value. Make sure that Data Table is selected.

UI Elements	Description
Name	<p>The name of the parameter to use. The following options are available:</p> <ul style="list-style-type: none"> ➤ To use an existing parameter, select it from the list. ➤ To create a new parameter, either use the default parameter name or enter a descriptive name for the parameter. <p>Note: For a list of naming conventions, see "Naming Conventions" on page 1779. 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).</p>
Location in Data Table	<ul style="list-style-type: none"> ➤ Global sheet. You store data in the Global tab when you want it to be available to all actions in your test and you want the data to control the number of test iterations. ➤ Current action sheet (local). You store data in the action's tab when you want to use the data in data table parameters for that action only and you want the data to control the number of action iterations. <p>For details, see "When to Choose Global or Action Data Table Parameters" on page 737.</p>
Regular expression	<p>Enables you to set the value of the parameter as a regular expression. For details, see "Regular Expressions Overview" on page 863.</p> <p>Note: This option is available only when parameterizing checkpoint and object property values.</p>
Use Data Table formula	<p>Enables you to use formulas predefined in the data table. For details on setting formulas, see "Formulas in Data Tables" on page 1332.</p> <p>For checkpoints, QuickTest inserts two columns in the data table. The first column contains a formula that checks the validity of the 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.</p>

Parameter Options Dialog Box (Environment)

This dialog box enables you to define settings for an environment parameter.

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.



To access	<ul style="list-style-type: none"> ➤ For argument values: In the keyword view, click the Configure the value button . ➤ For object property values: In the Object Repository window, select the object property value you want to parameterize and click the Configure the value button . ➤ For checkpoints: In the Configure value area of the Checkpoint Properties dialog box, select the Parameter radio button and click the Parameter Options button . ➤ For output value storage locations: In the Object Repository window, select the output object property value you want to parameterize and click the Modify button in the Configure value area.
------------------	--

Important information	<ul style="list-style-type: none">➤ 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.➤ You can also define environment variables using parameterization objects and methods in the Expert View. For details, see the <i>HP QuickTest Professional Object Model Reference</i>.
Relevant tasks	<ul style="list-style-type: none">➤ "How to Parameterize Values for Operations or Local Objects" on page 750➤ "How to Parameterize a Checkpoint Property Value" on page 751➤ "How to Use User-Defined External Environment Variables" on page 753
See also	<ul style="list-style-type: none">➤ "Environment Variable Parameters" on page 734 <p>This dialog box is similar to:</p> <ul style="list-style-type: none">➤ "Parameter Options Dialog Box (Test/Action Parameter)" on page 760➤ "Parameter Options Dialog Box (Data Table)" on page 763➤ "Parameter Options Dialog Box (Random Number)" on page 770

User interface elements are described below:

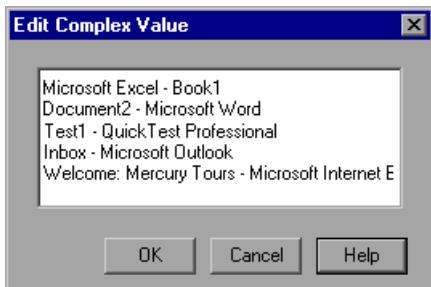
UI Elements	Description
Parameter types	<p>The type of parameter you want to use for the value. Make sure that Environment is selected.</p>
Name	<p>The name of the parameter. For an internal user-defined environment variable parameter, you can create a new parameter by using the default parameter name or entering a new, descriptive name. Alternatively, you can select an existing internal user-defined environment variable parameter from the list.</p> <p>Note:</p> <ul style="list-style-type: none"> ➤ 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. For details on modifying existing environment variable parameters, see "Environment Pane (Test Settings Dialog Box)" on page 1483. ➤ 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	<p>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.</p> <p>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.</p> <p>You can view or edit a complex value by clicking the View/Edit Complex Value button . For details, see "Edit Complex Value Dialog Box" on page 769.</p>

UI Elements	Description
Type	<p>The type of environment variable parameter (read-only):</p> <ul style="list-style-type: none"> ➤ internal user-defined ➤ external user-defined ➤ built-in
Regular expression	<p>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 details on regular expressions, see "Regular Expressions Overview" on page 863.</p>



Edit Complex Value Dialog Box

This dialog box enables you to edit a parameter value that cannot be displayed in the Parameter Options dialog box.



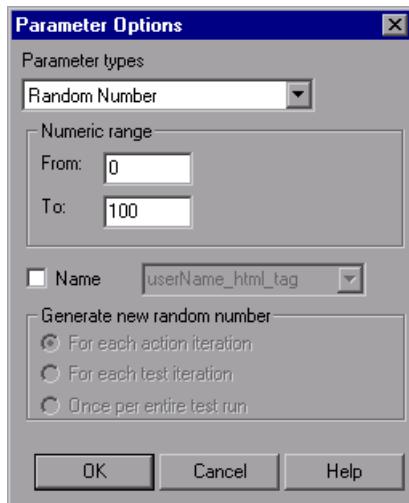
To access	<p>The Parameters Options dialog box displays the value of the parameter in the Value box. If the value cannot be entirely displayed in the Value box (such as a list), click the View/Edit Complex Value button to open the Edit Complex Value dialog box.</p>
------------------	---

Important information	<ul style="list-style-type: none"> ➤ Internal user-defined environment variable parameter. You can edit the value. ➤ External or built-in environment variable parameter. You can view the value but you cannot modify it in this dialog box.
Relevant tasks	"How to Use User-Defined External Environment Variables" on page 753
See also	"Environment Variable Parameters" on page 734

Parameter Options Dialog Box (Random Number)

This dialog box enables you to define settings for a random number parameter.

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.



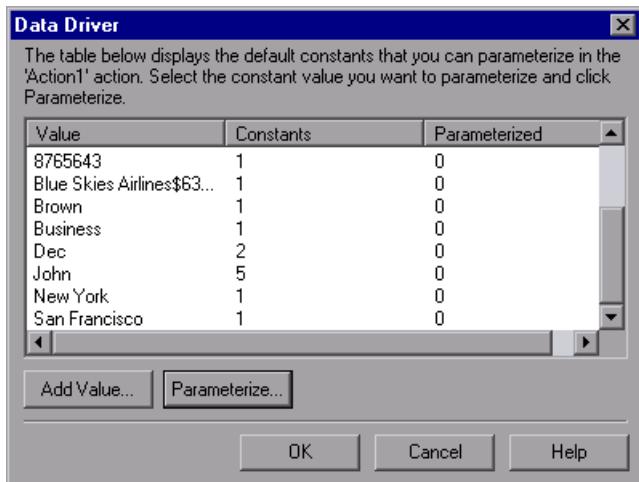
To access	<ul style="list-style-type: none"> ▶ For argument values: In the keyword view, click the Configure the value button . ▶ For object property values: In the Object Repository window, select the object property value you want to parameterize and click the Configure the value button . ▶ For checkpoints: In the Configure value area of the Checkpoint Properties dialog box, select the Parameter radio button and click the Parameter Options button . ▶ For output value storage locations: In the Object Repository window, select the output object property value you want to parameterize and click the Modify button in the Configure value area.
Important information	<ul style="list-style-type: none"> ▶ Random number parameters are not appropriate for non-numeric values, such as text or hypertext links. ▶ If you select an existing parameter in your test, when you modify the settings in this dialog box, all instances of that parameter are affected. ▶ You can also define random number variables using parameterization objects and methods in the Expert View. For details, see the <i>HP QuickTest Professional Object Model Reference</i>.
Relevant tasks	<ul style="list-style-type: none"> ▶ "How to Parameterize Values for Operations or Local Objects" on page 750 ▶ "How to Parameterize a Checkpoint Property Value" on page 751
See also	<ul style="list-style-type: none"> ▶ "Parameterizing Values Overview" <p>This dialog box is similar to:</p> <ul style="list-style-type: none"> ▶ "Parameter Options Dialog Box (Test/Action Parameter)" on page 760 ▶ "Parameter Options Dialog Box (Data Table)" on page 763 ▶ "Parameter Options Dialog Box (Environment)" on page 766

User interface elements are described below:

UI Elements	Description
Parameter types	The type of parameter you want to use for the value. Make sure that Random Number is selected.
Numeric range	<p>The range from which the random number is generated. You can modify the range by entering different values in the From and To boxes.</p> <p>Default range: 0 to 100 Minimum From value: 0 Maximum To value: 2147483647</p>
Name	The name of 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	<p>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:</p> <ul style="list-style-type: none"> ▶ For each action iteration. Generates a new number for each action iteration. ▶ For each test iteration. Generates a new number for 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 run session.

Data Driver Dialog Box

This dialog box displays a list of all the default constants for an action. For each constant value, it displays the number of times the constant value appears in the action.



To access	1 Display the action you want to parameterize. 2 Select Tools > Data Driver .
-----------	--

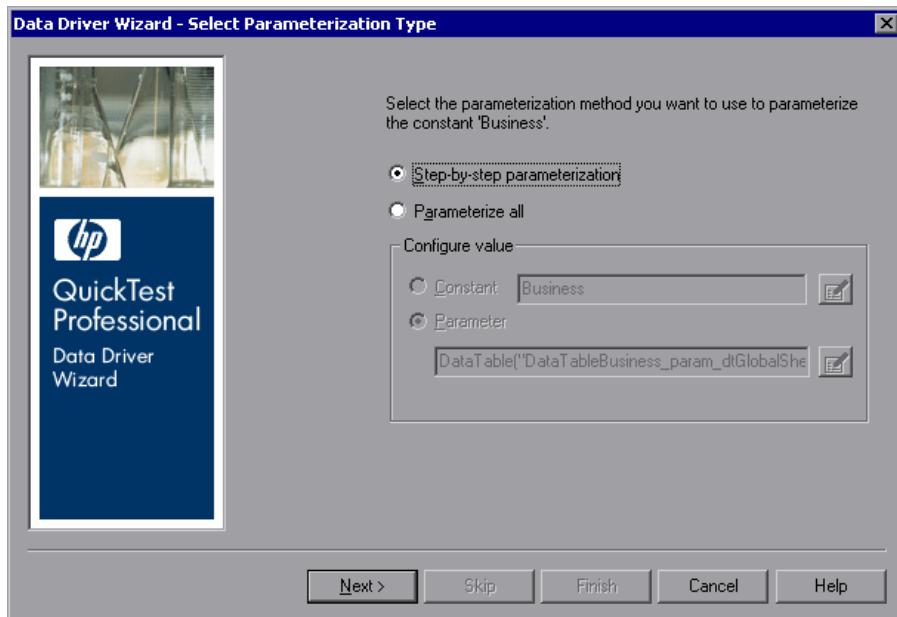
Important information	<p>► QuickTest scans the test for constants before the Data Driver opens. This may take a few moments. If the action being scanned contains a large number of lines and constant values, QuickTest warns you that loading the constants may take some time. You can choose whether to wait for the constants to load, or to open the Data Driver wizard quickly without constants.</p> <p>► 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 details on how to work with testing methods, see Chapter 27, "Working in the Expert View and Function Library Windows." For syntax and method information, see the <i>HP QuickTest Professional Object Model Reference</i>.</p>
Relevant tasks	<p>► "How to Parameterize Values for Operations or Local Objects" on page 750</p> <p>► "How to Parameterize a Checkpoint Property Value" on page 751</p>
See also	"Data Driver" on page 744

User interface elements are described below (unlabeled elements are shown in angle brackets):

UI Elements	Description
<constants list>	The list of default constant values as well as any constants you added during the current Data Driver session. The list also displays the number of times the constant value occurs in the action, and the number of occurrences that are set to be parametrized.
Add Value	Opens the Add Constant dialog box, which enables you to add a valid constant to the Constants list. Any object property or checkpoint value that exists in your action is a valid constant value. If you chose not to wait for the constants to load you can use this option to add the constant values that you want to parameterize to the Data Driver.
Parameterize	Opens the Data Driver Wizard, which enables you to parameterize all occurrences of the selected value. You can set the parameterization for each occurrence step-by-step, or you can parameterize all occurrences in a single operation.

Data Driver Wizard - Select Parameterization Type Page

This wizard enables you to select the parameterization method, and set parameterization options if you select the **Parameterize all** method.



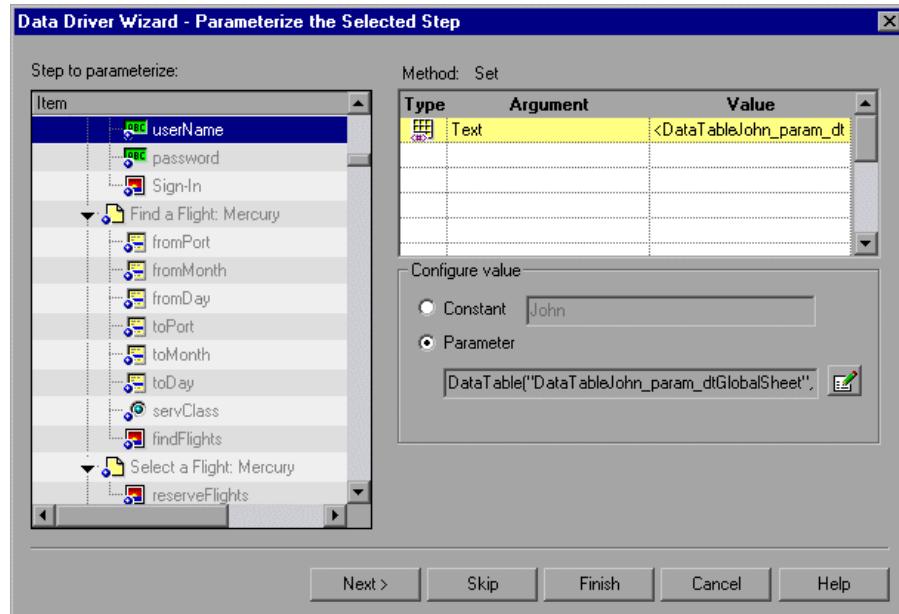
To access	1 Display the action you want to parameterize. 2 Select Tools > Data Driver and select the Parameterize button in the Data Driver dialog box.
Relevant tasks	<ul style="list-style-type: none"> ➤ "How to Parameterize Values for Operations or Local Objects" on page 750 ➤ "How to Parameterize a Checkpoint Property Value" on page 751
Wizard map	<p>This wizard contains:</p> <p>Data Driver Wizard - Select Parameterization Type Page > Data Driver Wizard - Parameterize the Selected Step Page (page 778)</p>

User interface elements are described below:

UI Elements	Description
Step-by-step parameterization	<p>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.</p> <p>The Next button is enabled when you select this option.</p>
Parameterize all	<p>Enables you to parameterize all occurrences of the selected value throughout the action.</p> <p>When you select this option the Configure value area is enabled.</p> <p>The Finish button is enabled when you select this option.</p>
Configure value	<p>Enables you to set the parameterization options you want to apply to all occurrences of the selected value.</p> <p>Select your parameterization preferences the same way that you would for an individual step. For details on setting parameterization options, see "Configure Value Area" on page 867.</p>
Next	<p>Opens the Data Driver Wizard - Parameterize the Selected Step Page.</p> <p>(Enabled only when Step-by-step parameterization is selected.)</p>
Skip	<p>Disabled for this page.</p>
Finish	<p>Saves your parameterization options and closes the Data Driver Wizard. The Data Driver Dialog Box displays the number of occurrences of the selected value to parameterize.</p> <p>Note: Your parameterization settings are not applied until you click OK on the Data Driver main page.</p>

Data Driver Wizard - Parameterize the Selected Step Page

This wizard page displays the steps in the current action and enables you to set your parameterization preferences for the selected step.



Wizard map	The Data Driver Wizard contains: Data Driver Wizard - Select Parameterization Type Page (page 776) > Data Driver Wizard - Parameterize the Selected Step Page
See also	"Data Driver" on page 744

User interface elements are described below (unlabeled elements are shown in angle brackets):

UI Elements	Description
Step to parameterize	Displays the steps in the current action. Highlights the first step with an object property or checkpoint value containing the constant value selected in the Data Driver Dialog Box.
<Step information>	The top right part of the dialog box displays information about the selected object property or checkpoint in the Step to parameterize area.
<Properties/arguments information> (for objects and checkpoints only)	<p>The Properties area displays the property to parameterize within the context of the other properties or arguments that apply to the selected step.</p> <p>Note: You can parameterize only the selected value. The other values are displayed for informative purposes only.</p>
Configure value area	<p>The default parameterization settings are selected for the value. To modify the settings click the Parameter Options button  to set the parameterization options you want to apply to this step.</p> <p>You select your parameterization preferences the same way that you would for an individual step. For details, see "Configure Value Area" on page 867.</p>
Next	Saves your parameterization preferences for the selected step, and opens the Parameterize the Selected Step page for the next occurrence of the selected value. If there are no remaining occurrences of the value, clicking Next opens the Finished page.

UI Elements	Description
Skip	The value of the selected step remains a constant (not parameterized), and the Parameterize the Selected Step page for the next occurrence of the selected value opens. If there are no remaining occurrences of the value, clicking Skip opens the Finished page.
Finish	Applies the parameterization settings of the current step to this step and all remaining steps containing the selected value. The Data Driver Wizard closes and the Data Driver main page shows how many occurrences you selected to parameterize and how many remain as constants.

23

Output Values

This chapter includes:

Concepts

- [Output Values Overview on page 782](#)

Tasks

- [How to Create or Modify a Standard Output Value Step on page 790](#)
- [How to Create or Modify a Table Output Value Step on page 793](#)
- [How to Create or Modify a Text or Text Area Output Value Step on page 794](#)
- [How to Create or Modify a Database Output Value Step on page 797](#)
- [How to Create or Modify an XML Output Value Step on page 798](#)
- [Reference on page 803](#)

Reference

- [Output Value Properties Dialog Box on page 803](#)
- [Table Output Value Properties Dialog Box \(Table Content Tab\) on page 812](#)
- [Table Output Value Properties Dialog Box \(Properties Tab\) on page 819](#)
- [Text / Text Area Output Value Properties Dialog Box on page 822](#)
- [Database Output Value Properties Dialog Box on page 830](#)
- [XML Output Properties Dialog Box on page 835](#)
- [Add Existing Output Value Dialog Box on page 841](#)

Concepts

Output Values Overview

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.

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 details, see "Parameterized Values in the Run Results" on page 1188.

For details about using output values for each add-in environment installed with QuickTest Professional, see "Supported Output Values" on page 1786.

This section also includes:

- "Output Value Categories" on page 783
- "Output Types and Settings" on page 786
- "Viewing and Editing Output Values" on page 787
- "Storing Output Values" on page 787

Output Value Categories

You can create the following categories of output values:

- Standard output values (described on page 783)
- Text and text area output values (described on page 784)
- Table output values (described on page 784)
- Database output values (described on page 785)
- XML output values (described on page 785)
- Existing output values (described on page 785)

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.
 - You can use standard output values to output text strings by specifying the **text** property of the object as an output value.
-

For task details, see "How to Create or Modify a Standard Output Value Step" on page 790.

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 task details, see "How to Create or Modify a Table Output Value Step" on page 793.

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).

Note: You can create a text area output value only while recording on Windows-based applications.

For task details, see "How to Create or Modify a Text or Text Area Output Value Step" on page 794.

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 task details, see "How to Create or Modify a Database Output Value Step" on page 797.

XML Output Values

You can use XML output values to capture and output the values of XML elements and attributes in XML documents.

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.

After the run session has finished, you can also view the captured data by opening the XML Output Value Results window from the Run Results Viewer. For details, see Chapter 31, "Run Results Viewer."

For task details, see "How to Create or Modify an XML Output Value Step" on page 798.

Existing Output Values

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.

For task details, see "Add Existing Output Value Dialog Box" on page 841.

 **Output Types and Settings**

In the Output Options dialog box you can set the output type and settings for each value, to 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 details, see "Default Output Definitions" on page 786.

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 Options Dialog Box (described on page 808).

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 details on creating output parameters for actions, see "How to Use Actions in Your Test" on page 542.

For details on data table sheets, see Chapter 38, "Data Table Pane."

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 details on the options available in the different Output Value Properties dialog boxes, see:

- "Output Value Properties Dialog Box" on page 803
- "Text / Text Area Output Value Properties Dialog Box" on page 822
- "Table Output Value Properties Dialog Box (Table Content Tab)" on page 812
- "Table Output Value Properties Dialog Box (Properties Tab)" on page 819
- "Database Output Value Properties Dialog Box" on page 830
- "XML Output Properties Dialog Box" on page 835

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:

- Test and action parameters (described on page 788)
- The run-time data table (described on page 788)
- Environment variables (described on page 789)

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 details on action parameters, see "Action Parameters" on page 531.

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.

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.

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.

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, a data table column, or an environment variable, the existing value is overwritten when the output value step runs. When the run session ends, the original value is restored.
 - You can output values only to internal user-defined environment variables and not to external or built-in environment variables, which are read-only.
-

Tasks

How to Create or Modify a Standard Output Value Step

If you are modifying an existing output value step proceed to the Set the options for the output value object step.

This task includes the following steps:

- "Prerequisites and considerations for inserting a new output value step" on page 790
- "Insert a new standard output value step while recording your test" on page 791
- "Insert a new output value step while editing your test" on page 792
- "Insert an existing output value while editing your test" on page 792
- "Set the options for the output value object" on page 792

Prerequisites and considerations for inserting a new output value step

Before inserting a new output value step, make sure to review all relevant information and perform all prerequisites:

Prerequisites	
Object visibility in application	During an editing session, make sure the object is visible in your application before inserting an output value step on it.
Active Screen	Make sure that the Active Screen contains sufficient data for the object for which you want to define an output value. For details, see "Active Screen Pane (Options Dialog Box)" on page 1434

Considerations	
Availability	<ul style="list-style-type: none"> ▶ Recording sessions ▶ Editing sessions ▶ Active Screen
Supported environments	"Supported Checkpoints and Output Values Per Add-in" on page 1783

Insert a new standard output value step while recording your test

1 Start a recording session.

2 Do one of the following:

- ▶ Select **Insert > Output Value > Standard Output Value**.
- ▶ Click **Insert Checkpoint or Output Value** down arrow and select **Standard Output Value**.



The pointer changes into a pointing hand. For details on using the pointing hand, see "Tips for Using the Pointing Hand" on page 157.



3 In your application, click the object for which you want to specify an output value. The Output Value Properties dialog box opens for most objects, and the Table Output Value Properties dialog box opens for **Table** items.

Note: If the location you clicked is associated with more than one object, the Object Selection dialog box opens, enabling you to select the object for which you want to specify an output value. For details, see "Object Selection Dialog Box" on page 155.

Insert a new output value step while editing your test

1 Do one of the following:



- Make sure the **Active Screen** button is selected, click a step whose Active Screen contains the object for which you want to specify an output value, and right-click the object for which you want to specify an output value and select **Insert Output Value**.
- Right-click the step in your test and select **Insert Output Value**.

If the location you clicked is associated with more than one object, the Object Selection dialog box opens. For details, see "Object Selection Dialog Box" on page 155.



- 2 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.

Insert an existing output value while editing your test

- 1 Select the step after which you want to insert the output value.
- 2 Select **Insert > Output Value > Existing Output Value**. The Add Existing Output Value Dialog Box opens (described on page 841), enabling you to select the test object from which you want to output values.

Set the options for the output value object

- For standard output values, specify the settings in the Output Value Properties dialog box. For details, see "Output Value Properties Dialog Box" on page 803.
- For **Table** objects, specify the settings in the Table Output Value Properties dialog box. For details, see "Table Output Value Properties Dialog Box (Table Content Tab)" on page 812 and "Table Output Value Properties Dialog Box (Properties Tab)" on page 819.
- For text strings or text areas, specify the settings in the Text / Text Area Output Value Properties dialog box. For details, see "Text / Text Area Output Value Properties Dialog Box" on page 822.

- For database objects, specify the settings in the Database Output Value Properties dialog box. For details, see "Database Output Value Properties Dialog Box" on page 830.
- For XML elements or values, specify the settings in the XML Output Properties dialog box. For details, see "XML Output Properties Dialog Box" on page 835.



How to Create or Modify a Table Output Value Step

If you are modifying an existing table output value step proceed to the Set the options for the table output value object step.

This task includes the following steps:

- "Prerequisites and considerations for inserting a new table output value step" on page 793
- "Insert a new table output value step" on page 794
- "Set the options for the table output value object" on page 794

Prerequisites and considerations for inserting a new table output value step

Before inserting a new table output value step, make sure to review all relevant information and perform all prerequisites:

Prerequisites	
Object visibility in application	During an editing session, make sure the object is visible in your application before inserting a table output value on it.
Considerations	
Availability	<ul style="list-style-type: none"> ➤ Recording sessions ➤ Editing sessions ➤ Active Screen
Supported environments	"Supported Checkpoints and Output Values Per Add-in" on page 1783

Insert a new table output value step

- 1 Insert a new output value step, as described in "How to Create or Modify a Standard Output Value Step" on page 790, selecting a table or grid object from your application.
- 2 (Optional) For certain objects in certain environments, before the Table Output Value Properties dialog box opens, the Define Row Range dialog box opens. You select the row range to output in this dialog box the same way you select the row range to check for table checkpoints. For details, see "Define/Modify Row Range Dialog Box" on page 652.

Set the options for the table output value object

- In the Table Output Value Properties dialog box, specify the settings for the table output value object. For details, see "Table Output Value Properties Dialog Box (Table Content Tab)" on page 812.
- In the Define/Modify Row Range dialog box, you modify the row range to output the same way you select the row range to check for table checkpoints. For details, see "Define/Modify Row Range Dialog Box" on page 652.

How to Create or Modify a Text or Text Area Output Value Step

If you are modifying an existing text or text area output value step proceed to the Set the options for the text or text area output value object step.

This task includes the following steps:

- "Prerequisites and considerations for inserting a new text or text area output value step" on page 795
- "Insert a new text or text area output value step" on page 796
- "Set the options for the text or text area output value object" on page 796

Prerequisites and considerations for inserting a new text or text area output value step

Before inserting a new text output value step, make sure to review all relevant information and perform all prerequisites:

Prerequisites	
Object visibility in application	During an editing session, make sure the object is visible in your application before inserting a text output value step on it.
Considerations	
Availability	<ul style="list-style-type: none"> ▶ Recording sessions ▶ Editing sessions ▶ Active Screen
Supported environments	"Supported Checkpoints and Output Values Per Add-in" on page 1783
Text recognition	Before you create a text or text area output value step for a Windows-based application, make sure you configure the required capture settings in the General > Text Recognition pane (Tools > Options > Text Recognition node). For more information, see "Text Recognition Pane (Options Dialog Box)" on page 1426 and "Text Recognition for Windows-Based Objects Overview" on page 848.
Text area selection	<ul style="list-style-type: none"> ▶ 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, make sure that the area defined is large enough so that the output text is always within its boundaries. For details, see "Text Recognition for Windows-Based Objects Overview" on page 848.

Insert a new text or text area output value step

Insert a new output value step, as described in "How to Create or Modify a Standard Output Value Step" on page 790, selecting a text string or text area in your application. The Text / Text Area Output Value Properties dialog box opens.

Note:

- ▶ To output text value while editing, you first highlight a text string in the Active Screen then right-click the string, and select **Insert Text Output Value**.
 - ▶ When you output a text area value, you first define the area containing the text you want QuickTest to check. When you select Text Area Output Value, the mouse turns into a crosshairs pointer. Click and drag the crosshairs pointer to define this area. Release the mouse button after outlining the area required. For more details, see the section on text area output values in "Output Value Categories" on page 783, and the important information in "Text / Text Area Output Value Properties Dialog Box" on page 822.
-

Tip: Hold down the left mouse button and use the arrow keys to make precise adjustments to the defined area.

Set the options for the text or text area output value object

In the Text / Text Area Output Value Properties Dialog Box, specify the settings for the output value object. For details, see "Text / Text Area Output Value Properties Dialog Box" on page 822.

How to Create or Modify a Database Output Value Step

If you are modifying an existing database output value step proceed to the "Set the options for the database output value object step".

This task includes the following steps:

- "Prerequisites and considerations for inserting a new database output value step" on page 797
- "Insert a new database output value step" on page 798
- "Set the options for the database output value object" on page 798

Prerequisites and considerations for inserting a new database output value step

Before inserting a new database output value step, make sure to review all relevant information and perform all prerequisites:

Prerequisites	
Object visibility in application	During an editing session, make sure the object is visible in your application before inserting a database output value step on it.
Considerations	
Availability	<ul style="list-style-type: none"> ➤ Recording sessions ➤ Editing sessions ➤ Active Screen
Supported environments	"Supported Checkpoints and Output Values Per Add-in" on page 1783

Insert a new database output value step

- 1** Insert a new output value step, as described in "How to Create or Modify a Standard Output Value Step", selecting the **Database Output Value** option. The Database Query Wizard opens.
- 2** Use the wizard to define a query to retrieve the data that you want to output, the same way you use the wizard to define a query for a database checkpoint. For details, see "Connect to Database Using ODBC Page (Database Query Wizard)".

After you finish defining your query, the Database Output Value Properties dialog box opens.

Set the options for the database output value object

In the Database Output Value Properties Dialog Box, specify the settings for the output value object. For details, see "Database Output Value Properties Dialog Box" on page 830.

How to Create or Modify an XML Output Value Step

If you are modifying an existing XML output value step proceed to the Set the options for the XML output value object step.

This task includes the following steps:

- "Considerations for inserting a new XML output value step" on page 799
- "Insert a new XML output value step" on page 799
- "Set the options for the XML output value object" on page 800

Considerations for inserting a new XML output value step

Before inserting a new XML output value step, make sure to review all relevant information:

Availability	<ul style="list-style-type: none"> ▶ Recording sessions ▶ Editing sessions (XML file and test object output values only)
Supported environments	"Supported Checkpoints and Output Values Per Add-in" on page 1783
XML output value types	<ul style="list-style-type: none"> ▶ XML output values for Web pages and Frames ▶ XML output values to directly access and retrieve specific XML files in your system ▶ XML test object output values to retrieve values of 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.

Insert a new XML output value step

- ▶ Insert a new output value step, as described in "How to Create or Modify a Standard Output Value Step" on page 790, selecting an XML object from your application. The XML Output Properties Dialog Box opens (as described on page 790).
- ▶ If you are inserting an XML output value step on an XML file or an XML test object, the **XML Source Selection - Output Properties** dialog box opens. You use this dialog box the same way you use the XML Source Selection - Checkpoint Properties dialog box. For details, see "XML Source Selection - Checkpoint / Output Value Properties Dialog Box" on page 712.

Note: The **XML Output Value (From Application)** option is available only when the Web Add-in is installed and loaded. For details 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.

Set the options for the XML output value object

- ▶ In the XML Output Properties Dialog Box, specify the settings for the XML output value. For details, see "XML Output Properties Dialog Box" on page 835.
 - ▶ If you are outputting values from a WebService object, you may need to update the XML tree to populate it. For details, see "How to Update the XML Hierarchy for XML Test Object Operation Output Value Steps (For WebService Test Objects Only)" on page 801.
-

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 details on XML standards, see: <http://www.w3.org/XML/>

For details on namespace standards, see: <http://www.w3.org/TR/1999/REC-xml-names-19990114/>

How to Update the XML Hierarchy for XML Test Object Operation Output Value Steps (For WebService Test Objects Only)

Note: 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.

This task describes the different ways you can update the XML hierarchy for XML test object operation output value steps.

This task includes the following steps:

- "Update the XML Tree Manually" on page 802
- "Import an XML Tree from a File" on page 802
- "Update the XML Tree Using Update Run Mode" on page 802

Update the XML Tree Manually

Use the options in the XML Output Properties Dialog Box (as described on page 835) to update the XML tree by adding elements, attributes, and values.

Import an XML Tree from a File



- 1 Use the **Import XML** button in the XML Output Properties Dialog Box (as described on page 835) to replace a node in the XML tree.
- 2 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 22, "Parameterizing Values."

Update the XML Tree Using Update Run Mode



- 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 output value step.
- 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.
- If you want to confirm that QuickTest successfully updated your output value step, expand the tree in the Run Results Viewer and select the XML output value step. Then check that **Update done** is displayed in the right-hand pane. (If the Run Results Viewer did not open automatically at the end of the run, click the **Results** button or select **Automation > Results**.)

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.

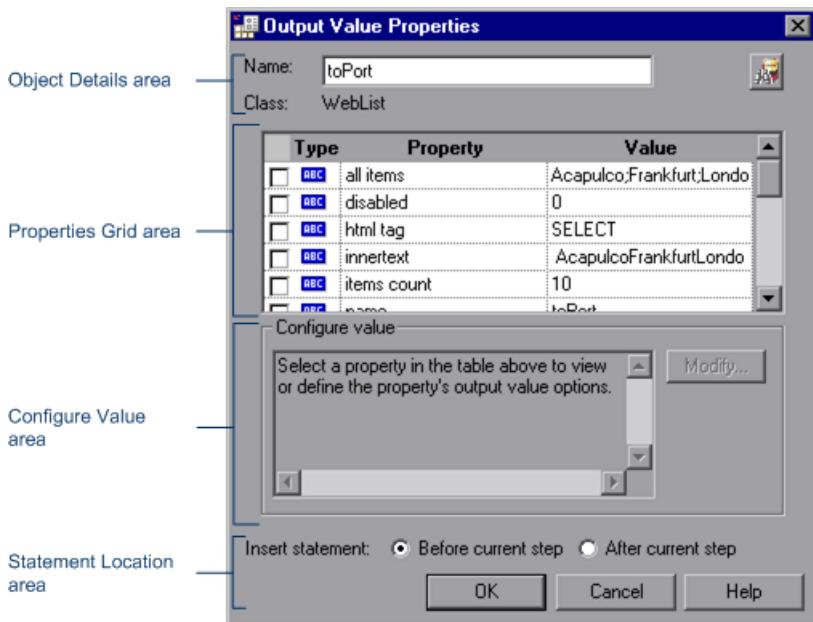
For details on using Update Run Mode, see "How to Update Test Object Descriptions, Checkpoints, Output Values, or Active Screens" on page 1252.

Reference

Output Value Properties Dialog Box

This dialog box enables you to define and modify properties for a standard output value.

The following image shows an example of the Output Value Properties dialog box when a WebList object is selected.



To access	Use one of the following: <ul style="list-style-type: none"> ➤ Insert a new output value step and select an object from your application. For details, see "How to Create or Modify a Standard Output Value Step" on page 790. ➤ In the Keyword View, right-click an existing output value step and select Output Value Properties. ➤ In the local or shared object repository, click an existing output value object. The output value details are displayed on the right side of the object repository dialog box, in the Object Details area.
Important information	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.
Relevant tasks	"How to Create or Modify a Standard Output Value Step" on page 790
See also	"Output Options Dialog Box" on page 808

Object Details Area

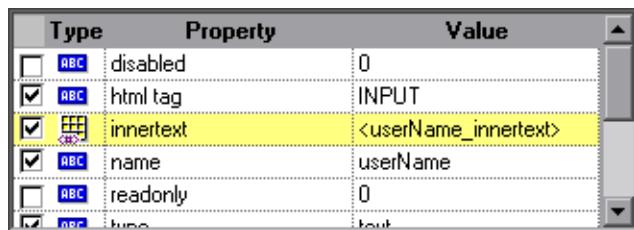


User interface elements are described below:

UI Elements	Description
Name	The name that QuickTest assigns to the output value object. By default, the name is the same as the name of the object on which the output value step is being performed. You can specify a different name for the output value object or accept the default name. For a list of naming conventions, see "Naming Conventions" on page 1779.
Class	The type of object (read-only).
	Find in Repository. Displays the output value object in its object repository. Available only when editing an existing output value step. It is not available when creating a new output value step.

Properties Grid Area

The following image shows the Properties Grid area when only some of the properties are selected. Specific properties may vary depending on the type of object for which you are defining output values.



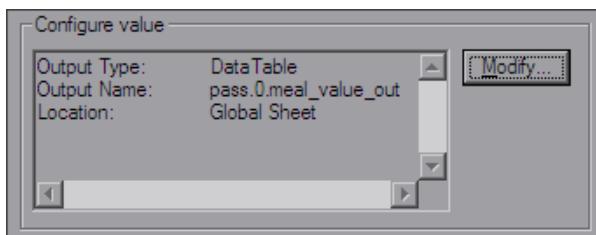
User interface elements are described below:

UI Elements	Description
Check box	Enables you to select one or more property for the object, and specify the output options for each property value you select.

UI Elements	Description
Type	<p>The  icon indicates that the value of the property is currently a constant.</p> <p>The  icon indicates that the value of the property is currently a test 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 "Configure Value Area" on page 867.

Configure Value Area

The following image shows the Configure Value area when the data table is used to store the output value.



User interface elements are described below:

UI Elements	Description
Configure value	The output definition for the selected property, in read-only mode.

UI Elements	Description
Modify	<p>Opens the Output Options dialog box, enabling you to change the output type and/or settings for the selected value.</p> <p>For details, see "Output Options Dialog Box" on page 808.</p>

Statement Location Area

The following image shows the statement location area when inserting a new output value step during an editing session. When inserting a new output value step during a recording session, the **Insert statement** option is not available.



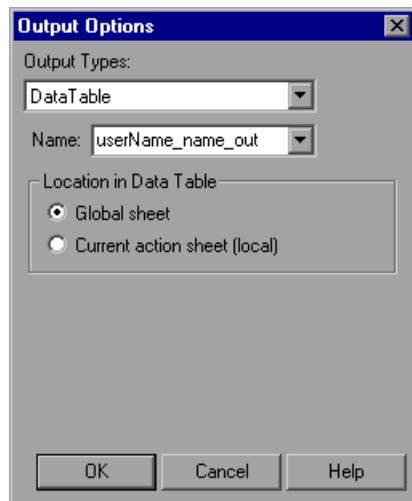
User interface elements are described below:

UI Elements	Description
Insert statement	<p>Specifies whether to insert the output value step before or after the currently selected step. The default value is Before current step.</p> <p>Note: Available only when inserting a new output value step during an editing session. During a recording session, the output value step is always inserted as the next step.</p>

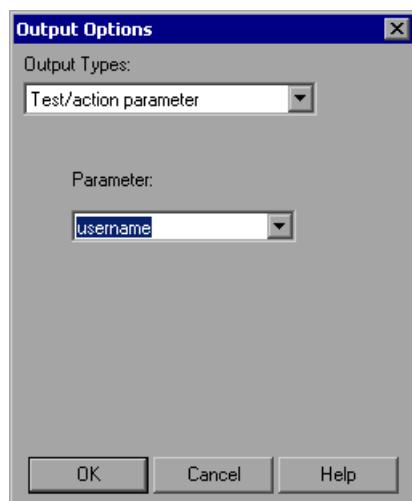
Output Options Dialog Box

This dialog box enables you to define the output type for the specified property.

Data Table



Test/Action Parameter



Environment Variable



To access	In the Configure Value area of any Output Value Properties dialog box, click the Modify button.
Important information	<ul style="list-style-type: none"> ► You can only output a value to an action parameter if the parameter has been defined as an output parameter for the calling action. ► If at least one output parameter is defined in the action, QuickTest may display Test/action parameter as the default output type. Otherwise, Data Table may be displayed as the default output type.
See also	<ul style="list-style-type: none"> ► "Output Value Properties Dialog Box" on page 803 ► "Table Output Value Properties Dialog Box (Table Content Tab)" on page 812 ► "Table Output Value Properties Dialog Box (Properties Tab)" on page 819 ► "Text / Text Area Output Value Properties Dialog Box" on page 822 ► "Database Output Value Properties Dialog Box" on page 830 ► "XML Output Properties Dialog Box" on page 835

Data Table Option

UI Elements	Description
Data Table Output type	Enables you to output values to the run-time data table. This enables you to store and retrieve values during a data-driven test (or action) that runs several times. In each repetition, or iteration , QuickTest stores the value in a different row within the data table.
Name	<p>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.</p> <p>For a list of naming conventions, see "Naming Conventions" on page 1779.</p>
Location in Data Table	<p>Specifies whether to add the Data Table parameter (column name) in the global or current action sheet in the data table.</p> <p>For details on the use of data in the global and current action sheets, see "Action and Test Iterations Using the Data Table" on page 526.</p> <p>For details on actions, see Chapter 14, "Actions".</p> <p>For details on outputting values to the data table, see "Run-time Data Table".</p>

Test/Action Parameter Option

UI Elements	Description
Test/action parameter Output type	<p>Enables you to 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.</p> <p>For details on outputting a value to a test or action parameter, see "Storing Output Values".</p> <p>For details on creating and using parameters, see Chapter 22, "Parameterizing Values."</p>

UI Elements	Description
Parameter	<p>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.</p> <p>Tip: You can resize the display, as needed, and, if the list of parameters is long, you can scroll through the list.</p>

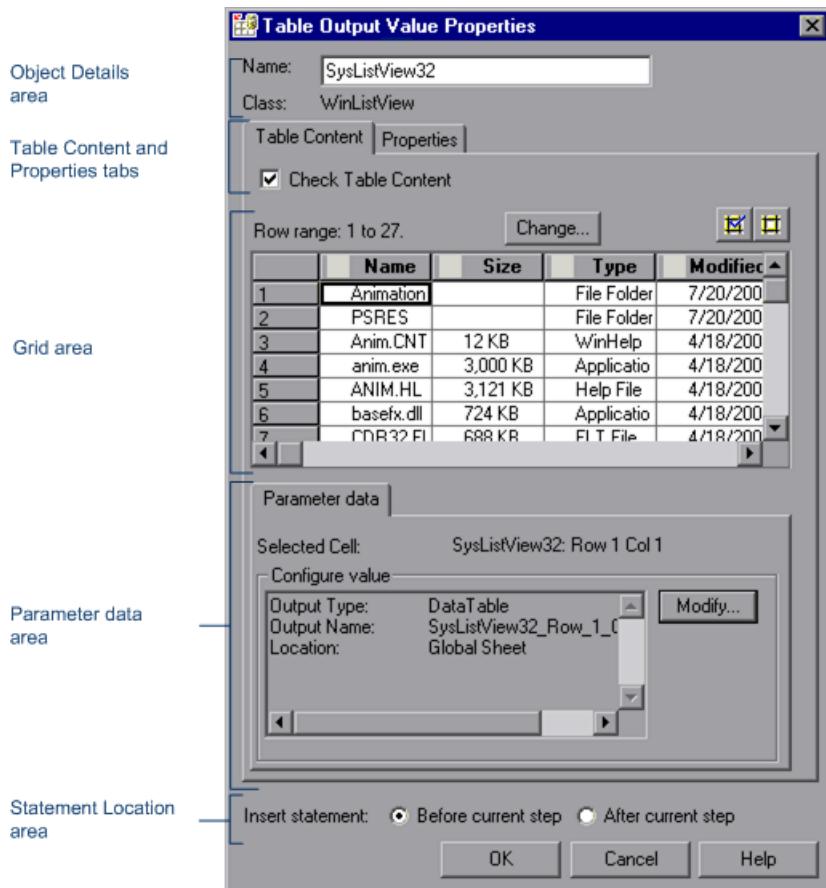
Environment Variable Option

UI Elements	Description
Environment Output type	<p>Enables you to specify the internal user-defined environment variable in which to store the selected value for the duration of the run session.</p>
Name	<p>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.</p> <p>Note: 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.</p>
Type	<p>The environment variable type. Because it is not possible to output values to external or built-in environment variables, the type is always User-defined - internal.</p> <p>For details on environment variables, see "Environment Variable Parameters" on page 734.</p>

Table Output Value Properties Dialog Box (Table Content Tab)

This dialog box enables you to define and modify properties for a table output value object. For some environments, this dialog box also enables you to output the properties of the object.

The following image shows an example of the table output value properties dialog box when outputting values for a WinListView object.

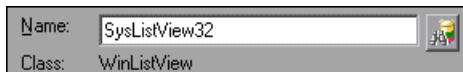


You can specify the table cells whose content you want to output. Depending on the environment, you do this either in the Table Content tab of the Table Output Value Properties dialog box—or directly in the Table Output Value Properties dialog box, if the dialog box does not contain any tabs.

To access	Do one of the following: <ul style="list-style-type: none"> ▶ Insert a new output value step and select a table object from your application. For details, see "How to Create or Modify a Table Output Value Step" on page 793. ▶ In the Keyword View, right-click an existing output value step and select Output Value Properties. ▶ In the local or shared object repository, click an existing output value object. The output value details are displayed on the right side of the object repository dialog box, in the Object Details area.
Important information	<ul style="list-style-type: none"> ▶ Most of the elements described in this section are available regardless of whether the Table Output Value Properties dialog box contains tabs. ▶ If the Table Output Value Properties dialog box contains tabs, you use the Table Content tab to check table content.
Relevant tasks	"How to Create or Modify a Table Output Value Step" on page 793
See also	<ul style="list-style-type: none"> ▶ "Define/Modify Row Range Dialog Box" on page 652 ▶ "Output Options Dialog Box" on page 808

Object Details Area

The following image shows the Object Details area when a WinListView object is selected during an editing session.



User interface elements are described below:

UI Element	Description
Name	<p>The name that QuickTest assigns to the output value object. By default, the name is the same as the name of the object on which the output value step is being performed. You can specify a different name for the output value object or accept the default name.</p> <p>For a list of naming conventions, see "Naming Conventions" on page 1779.</p>
Class	The type of object (read-only).
	Find in Repository. Displays the output value object in its object repository. Available only when editing an existing output value step. It is not available when creating a new output value step.

Table Content and Properties Tabs

The following image shows the Table Content and Properties Tabs area when each tab is selected.



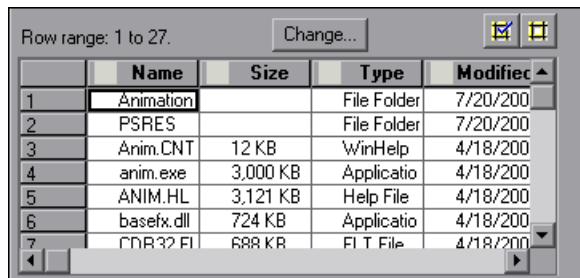
Important information	<ul style="list-style-type: none"> ➤ The check boxes are displayed only if the Table Output Value Properties dialog box contains tabs. ➤ If the Table Output Value Properties dialog box does not contain tabs, QuickTest automatically checks table content as defined in the dialog box.
------------------------------	--

User interface elements are described below:

UI Element	Description
Table Content	Displays the table content areas, which include: ► "Grid Area" on page 815 ► "Parameter Data Area" on page 817
Check Table Content	Instructs QuickTest to output the values of the table object. (Selected by default.)
Properties	Displays the object properties. For details, see "Table Output Value Properties Dialog Box (Properties Tab)" on page 819.
Check Properties	Instructs QuickTest to output the properties of the table object. (Cleared by default.)

Grid Area

The following image shows the Grid area when row range selection is enabled.



A screenshot of a Windows File Explorer window. The title bar says "Row range: 1 to 27." Below the title bar is a toolbar with a "Change..." button and two checkboxes. The main area is a grid table with columns labeled "Name", "Size", "Type", and "Modified". The first row has a checkbox in the first column, indicating it is selected. The grid contains the following data:

	Name	Size	Type	Modified
1	Animation		File Folder	7/20/200
2	PSRES		File Folder	7/20/200
3	Anim.CNT	12 KB	WinHelp	4/18/200
4	anim.exe	3,000 KB	Applicatio	4/18/200
5	ANIM.HL	3,121 KB	Help File	4/18/200
6	basefx.dll	724 KB	Applicatio	4/18/200
7	CDR32.FI	688 KB	FI T File	4/18/200

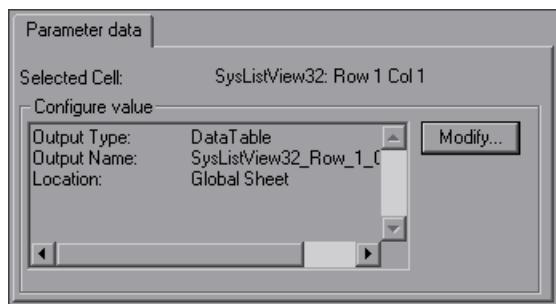
Important information	<ul style="list-style-type: none"> ➤ The column header names are captured from the table you selected for your output value. ➤ Double-clicking on the grid toggles the settings for all selected cells. Therefore, if you double-click a row header, a column header, or the top left corner of the grid, any cells that were previously included in the output are removed from it, and any cells that were not previously included in the output are added to it. ➤ You can change the column widths and row heights of the grid by dragging the column and row header dividers. ➤ If row range selection is supported, the row range you specify when creating the output value is displayed above the grid.
------------------------------	---

User interface elements are described below:

UI Element	Description
<grid area>	The grid area displays the captured\expected values of all the cells in the table. Only the ones with blue checkmarks are outputted. You can instruct QuickTest to output the entire table, specific rows, specific columns, or specific cells. Note: QuickTest checks only cells containing a check mark.
Change	Enables you to define or modify the row range to output by opening the "Define/Modify Row Range Dialog Box" on page 652.
	Add/Remove Output. Add or remove the selected cells from the output.

Parameter Data Area

The following image shows this area when the default output definition for the value is selected.



User interface elements are described below:

UI Element	Description
Selected Cell	The name of the object and location of the cell to output.
Configure value	The current output value settings for the selected cell. When you create a new output value, the default output definition is displayed for the value. For details, see "Default Output Definitions" on page 786.
Modify	Enables you to set parameterization and other preferences for each of the cells in your output value, by opening the Output Options Dialog Box (described on page 808).

Statement Location Area

The following image shows the statement location area when inserting a new output value step during an editing session. When inserting a new output value step during a recording session, the **Insert statement** option is not available.



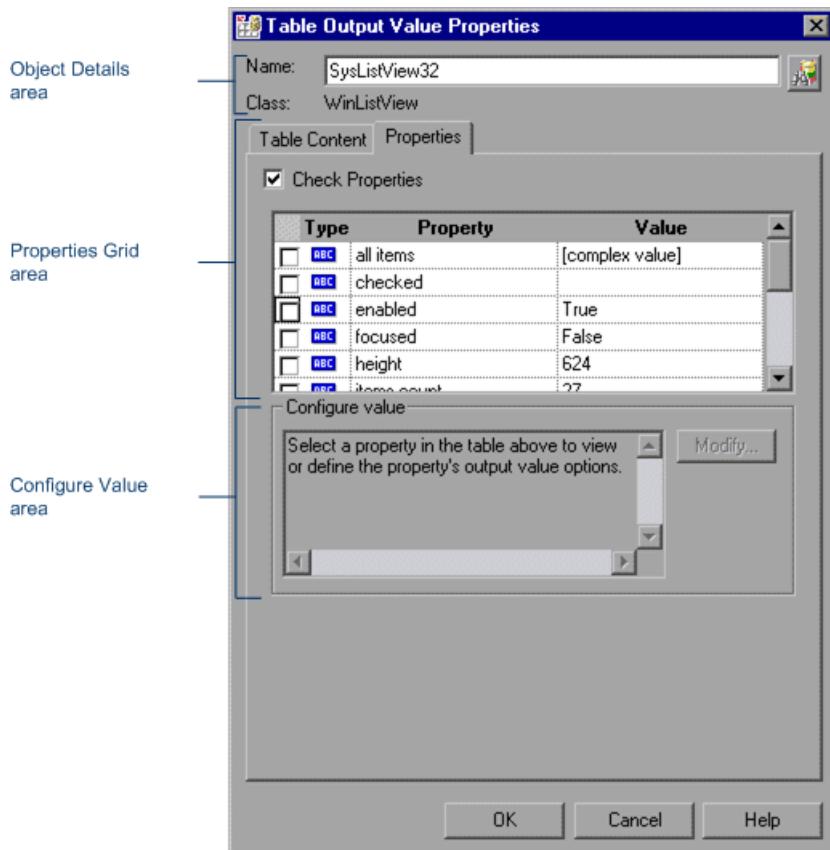
User interface elements are described below:

UI Element	Description
Insert statement	<p>Specifies whether to insert the output value step before or after the currently selected step. The default value is Before current step.</p> <p>Note: Available only when inserting a new output value step during an editing session. During a recording session, the output value step is always inserted as the next step.</p>

Table Output Value Properties Dialog Box (Properties Tab)

This tab enables you to specify which table (or grid) properties you want to output.

The following image shows an example of the Properties tab of the Table Output Value Properties dialog box when outputting values for a WinListView object.



To access	<p>1 Open the Table Output Value Properties dialog box by using one of the following:</p> <ul style="list-style-type: none"> ➤ Insert a new output value step and select a table object from your application. For details, see "How to Create or Modify a Table Output Value Step" on page 793. ➤ In the Keyword View, right-click an existing output value step and select Output Value Properties. ➤ In the local or shared object repository, click an existing output value object. The Output Value details are displayed on the right side of the object repository dialog box, in the Object Details area. <p>2 In the Table Output Value Properties dialog bog, select the Properties tab.</p>
Important information	<ul style="list-style-type: none"> ➤ This tab is available only for certain objects in certain environments. ➤ Selecting the Check Properties check box instructs QuickTest to output the properties of the table object. (Cleared by default.) ➤ 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. ➤ For details on general table output value options located outside the Properties tab, such as Name and Output Value timeout, see "Table Output Value Properties Dialog Box (Table Content Tab)" on page 812.
Relevant tasks	<p>"How to Create or Modify a Table Output Value Step" on page 793</p>

Object Details Area

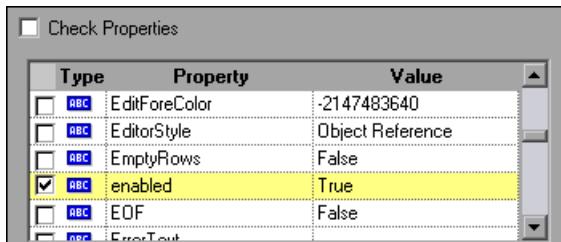
For a user interface description of this area, see "Object Details Area" on page 813.

Table Content and Properties Tabs

For a user interface description of this area, see "Table Content and Properties Tabs" on page 814.

Properties Grid Area

The Properties grid displays the table object's default properties, including the properties, their values, and their types. It is identical to the Properties Grid area of the Output Value Properties dialog box.



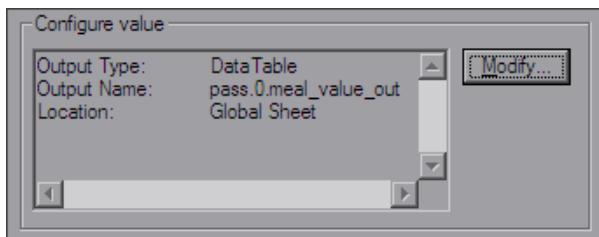
A screenshot of a Windows-style Properties Grid window. At the top left is a checkbox labeled "Check Properties". The main area is a table with three columns: "Type", "Property", and "Value". The "Type" column contains icons representing different property types. The "Property" column lists the names of the properties, and the "Value" column shows their current values. One row, "enabled", has a checked checkbox in the Type column and is highlighted with yellow background and green border. The other rows show "EditForeColor" (value: 2147483640), "EditorStyle" (value: Object Reference), "EmptyRows" (value: False), "EOF" (value: False), and "ErrorText" (value: False).

Type	Property	Value
	EditForeColor	2147483640
	EditorStyle	Object Reference
	EmptyRows	False
✓	enabled	True
	EOF	False
	ErrorText	

For user interface description, see "Properties Grid Area" on page 805.

Configure Value Area

The **Configure value** area enables you to define the expected value of the property as a **Constant** or a **Parameter**. It is identical to the Configure Value area of the Output Value Properties dialog box.



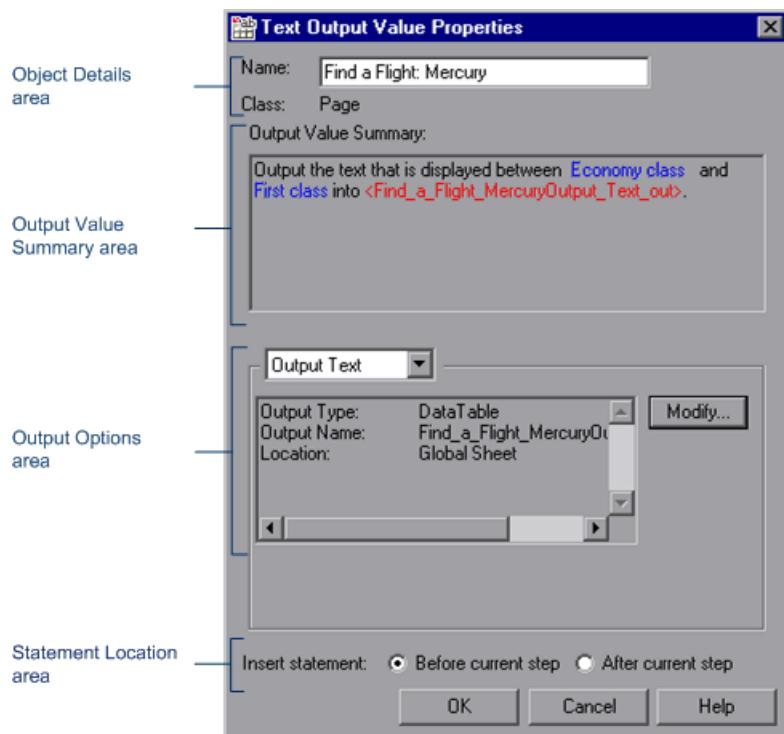
For user interface description, see "Configure Value Area" on page 806.

Text / Text Area Output Value Properties Dialog Box

The Text Output Value Properties and Text Area Output Value Properties dialog boxes enable you to define the output value settings for a selected text string, and to define the options for the text displayed before and after the selected 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 following image shows example of the Text or Text Area Properties dialog box when outputting a text value in an existing test during an editing session. The Text Output Value Properties dialog box options differ slightly during a recording session or when editing an existing output value. The Text Area Output Value Properties dialog box is similar to the Text Output Value Properties dialog box.



To access	<p>Use one of the following:</p> <ul style="list-style-type: none"> ➤ Insert a new output value step and select a text string or text area from your application. For details, see "How to Create or Modify a Text or Text Area Output Value Step" on page 794. ➤ In the Keyword View, right-click an existing output value step and select Output Value Properties. ➤ In the local or shared object repository, click an existing output value object. The output value details are displayed on the right side of the object repository dialog box, in the Object Details area.
Important information	<p>Consider the following when defining the area for a text area output value:</p> <ul style="list-style-type: none"> ➤ 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 (Output 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.
Relevant tasks	"How to Create or Modify a Text or Text Area Output Value Step" on page 794
See also	"Output Options Dialog Box" on page 808

Object Details Area

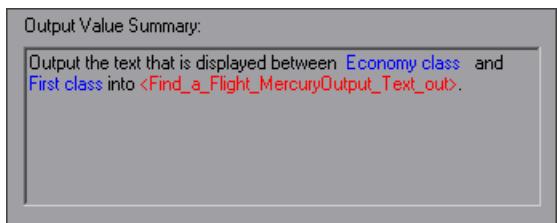


User interface elements are described below:

UI Element	Description
Name	The name that QuickTest assigns to the output value object. By default, the name is the same as the name of the object on which the output value step is being performed. You can specify a different name for the output value object or accept the default name. For a list of naming conventions, see "Naming Conventions" on page 1779.
Class	The type of object (read-only).
	Find in Repository. Displays the output value object in its object repository. Available only when editing an existing output value step. It is not available when creating a new output value step.

Output Value Summary Area

The following image shows an example of the Output Value Summary area when the output value is the text displayed between **Economy class** (the **Text Before** value) and **First class** (the **Text After** value).

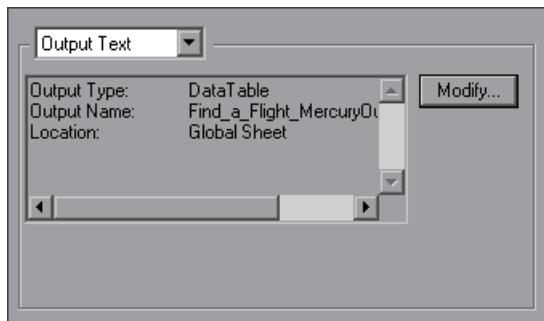


User interface elements are described below:

UI Element	Description
Output Value Summary	<p>Summarizes the selected text for the output value. It displays the text you selected when creating the output value, plus the text before and after it. QuickTest automatically displays the output text in red, and the text before and after the output text in blue.</p> <p>For text output values in Web-based environments, it displays the text you selected when creating the output value, plus some text before and after it. For text and text area output values in Windows-based environments, it displays the text you selected when creating the output value.</p> <p>Note:</p> <ul style="list-style-type: none">➤ In Windows-based environments, if there is more than one line of text selected, the Output Value 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 output value.➤ For a text area output value, the output value string contains all the text in the selected area. Therefore, 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.

Output Text Options Area

The following image shows an example of the output text options area with **Output Text** selected in the drop-down list.

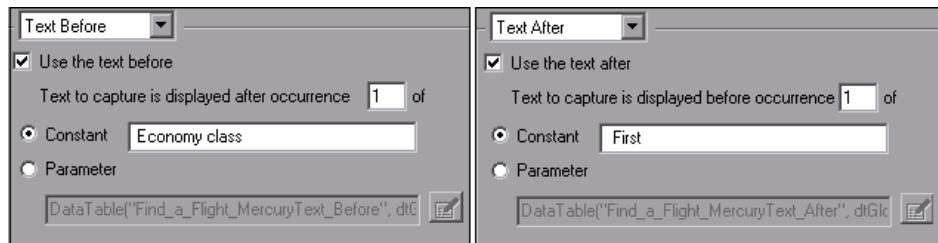


User interface elements are described below:

UI Element	Description
<Text type list>	Enables you to select the type of text for which to define output options. The following types are available: <ul style="list-style-type: none">► Output Text► Text Before► Text After
<Text Area list>	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 details, see "Default Output Definitions" on page 786.
Modify	Enables you to set parameterization and other preferences for each of the string elements in your output value, by opening the Output Options Dialog Box (described on page 808).

Text Before / Text After Options

The following image shows this area when the **Text Before** or **Text After** are selected from the list box.



User interface elements are described below:

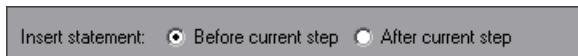
UI Element	Description
Use the text before / Use the text after	<p>When selected, the current Text Before / Text After value is displayed in the Constant box.</p> <p>When cleared, QuickTest retrieves the value of the first occurrence of the defined output string, regardless of the text displayed before 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>

UI Element	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>
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>

UI Element	Description
Parameter	Sets the Text Before or Text After value as a parameter. For details on specifying parameter values, see "How to Configure Constant and Parameter Values" on page 866.

Statement Location Area

The following image shows the statement location area when inserting a new output value step during an editing session. When inserting a new output value step during a recording session, the **Insert statement** option is not available.



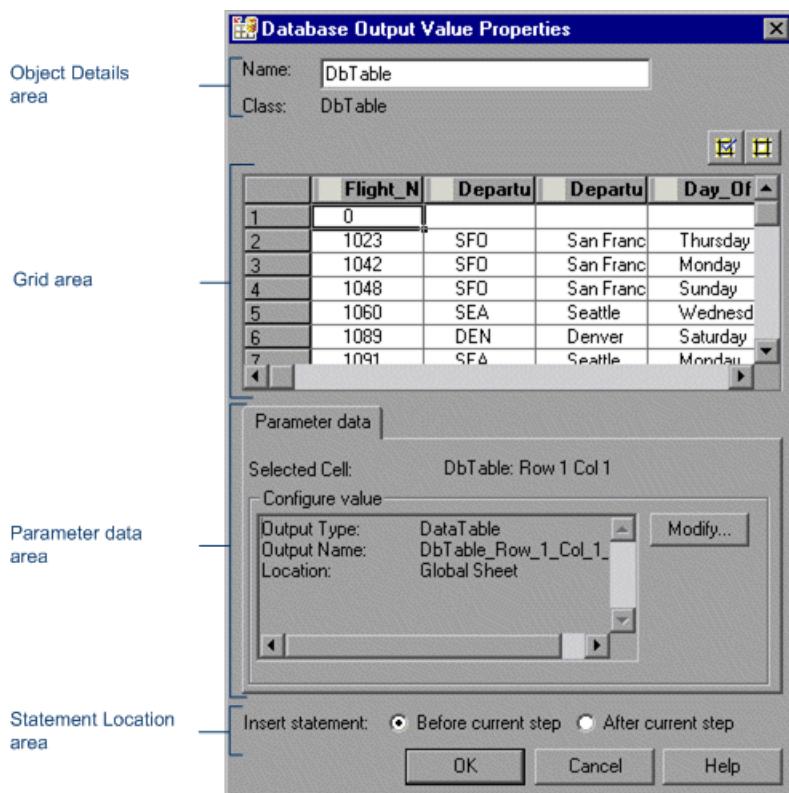
User interface elements are described below:

UI Element	Description
Insert statement	Specifies whether to insert the output value step before or after the currently selected step. The default value is Before current step . Note: Available only when inserting a new output value step during an editing session. During a recording session, the output value step is always inserted as the next step.

Database Output Value Properties Dialog Box

This 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.

The following image shows an example of the Properties tab of the Table Output Value Properties dialog box when outputting values for a DbTable object.



To access	<p>Do one of the following:</p> <ul style="list-style-type: none"> ➤ Insert a new output value step and select a database object from your application. For details, see "How to Create or Modify a Database Output Value Step" on page 797. ➤ In the Keyword View, right-click an existing output value step and select Output Value Properties. ➤ In the local or shared object repository, click an existing output value object. The Output Value details are displayed on the right side of the object repository dialog box, in the Object Details area.
Relevant tasks	"How to Create or Modify a Database Output Value Step" on page 797
See also	"Output Options Dialog Box" on page 808

Object Details Area

The following image shows the Object Details area when a DbTable object is selected during an editing session.

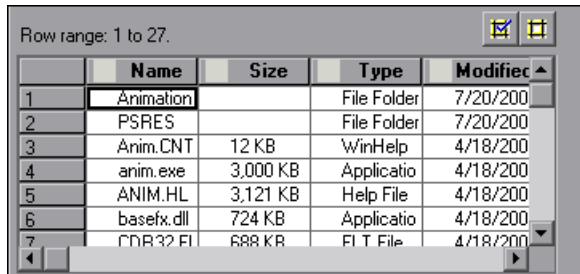


User interface elements are described below:

UI Element	Description
Name	The name that QuickTest assigns to the output value object. By default, the name is the same as the name of the object on which the output value step is being performed. You can specify a different name for the output value object or accept the default name. For a list of naming conventions, see "Naming Conventions" on page 1779.
Class	The type of object (read-only).
	Find in Repository. Displays the output value object in its object repository. Available only when editing an existing output value step. It is not available when creating a new output value step.

Grid Area

The following image shows the Grid area when the top left cell is highlighted.



A screenshot of a Windows-style dialog box titled "Define/Modify Row Range". The title bar has minimize, maximize, and close buttons. Below the title bar, it says "Row range: 1 to 27.". The main area is a grid table with the following columns: Name, Size, Type, and Modified. The first row (row 1) is selected, indicated by a blue border around the "Name" cell which contains "Animation". The data rows are as follows:

	Name	Size	Type	Modified
1	Animation		File Folder	7/20/200
2	PSRES		File Folder	7/20/200
3	Anim.CNT	12 KB	WinHelp	4/18/200
4	anim.exe	3,000 KB	Application	4/18/200
5	ANIM.HL	3,121 KB	Help File	4/18/200
6	basefx.dll	724 KB	Application	4/18/200
7	CDR32.FI	699 KB	Font File	4/18/200

Important information	<ul style="list-style-type: none"> ► 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 output are removed from it, and any cells that were not previously included in the output are added to it. ► You can change the column widths and row heights of the grid by dragging the column and row header dividers. ► If row range selection is supported, the row range you specify when creating the output value is displayed above the grid.
See also	"Define/Modify Row Range Dialog Box" on page 652

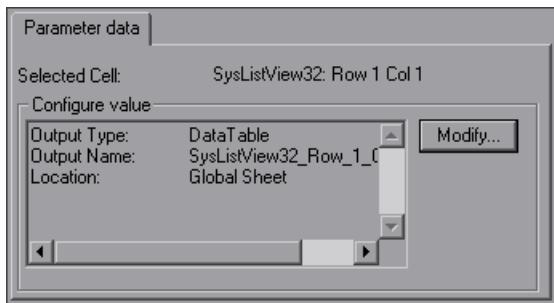
User interface elements are described below:

UI Element	Description
<grid area>	<p>The grid area displays the captured\expected values of all the cells in the table. Only the ones with blue checkmarks are outputted. You can instruct QuickTest to output the entire table, specific rows, specific columns, or specific cells.</p> <p>Note: QuickTest checks only cells containing a check mark.</p>

UI Element	Description
Change	Enables you to define or modify the row range to output by opening the "Define/Modify Row Range Dialog Box" on page 652.
	Add/Remove Output. Add or remove the selected cells from the output.

Parameter Data Area

The following image shows this area when the default output definition for the value is selected.



User interface elements are described below:

UI Element	Description
Selected Cell	The name of the object and location of the cell to output.
Configure value	The current output value settings for the selected cell. When you create a new output value, the default output definition is displayed for the value. For details, see "Default Output Definitions" on page 786.
Modify	Enables you to set parameterization and other preferences for each of the cells in your output value, by opening the Output Options Dialog Box (described on page 808).

Statement Location Area

The following image shows the statement location area when inserting a new output value step during an editing session. When inserting a new output value step during a recording session, the **Insert statement** option is not available.

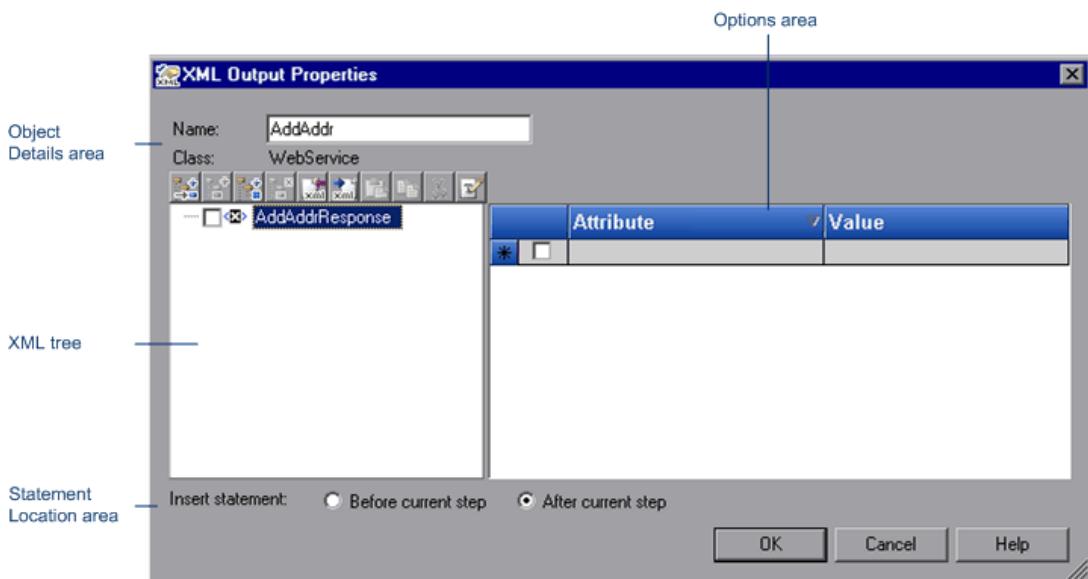


User interface elements are described below:

UI Element	Description
Insert statement	Specifies whether to insert the output value step before or after the currently selected step. The default value is Before current step . Note: Available only when inserting a new output value step during an editing session. During a recording session, the output value step is always inserted as the next step.

XML Output Properties Dialog Box

This 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.



To access	<p>Use one of the following:</p> <ul style="list-style-type: none"> ▶ Insert a new output value step and select a Web page or frame that contains an XML document. For details, see "How to Create or Modify an XML Output Value Step" on page 798. ▶ In the Keyword View, right-click an existing XML output value step and select Output Value Properties. ▶ In the local or shared object repository, click an existing XML output value object. The Output Value details are displayed on the right side of the object repository dialog box, in the Object Details area.
------------------	---

Relevant tasks	"How to Create or Modify an XML Output Value Step" on page 798
See also	"Output Options Dialog Box" on page 808

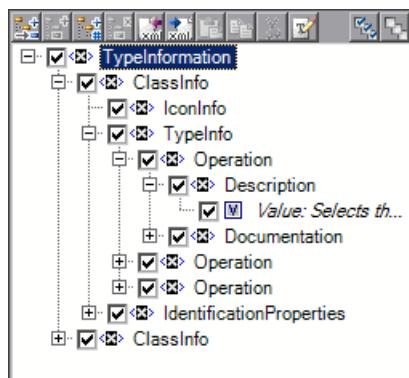
Object Details Area



User interface elements are described below:

UI Element	Description
Name	The name that QuickTest assigns to the XML output value object. By default, the name is the same as the name of the object on which the XML output value step is being performed. You can specify a different name for the XML output value object or accept the default name. For a list of naming conventions, see "Naming Conventions" on page 1779.
Class	The type of object (read-only).
	Find in Repository. Displays the output value in its object repository. Available only when editing an existing output value step. It is not available when creating a new output value.

XML Tree



Important information

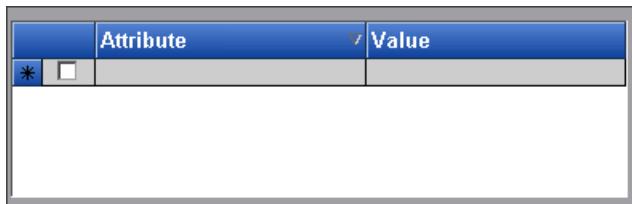
- The XML tree displays the hierarchical relationship between each element and value in the XML tree, enabling you to select the element and/or attribute values that you want to output. Each element node is displayed with a icon. Each value node is displayed with a icon.
- When you create an XML output 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 details, see "How to Update the XML Hierarchy for XML Test Object Operation Output Value Steps (For WebService Test Objects Only)" on page 801.
- The XML tree pane and the **Attribute** and **Value** columns in the right pane are resizable.
- 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.
- 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.

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 test editor. For details, see "Edit XML as Text Dialog Box" on page 710.

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>

Options Area



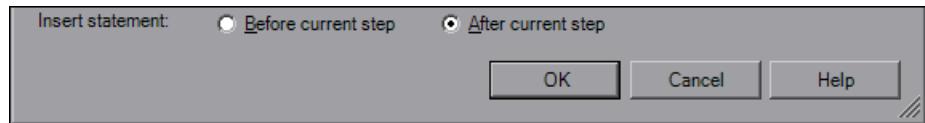
Important information	The Attribute and Value columns are resizable.
------------------------------	--

User interface elements are described below:

UI Element	Description
Attribute	The list of attributes for the selected element or value node in the XML tree.
Value	<p>The list of values for the selected element or value node in the XML tree.</p> <p>Note: 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. For details on the options available for each parameter type, see:</p> <ul style="list-style-type: none"> ➤ Data Table. "Data Table Parameters" on page 731. ➤ Environment. "Environment Variable Parameters" on page 734. ➤ Random Number. "Parameter Options Dialog Box (Random Number)" on page 770.

Statement Location Area

The following image shows the statement location area when inserting a new output value step during an editing session. When inserting a new output value step during a recording session, the **Insert statement** option is not available.



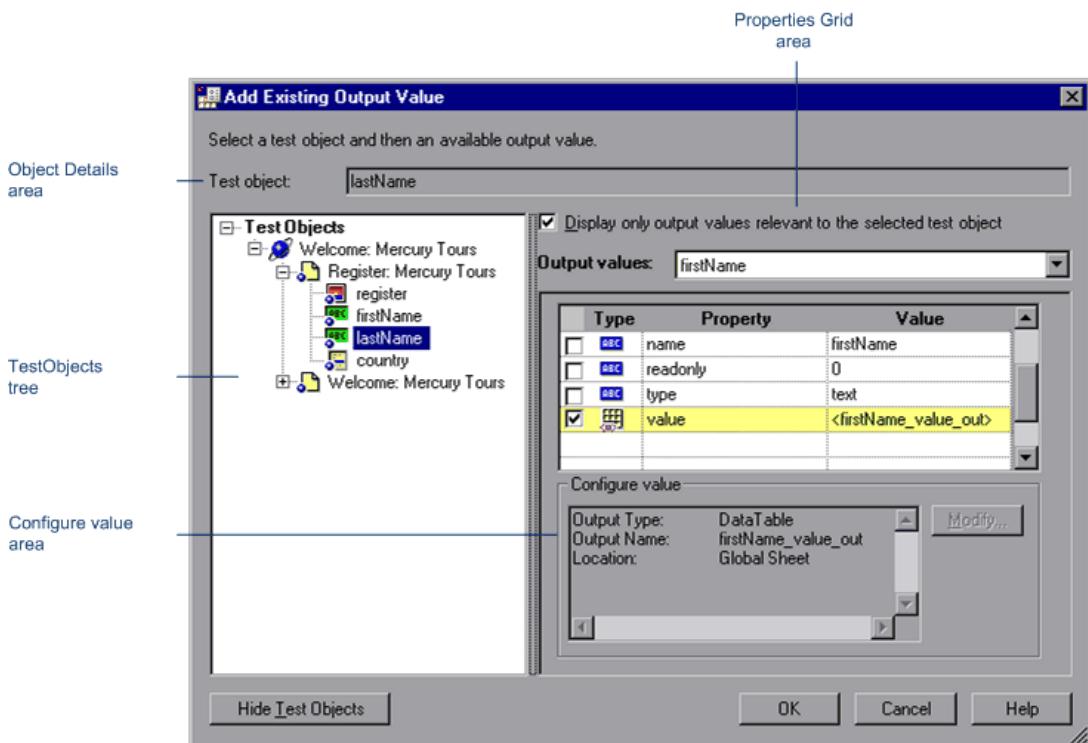
User interface elements are described below:

UI Element	Description
Insert statement	<p>Specifies whether to insert the output value step before or after the currently selected step. The default value is Before current step.</p> <p>Note: Available only when inserting a new output value step during an editing session. During a recording session, the output value step is always inserted as the next step.</p>

Add Existing Output Value Dialog Box

This dialog box enables you to select the test object to which you want to add an existing output value.

The following image shows an example of the Output Value Properties dialog box when the **TestObjects** tree is shown.



To access	Insert a new output value step and select the Existing Output Value option. For details, see "How to Create or Modify a Standard Output Value Step" on page 790.
Important information	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.

Relevant tasks	"How to Create or Modify a Standard Output Value Step" on page 790
See also	"Output Options Dialog Box" on page 808

Object Details Area

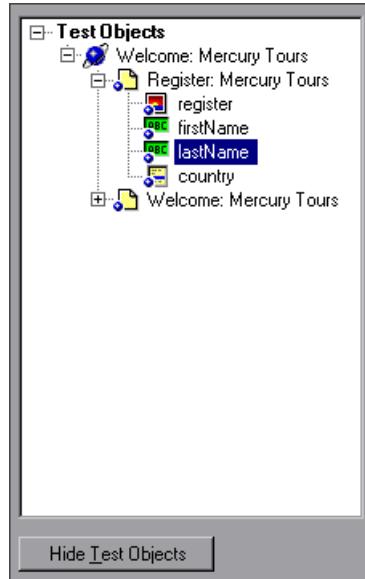
Select a test object and then an available output value.

Test object:

User interface elements are described below:

UI Element	Description
Test object	The test object for which you are adding an output value (read-only).

TestObjects Tree Area

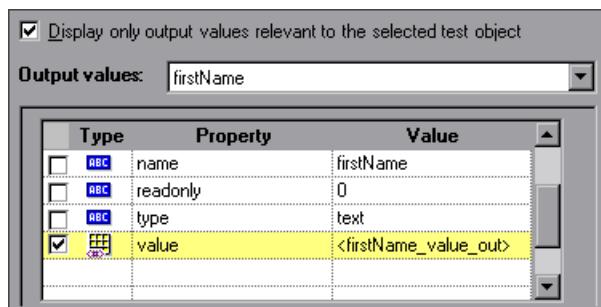


User interface elements are described below:

Option	Description
TestObjects tree	The objects stored in the object repositories associated with the current action.
Show/Hide Test Objects	Shows or hides the TestObjects tree.

Properties Grid Area

The following image shows the Properties Grid area when only part of the properties are selected. Specific properties may vary depending on the type of object for which you are defining output values.

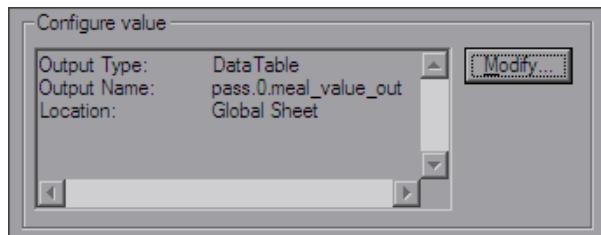


User interface elements are described below:

UI Element	Description
Display only output values relevant to the selected test object	<p>Instructs QuickTest to determine 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 display only those output value objects in the Output Values list.</p> <p>Note: 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 output values that can apply to that object.</p>
Output values	<p>The output values 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>
Check box	<p>Enables you to select one or more property for the object, and specify the output options for each property value you select.</p>
Type	<p>The  icon indicates that the value of the property is currently a constant.</p> <p>The  icon indicates that the value of the property is currently a test or action parameter.</p> <p>The  icon indicates that the value of the property is currently a Data Table parameter.</p> <p>The  icon indicates that the value of the property is currently an environment variable parameter.</p> <p>The  icon indicates that the value of the property is currently a random number parameter.</p>
Property	<p>The name of the property.</p>
Value	<p>The expected value of the property.</p>

Configure Value Area

The following image shows the Configure Value area when the data table is used to store the output value.



User interface elements are described below:

UI Element	Description
Configure value area	The output definition for the selected property, in read-only mode.
Modify	Opens the Output Options dialog box, enabling you to change the output type and/or settings for the selected value. For details, see "Output Options Dialog Box" on page 808.

24

Text Recognition for Windows-Based Objects

This chapter includes:

Concepts

- [Text Recognition for Windows-Based Objects Overview](#) on page 848
- [Checking Text in an Image - Use-Case Scenario](#) on page 848

Tasks

- [How to Configure Text Recognition Settings](#) on page 852

Reference

- [Guidelines for Text Recognition](#) on page 854
- [Text Recognition and Development Environments](#) on page 857

Concepts

Text Recognition for Windows-Based Objects Overview

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.

When you use one of these options, QuickTest identifies text in your application using either a Windows API-based mechanism or an OCR (optical character recognition) mechanism.

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.

When QuickTest uses the OCR mechanism, a number of factors can affect the text it retrieves. Depending on the characteristics of the text you want to retrieve, you can adjust several OCR configuration options to optimize the way the text is captured.

You use the Text Recognition Pane (Options Dialog Box) (described on page 1426) to specify the preferred text recognition mechanism and OCR-specific settings.

Checking Text in an Image - Use-Case Scenario

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.

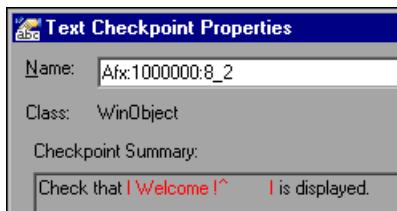
For one of his projects, Ben also needed to verify the text in the graphics, so he decided to use text checkpoints.

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:



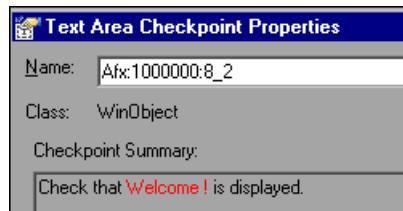
Ben noticed that there were extra characters in the **Checkpoint Summary** area of the text checkpoint, but he did not know why.

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.

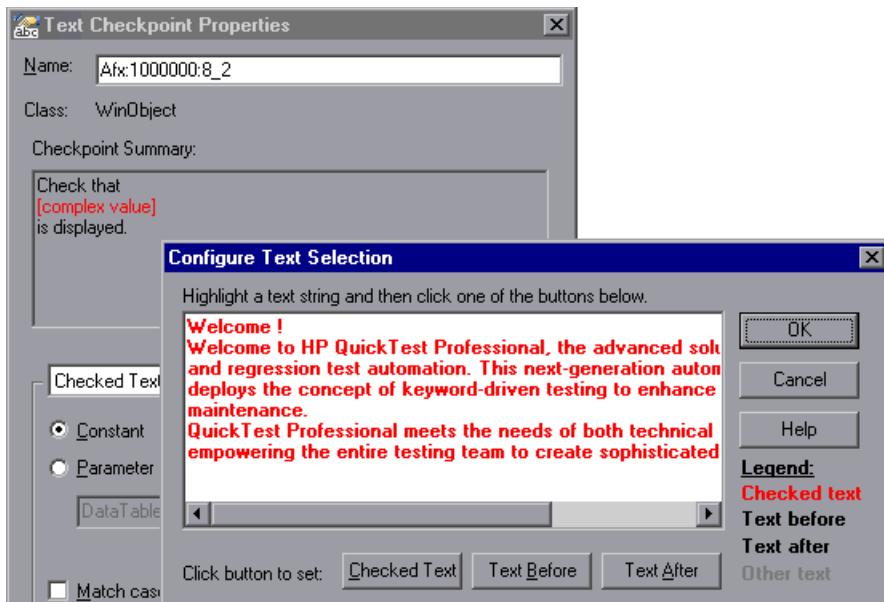
George also pointed out that the area Ben defined for the text checkpoint consisted of multiple text blocks because the text was not uniform in font size, color, or background. The title area consisted of white characters on a blue-gray background, while the remaining text was smaller and consisted of 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.

Tasks

How to Configure Text Recognition Settings

This task describes a suggested workflow for configuring QuickTest text recognition settings.

This task includes the following steps:

- "Prerequisites" on page 852
- "Analyze the characteristics of the text" on page 852
- "Set the appropriate options in the Text Recognition Pane" on page 853
- "Test the text recognition settings" on page 853
- "Adjust the settings as necessary" on page 853
- "Results" on page 853

1 Prerequisites

In your application, display the text you want to capture.

2 Analyze the characteristics of the text

Determine whether you can capture the text using a text (or text-like) property instead of using a text recognition mechanism.

If it must use text-recognition, analyze whether the Windows API or OCR mechanisms are more likely to meet your needs.

For details, see "Text Recognition for Windows-Based Objects Overview" on page 848 and "Guidelines for Text Recognition" on page 854.

3 Set the appropriate options in the Text Recognition Pane

For details, see "Text Recognition Pane (Options Dialog Box)" on page 1426.

4 Test the text recognition settings

Insert a text checkpoint, output value step, or a **GetText/GetVisibleText** method and run the step.

For details, see "How to Create or Modify a Text or Text Area Checkpoint Step" on page 657, "How to Create or Modify a Text or Text Area Output Value Step" on page 794, and the *HP QuickTest Professional Object Model Reference*.

5 Adjust the settings as necessary

If the captured text is not as expected, analyze the problems and adjust the Text Recognition options to fine tune the way QuickTest captures your text.

For more details, see "Text Recognition Pane (Options Dialog Box)" on page 1426 and "Guidelines for Text Recognition" on page 854.

6 Results

You can now use text checkpoints, output values, and **GetText/GetVisibleText** steps in your tests to capture this text from your application.

Reference

Guidelines for Text Recognition

- ▶ The Windows API text recognition mechanism is provided as is. If it does not work as you expect, change the option to use OCR.

However, although QuickTest uses a market-leading OCR mechanism and the accuracy is usually very high, keep in mind that as with all OCR technologies, these engines are non-deterministic in the way they are built and interpret text and therefore accuracy of recognition for some text patterns may be better than others. The OCR mechanism does not guarantee 100% accuracy.

Additionally, text-capturing steps may behave differently in different run sessions depending on the operating system version you are using, the installed service packs, other installed toolkits, the APIs used in your application, and so on.

- ▶ When using the OCR mechanism, the larger the text, the better the text recognition.
- ▶ When working with text area checkpoints or specifying the text area in methods such as **GetTextLocation**, 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 "Checking Text in an Image - Use-Case Scenario" on page 848.
-

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

For example, this may be useful if the default OCR settings result in additional unexpected characters in the captured text.

- On Windows Vista 32-bit or any 64-bit operating system, QuickTest text recognition features (such as text checkpoints and output values, **GetVisibleText** and **GetTextLocation** test object methods, and **TextUtil.GetText** and **TextUtil.GetTextLocation** reserved object methods) are limited and are not always reliable.

Workaround: On Windows Vista, you can improve text recognition by applying the Windows Vista 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. To do this, in the **General > Text Recognition** pane of the Options dialog box, select **Single text block mode**.
- 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 the **Important Information** section in "Text / Text Area Checkpoint Properties Dialog Box" on page 660 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 objects ▶ Text area output values for all 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 objects ➤ Text area output values for all 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
Silverlight	Full support except as listed in the Not Supported column.	<ul style="list-style-type: none"> ➤ GetTextLocation method
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

Development Environment	Text Recognition	
	Supported	Not Supported
VisualAge	Full text recognition support	N/A
Visual Basic	Full text recognition support	N/A
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 objects ▶ Text area output values for all objects ▶ GetTextLocation method ▶ GetVisibleText method
Web Services	N/A	No text recognition support
WPF	Full support except as listed in the Not Supported column.	<ul style="list-style-type: none"> ▶ GetTextLocation method

25

Value Configuration and Regular Expressions

This chapter includes:

Concepts

- ▶ [Value Configuration Overview on page 862](#)
- ▶ [Regular Expressions Overview on page 863](#)

Tasks

- ▶ [How to Configure Constant and Parameter Values on page 866](#)

Reference

- ▶ [Configure Value Area on page 867](#)
- ▶ [Constant Value Options Dialog Box on page 870](#)
- ▶ [Value Configuration Options Dialog Box on page 872](#)
- ▶ [Regular Expression Characters and Usage Options on page 875](#)
- ▶ [Regular Expression Evaluator on page 882](#)
- ▶ [Smart Regular Expression List on page 884](#)

Concepts

Value Configuration Overview

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 for some values to increase the flexibility and adaptability of your tests. For details on regular expressions, see "Regular Expressions Overview" on page 863.

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 manually defined value that remains unchanged for the duration of the test. In certain contexts, you can define a constant value using a regular expression.
- **Parameter.** A value that is defined or generated externally and is retrieved during a run session. 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 22, "Parameterizing Values."



Regular Expressions Overview

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.

Regular expressions are used to identify objects and text strings with varying values. You can use regular expressions to instruct QuickTest to find a value that matches a particular pattern or condition instead of a specific hard-coded value.

Use case examples:

- Suppose 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. When the relevant step runs, QuickTest compares the regular expression that you provide with the corresponding value in your application.
- Suppose the text property of an object is a date value, but the displayed date changes according to the current date. You can define a regular expression for the date, so that QuickTest can identify the object that contains text with the expected date format, rather than the exact date value.

Whenever a QuickTest feature supports regular expressions, the relevant QuickTest dialog box includes a **Regular Expression** check box. Selecting this check box instructs QuickTest to treat the provided value as a regular expression. Some dialog boxes that contain a **Regular Expression** check box, also contain a right arrow adjacent to the text box for the value. Clicking this arrow enables you to select regular expression characters from a drop-down list, and to test your regular expression to make sure it suits your needs. For more information, see "Smart Regular Expression List" on page 884 and "Regular Expression Evaluator" on page 882.

The following are some examples of situations in which you can use regular expressions:

- When defining the property values of an object in dialog boxes or in programmatic descriptions
- When defining expected values for checkpoints
- When defining pop-up window conditions in a recovery scenario

You can use regular expressions only for values of type **string**.

For details on defining regular expressions, including regular expression syntax, see "Regular Expression Characters and Usage Options" on page 875.

This section also includes:

- "Regular Expressions for Property Values" on page 864
- "Regular Expressions in Checkpoints" on page 865

Regular Expressions for Property Values

If you expect the value of an object property in your application to change in a predictable way during each run session, you can use regular expressions when defining identification property values, for example, in the Object Repository window, or in programmatic descriptions. For more information on programmatic descriptions, see "Programmatic Descriptions" on page 946.

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 this variation in the button name when identifying the button in your application.



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 for 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 "Configure Value Area" on page 867.

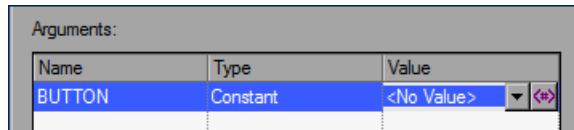
For example, for table checkpoints you can set cell values as regular expressions, and for XML checkpoints you can set attribute or element values as regular expressions. For more information on specific checkpoint types, see the relevant chapter for the checkpoint type.

Tasks

How to Configure Constant and Parameter Values

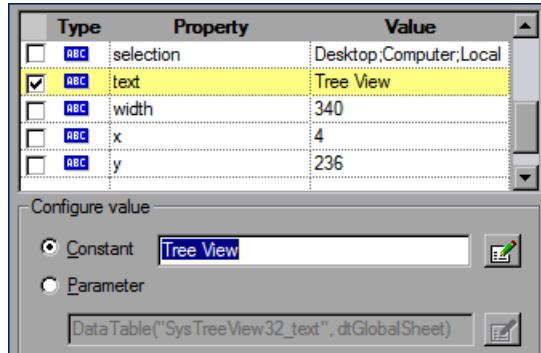
This task describes how to define a value as a constant or a parameter:

- In the Value Configuration Options dialog box (available from the Keyword View, Step Generator, or Object Repository window, and so on), click the parameterization button  for a selected value and define the value. For example:



For details, see "Value Configuration Options Dialog Box" on page 872.

- Above the **Configure value** area of a dialog box, select a property or argument, for example:



Then, in the **Configure value** area, select the **Constant** or **Parameter** radio button and click the **Constant Value Options** or **Parameter Options** button . For details, see "Configure Value Area" on page 867 or "Parameter Options Dialog Box (Data Table)" on page 763, respectively.

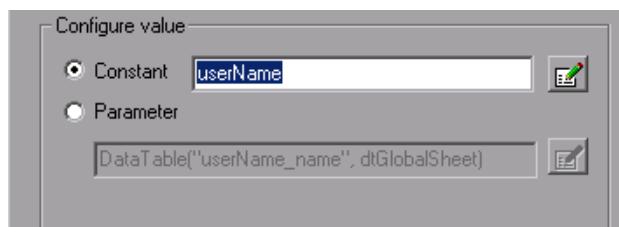
Reference

Configure Value Area

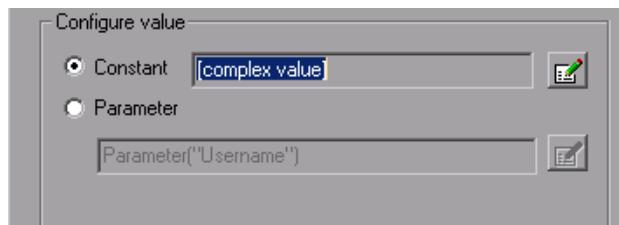
This area enables you to configure object property values or the values of the operation arguments defined for the step.

The following examples illustrate various value definitions:

Single-line constant



Complex constant



Parameter



To access	This area is available in many dialog boxes, for example, the various Checkpoint Properties and Output Value dialog boxes, the various Parameter Options dialog boxes, the Data Driver Wizard, and so on.
------------------	---

User interface elements are described below:

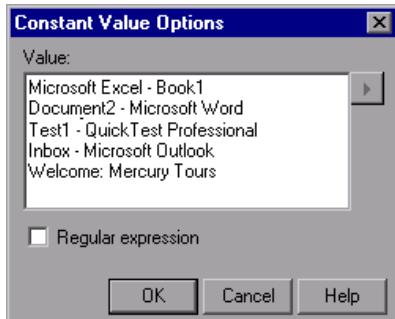
UI Elements	Description
Constant	<p>A manually defined value that remains unchanged for the duration of the run. In certain contexts, you can define a constant value using a regular expression.</p> <p>This is the default option.</p> <p>If the entire value cannot be displayed in the Constant box, it is 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.</p> <p>You can edit a single-line value directly in the Constant box or by clicking the Constant Value Options button . This also enables you to define a string value or complex value as a regular expression. For details on regular expressions, see "Regular Expressions Overview" on page 863.</p>

UI Elements	Description
	<p>Parameter A value that is defined or generated externally and is retrieved during a run session. For example, a parameter value may be defined in an external file or generated by QuickTest.</p> <p>The Parameter box displays:</p> <ul style="list-style-type: none"> ➤ The current parameter definition for a value that is already parameterized ➤ The default parameter definition for a value that is not yet parameterized. <p>For more information on default parameter definitions, see "Default Parameter Values" on page 757.</p> <p>You can click the Parameter Options button  to open the Parameter Options dialog box that enables you to select a different parameter type or modify the parameter settings for the value. For details, see:</p> <p>For tests:</p> <ul style="list-style-type: none"> ➤ "Parameter Options Dialog Box (Test/Action Parameter)" on page 760 ➤ "Parameter Options Dialog Box (Data Table)" on page 763 ➤ "Parameter Options Dialog Box (Environment)" on page 766 ➤ "Parameter Options Dialog Box (Random Number)" on page 770 <p>For more details on using parameters in your tests, see Chapter 22, "Parameterizing Values."</p>

Constant Value Options Dialog Box

This dialog box enables you to edit the value of a constant.

For a **complex value** (a value that cannot be displayed entirely in the **Constant** box in the **Configure value** area of a dialog box), the Constant Value Options dialog box expands to show the entire contents of the value.



To access	Click the Constant Value Options button  in the Configure value area of a dialog box.
See also	"Value Configuration Options Dialog Box" on page 872

User interface elements are described below:

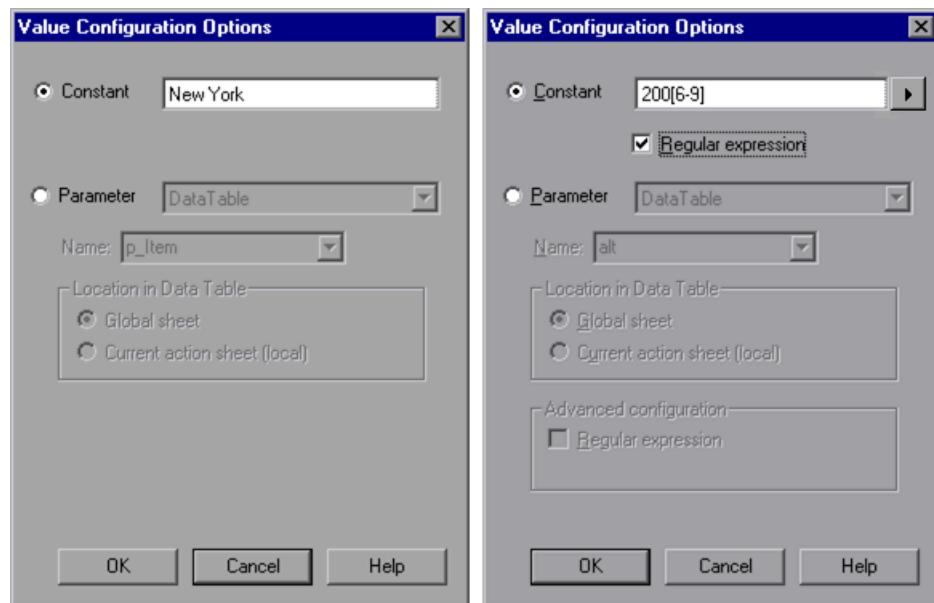
UI Elements	Description
Value	The value for the constant.
Regular expression	<p>Sets the defined value as a regular expression. If you select this option, an arrow is enabled to the right of the Value text box, enabling you to insert regular expression characters and to test your regular expression to make sure that it suits your needs.</p> <ul style="list-style-type: none">▶ For general information on regular expressions, see "Regular Expressions Overview" on page 863.▶ For information on defining a regular expression, see "Regular Expression Characters and Usage Options" on page 875.▶ For information on inserting regular expression characters directly from a list, see "Smart Regular Expression List" on page 884.▶ For information on testing your regular expression, see "Regular Expression Evaluator" on page 882.

Value Configuration Options Dialog Box

This dialog box enables you to define a selected value as a constant or a parameter. In some situations, you can also define the constant or parameter using a regular expression.

View Image

The following examples illustrate the Value Configuration Options dialog box with and without the **Regular expression** check box. The parameter options shown in this dialog box change according to the parameter type selected in the **Parameter** box.



To access

In the Keyword View, click the parameterization button  for a selected value.

User interface elements are described below:

UI Elements	Description
Constant	<p>A manually defined value that remains unchanged for the duration of the run.</p> <p>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 details, see "How to Define Values for Your Step Arguments" on page 491.</p>
Regular expression	<p>Enables you to specify a constant value using a regular expression.</p> <p>This option is available only for string value types in some situations, for example, when parameterizing an object identification property value. This option is not available for method arguments.</p> <p>To use a regular expression: Enter a regular expression in the Constant box and select the Regular expression check box.</p> <p>For more details, see "Regular Expressions Overview" on page 863, "Smart Regular Expression List" on page 884, and "Regular Expression Evaluator" on page 882.</p>

UI Elements	Description
	<p>Parameter A value that is defined or generated externally and is retrieved during a run session. The parameter section displays:</p> <ul style="list-style-type: none"> ➤ The current parameter type and details for the value if a value is already parameterized. ➤ The default parameter type and details for the value if a value is not yet parameterized. You can change the default definition by selecting a different parameter type or modifying the parameter settings for the value. For details, see "Default Parameter Values" on page 757. <p>Options include:</p> <p>The options in the Parameter section change according to the parameter type you select and are very similar to the options in the Parameter Options dialog box.</p> <ul style="list-style-type: none"> ➤ Test/action parameter. See "Parameter Options Dialog Box (Test/Action Parameter)" on page 760. ➤ Data Table parameter. See "Parameter Options Dialog Box (Data Table)" on page 763. ➤ Environment parameter. See "Parameter Options Dialog Box (Environment)" on page 766. <p>Note: Only value types that match the value type you are parameterizing are shown in the Name list.</p> <ul style="list-style-type: none"> ➤ Random Number parameter. See "Parameter Options Dialog Box (Random Number)" on page 770. <p>For more details on using parameters in your tests, see "Parameterizing Values" on page 723.</p>

Regular Expression Characters and Usage Options

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 any dialog box that contains the **Regular Expression** check box, and then you select the **Regular Expression** check 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.

All programmatic description property values are automatically treated as regular expressions. For more information on programmatic descriptions, see "Programmatic Descriptions" on page 946.

This section describes some of the more common options that can be used to create 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.

If the backslash character is not used for either of these purposes, it is ignored.

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 (
- one\two matches the string onetwo

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

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 878. For more information on the | regular expression character, see "Matching One of Several Regular Expressions (|)" on page 879.

Matching Any Single Character in a List ([xy])

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 ([^xy])

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 ([x-y])

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 arguments to an alternation operator (|) or a repetition operator: (*, + , ?, {})

Matching One of Several Regular Expressions (|)

A vertical line (|) instructs QuickTest to match one of a choice of expressions. For example:

foo|bar

causes QuickTest to match either foo or bar

fo(o|b)ar

causes QuickTest to match either foobar or foob

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)

\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)

\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

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.

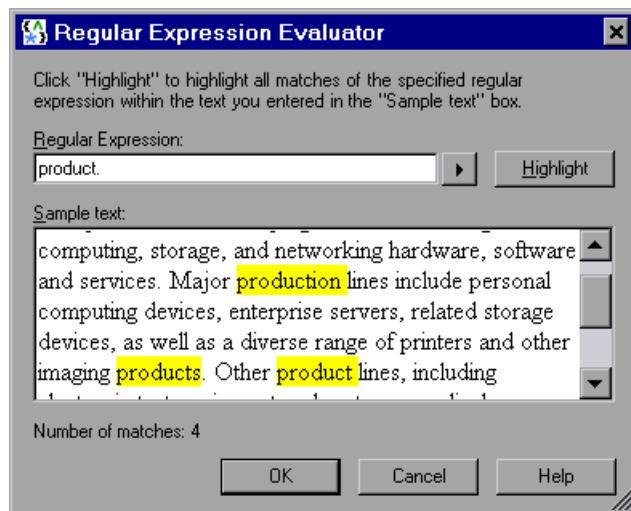
([0-9]?[0-9]?[0-9]|1[01][0-9][0-9]|1200)

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 > VBScript User's Guide > Introduction to Regular Expressions**).

Regular Expression Evaluator

This dialog box enables you to create and test a regular expression to determine whether it suits your needs. You can enter a regular expression and sample text to test. When you click **Highlight**, QuickTest searches the sample text for matches, highlights these matches, and displays the number of matches.

The following example searches for the word product followed by a space or other character because the regular expression includes the period (.) character.



To access	Use one of the following: <ul style="list-style-type: none"> ▶ Select the Tools > Regular Expression Evaluator menu command. ▶ Click the Smart Regular Expression button  to the right of a value box in any dialog box in which the Regular Expression check box is selected, and select the Open Regular Expression Evaluator list item.
------------------	--

Important information	The list of selectable regular expression characters available from this dialog box is also available from other QuickTest dialog boxes that contain a selected Regular Expression check box and the Smart Regular Expression button  .
See also	<p>Conceptual overview: "Regular Expression Characters and Usage Options" on page 875</p> <p>Additional related topics: "Smart Regular Expression List" on page 884</p>

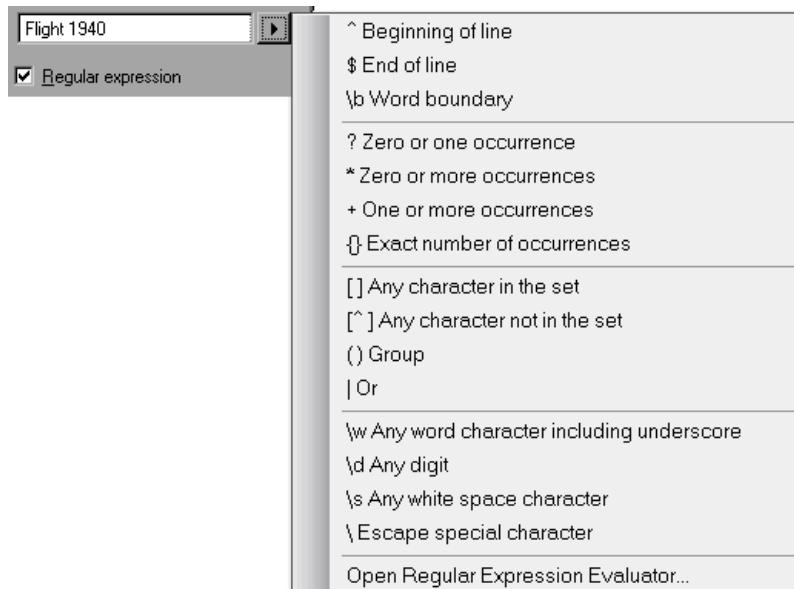
User interface elements are described below:

UI Elements	Description
	<p>Smart Regular Expression. Click to display a list of regular expression characters that you can select. For more information on each of the regular expression characters, see "Smart Regular Expression List" on page 884.</p>
Regular Expression	<p>The regular expression to test. Enter the regular expression in this box.</p> <p>Tip: Click the right arrow  to display a list of regular expression characters that you can insert.</p>
Highlight	<p>Searches for the regular expression in the Sample text area and highlights all matches.</p> <p>Note: Before you click Highlight, make sure that the text to search is displayed in the Sample text area.</p>
Sample text	<p>The text to use when testing the regular expression.</p> <p>Note: Text may already be displayed in the Sample text area when the dialog box opens, depending on the steps you performed to open this dialog box.</p> <p>The Sample text area is editable. You can insert any text to test your regular expression.</p>
Number of matches	The number of matches for the regular expression within the sample text. These matches are highlighted in the Sample text area.

Smart Regular Expression List

This list:

- Displays a list of commonly used regular expression characters.
- Enables you to select a regular expression character from the list and insert it in a value.
- Enables you to access the Regular Expression Evaluator (described on page 882).



Note: If you open the Smart Regular Expression list from the Regular Expression Evaluator, the **Open Regular Expression Evaluator** command is not included in the list.

To access	Click the Smart Regular Expression button  to the right of a value box in any dialog box in which the Regular Expression check box is selected.
See also	<p>Conceptual overview: "Regular Expressions Overview" on page 863 and "Regular Expression Characters and Usage Options" on page 875</p> <p>Additional related topics: "Regular Expression Evaluator" on page 882</p>

User interface elements are described below (unlabeled elements are shown in angle brackets):

UI Elements	Description
<Regular expression options>	<p>The list of regular expression characters that you can select when creating a regular expression.</p> <p>When you select a regular expression character from the list, QuickTest inserts it in the currently open dialog box and closes the list. Depending on its type, a regular expression character can be inserted:</p> <ul style="list-style-type: none"> ➤ At the cursor location ➤ Around selected text ➤ In place of selected text <p>Example 1: If you select text and then select the [] Any character in the set option, the brackets are inserted on either side of the selected text, as follows: [text]</p> <p>Example 2: If you select text and then select the \d Any digit option, the selected content is replaced with \d. For example, if you select 1 in Document1 and then select the \d Any digit option, the selected content is replaced with \d, as follows: Document1 becomes Document\d</p> <p>Note: The \n New line option is available only if applicable.</p> <p>For more information on working with regular expression characters, see "Regular Expression Characters and Usage Options" on page 875.</p>

UI Elements	Description
Open Regular Expression Evaluator	<p>Opens the Regular Expression Evaluator dialog box, enabling you to test your regular expression to make sure that it suits your needs. For more information, see "Regular Expression Evaluator" on page 882.</p> <p>Note: This command is not available from the Smart Regular Expression List dialog box.</p>

26

User Interface-Based Programming Operations

This chapter includes:

Concepts

- [Programming Statements Overview on page 888](#)
- [Test Synchronization on page 894](#)
- [Message Statements on page 897](#)

Tasks

- [How to Insert Steps Using the Step Generator on page 900](#)
- [How to Insert Conditional Statements from the Keyword View on page 901](#)
- [How to Insert Loop Statements In the Keyword View on page 904](#)
- [How to Generate With Statements for Your Test on page 906](#)

Reference

- [Add Synchronization Point Dialog Box on page 910](#)
- [Comment Properties Dialog Box on page 912](#)
- [Insert Comment Dialog Box on page 912](#)
- [Insert Report Dialog Box on page 914](#)
- [Step Generator Dialog Box on page 916](#)
- [Storage Location Options Dialog Box on page 924](#)

Concepts

Programming Statements Overview

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 (programming statements) 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.** These affect the way the test runs, such as Conditional Statements and Synchronization Points. These are often used to control the logical flow of a test.
- **Comments.** Use comments to explain sections of your tests to improve readability and to make them easier to update. A comment is an explanatory remark in a program, and does not get processed when QuickTest runs. For details, see "Insert Comment Dialog Box" on page 912.

This section also includes:

- "Step Generator" on page 889
- "Conditional Statements" on page 890
- "With Statements" on page 892

 **Step Generator**

The Step Generator enables you to add steps by selecting from a range of context-sensitive options and entering the required values, 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.

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.

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 "Keyword View User Interface" on page 504.

For more information, see "Step Generator Dialog Box" on page 916.



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.

The statement has the following syntax:

If *condition* **Then** *statements* [**Else** *elsestatements*] **End If**

Or, you can use the block form syntax:

```
If condition Then  
    [statements]  
[ElseIf condition-n Then  
    [elseifstatements] . . .  
[Else  
    [elsestatements]  
End If
```

For information inserting conditional statements using the Step Generator, see "Step Generator Dialog Box" on page 916.

For information on working with conditional steps in the Expert View, see "Comments, Control-Flow, and Other VBScript Statements" on page 960, and the VBScript documentation (select **Help > QuickTest Professional Help > VBScript Reference**).

Example:

The following example checks whether a valid order number is entered in the Order No. 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 Run 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.

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 run 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
```

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
```

Example:

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")
```

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.

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.

For more information, see "How to Generate With Statements for Your Test" on page 906.

Test Synchronization

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.

This section also includes:

- "Synchronization Points" on page 895
- "Exist and Wait Statements" on page 896
- "Timeout Values Modification" on page 897

 **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. For more information, see "Add Synchronization Point Dialog Box" on page 910.

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 "Exist and Wait Statements" on page 896.

 **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*.



Timeout Values Modification

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 "Run Pane (Test Settings Dialog Box)" on page 1471.
- 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*.



Message Statements

You can generate messages in your test that are displayed in the Run Results Viewer. You can also choose to display messages on screen while running your test.

This section includes:

- "Sending Messages to the Run Results" on page 898
- "Displaying Messages During the Run Session" on page 899

Sending Messages to the Run Results

You can send the following types of messages to the run results:

- **Test-specific messages.** You can add notes run results to provide additional information about the test run. For example, you can specify the application version tested, the operating system used, languages, and so on.

In the Run Results Viewer, this information is displayed in the **Executive Summary** page. For details, see "Run Results Viewer User Interface" on page 1107.

You add a note by inserting a **Reporter.ReportNote** step in your test.

Example:

`Reporter.ReportNote "This test was run from 12.34.56.89 using a wireless connection."`

For details, see the **Utility** section of the *HP QuickTest Professional Object Model Reference*.

- **Step-specific messages.** You can define a message that QuickTest sends to your run 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 run results indicating that the object is absent.

You send messages to the run results using the Insert Report dialog box. For details, see "Insert Report Dialog Box" on page 914.

When you add a report step, a **Reporter.ReportEvent** step is inserted.

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.

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 Run 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.

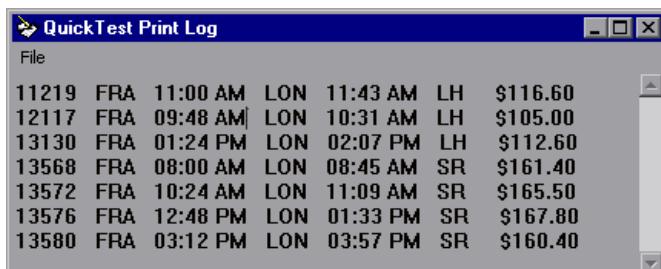
```
Set FlightsList = Window("Flight Reservation").Dialog("Flights Table").
WinList("From")
```

```
For i = 1 to FlightsList.GetItemCount
```

```
Print FlightsList.GetItem(i - 1)
```

```
Next
```

The QuickTest Print Log window remains open throughout the run session, until you close it.



Tasks

How to Insert Steps Using the Step Generator

This task describes how to insert steps using the Step Generator.

1 Prerequisites

Select where in your test the new step should be inserted.

2 Open the Step Generator dialog box

Open the Step Generator from one of the following locations:

- ▶ Keyword View
- ▶ Expert View
- ▶ Active Screen

For more information, see "Step Generator Dialog Box" on page 916.

3 Define the category, type and operations for the step

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.

4 View the step documentation or syntax

In the Step Generator, view the step documentation or statement syntax and add your new step or statement to your test or function library.

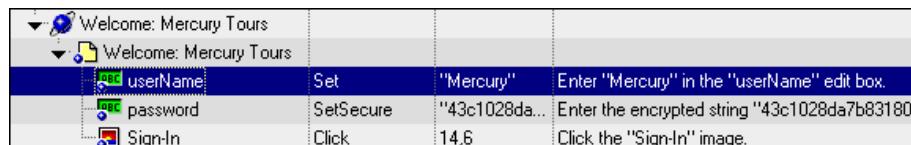
5 Results

The Step Generator 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.

How to Insert Conditional Statements from the Keyword View

This task describes how to insert a conditional **If...Then...Else** statement from 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:



A screenshot of the Keyword View interface. A table displays a list of steps under the category 'Welcome: Mercury Tours'. The 'userName' step is highlighted with a blue selection bar. The table has four columns: Step Name, Operation, Value, and Description. The 'userName' row contains the following data:

Step	Operation	Value	Description
userName	Set	"Mercury"	Enter "Mercury" in the "userName" edit box.
password	SetSecure	"43c1028da7b83180"	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			
userManager	Set	"Mercury"	Enter "Mercury" in the "userName" edit box.
IF [] Statement	True		Check whether (True) is true. If so:
userManager	SetSecure	"43c1028da7b83180"	Enter the encrypted string "43c1028da7b83180"
Sign-In	Click	14,6	Click the "Sign-In" image.

- 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			
userManager	Set	"Mercury"	Enter "Mercury" in the "userName" edit box.
IF [] userManager	Set		<No documentation summary is available for the object.
userManager	SetSecure	"43c1028da7b83180"	Enter the encrypted string "43c1028da7b83180"
Sign-In	Click	14,6	Click the "Sign-In" image.

- 4 Click in the **Operation** cell and select the operation you want to perform. For example:

▼ Welcome: Mercury Tours			
▼ Welcome: Mercury Tours			
userManager	Set	"Mercury"	Enter "Mercury" in the "userName" edit box.
IF [] userManager	Exist		Check whether the "userName" edit box exists.
userManager	SetSecure	"43c1028da7b83180"	Enter the encrypted string "43c1028da7b83180"
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, the **Exist** property does not require a value in 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			
userManager	Set	"Mercury"	Enter "Mercury" in the "userName" edit box.
IF [] userManager	Exist		Check whether the "userName" edit box exists.
userManager	Set	DataTable("p_Us...")	Enter <the value of the 'p_UserNames' Data
userManager	SetSecure	"43c1028da7b83180"	Enter the encrypted string "43c1028da7b83180"
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 userN	Exist		Check whether the "userName" edit box exists. If so...
userN	Set	DataTable("p_Us...")	Enter <the value of the 'p_UserName' Data Table
passw	SetSecure	"43c1028da7b831801c87..."	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:

	Statement	Option
	If...Then	Insert > Conditional Statement > If...Then
	Elself...Then	Insert > Conditional Statement > Elself...Then
	Else	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 Run Results.

>Welcome: Mercury Tours			
>Welcome: Mercury Tours			
If userN	Exist		Check whether the "userName" edit box exists. If so...
userN	Set	DataTable("p_Us...")	Enter <the value of the 'p_UserName' Data Table
Else			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 "Standard Steps After a Conditional or Loop Block" on page 487.

How to Insert Loop Statements In the Keyword View

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:

Icon	Statement	Description
	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.
- 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.
- 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 "Standard Steps After a Conditional or Loop Block" on page 487.

Note: For more information on working in the Expert View, see Chapter 27, "Working in the Expert View and Function Library Windows."

Example:

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 co
▼ Statement		For i = 1 To ItemsCount - 1	Repeat the following step(s) one or more tim
toDay	GetItem	i	Retrieve the value of the item with index
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
```

How to Generate With Statements for Your Test

This task describes the steps to generate and manage **With** statements in your test.

This task includes the following steps:

- "Instruct QuickTest to generate With statements while recording" on page 906
- "Generate With statements for existing actions in the Expert View" on page 907
- "Remove With statements from an action in the Expert View" on page 909

Instruct QuickTest to generate With statements while recording

- 1 In the Test Settings dialog box (described on page 1422), select **General** pane > **Automatically generate "With" statements after recording**.
- 2 In the **Generate "With" statements for __ or more objects** box, enter the minimum number of consecutive, identical objects for which you want to apply the **With** statement. The default is 2.

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.

- 3 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.

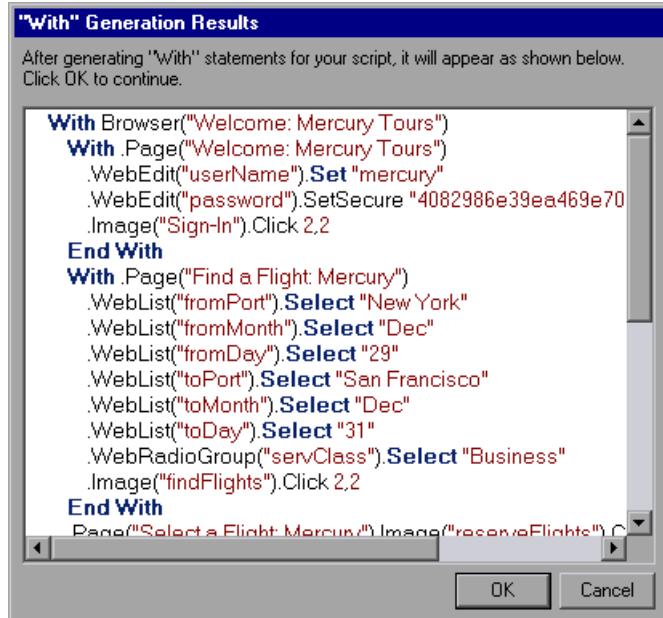
Generate With statements for existing actions in the Expert View

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 In the Test Settings dialog box (described on page 1422), select **General** pane > **Automatically generate "With" statements after recording**.
- 2 Confirm that the proper number is set for the **Generate "With" statements for __ or more objects**. The default is 2.
- 3 Display the action for which you want to generate **With** statements.

- 4** From the Expert View, select **Edit > Advanced > Apply "With" to Script**.
 The "With" Generation Results window opens.



Each **With** statement contains only one object.

- 5** To confirm the generated results, click **OK**. The **With** statements are applied to the action.

Tips:

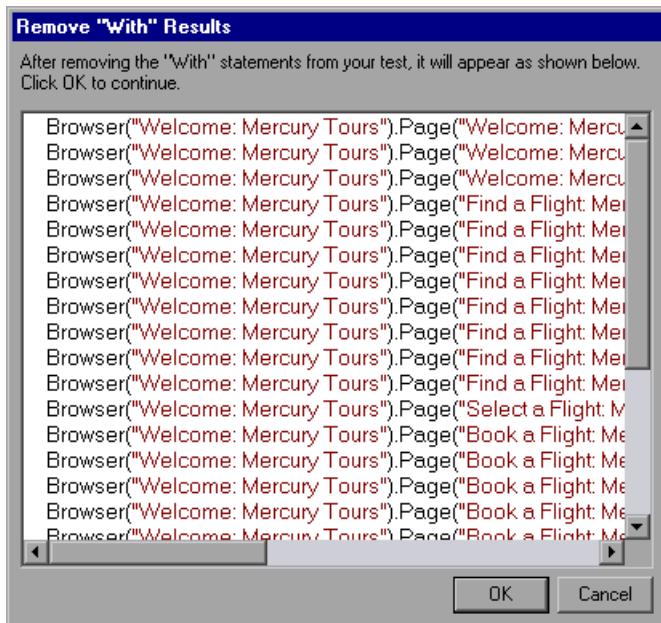
- You can search for text strings in the Generation Results window by pressing **CTRL+F**. For more information on the Find dialog box, see "Find Dialog Box" on page 991.
- If you type a **With** statement (as opposed to creating it using the procedure described above), select **Edit > Advanced > Apply "With" to Script** to enable IntelliSense within the **With** statement.

Remove With statements from an action in the Expert View

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.

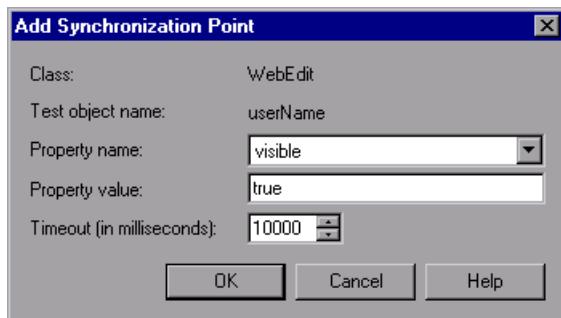


- 3 To confirm the results, click **OK**. The **With** statements are replaced with the standard statement format.

Reference

Add Synchronization Point Dialog Box

This dialog box enables you to insert a **WaitProperty** statement to synchronize your test.



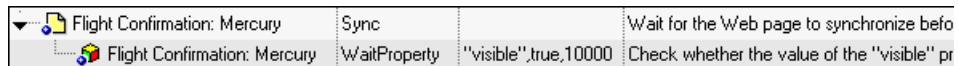
To access	<p>1 Start a recording session. 2 Display the screen or page in your application that contains the object for which you want to insert a synchronization point. 3 In the QuickTest window, select Insert > Synchronization Point.</p>
Important information	<ul style="list-style-type: none"> ➤ When you insert a synchronization point, the pointer changes to a pointing hand. For more information on using the pointing hand, see "Tips for Using the Pointing Hand" on page 157. ➤ For more information on the WaitProperty method, see the <i>HP QuickTest Professional Object Model Reference</i>.
See also	<p>"Test Synchronization" on page 894</p>

User interface elements are described below:

UI Elements	Description
Class	The test object class of the selected object.
Test object name	The name of the selected object.
Property name	The list of the identification properties associated with the object class. Select the property you want to use for the synchronization point.
Property value	Enables you to specify the property value for which QuickTest should wait before continuing to the next step in the test. The property values that the object has at the time that you insert the synchronization point do not impact the specified synchronization point.
Timeout (in milliseconds)	The time (in milliseconds) that you want QuickTest to wait before continuing to the next step, if the specified property value was not achieved.

View Example

After you insert a synchronization point for the **Update Order** button, it may look something like this:

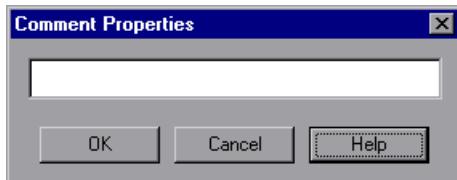


In the Expert View, this appears as:

```
Browser("Welcome: Mercury Tours").Page("Flight Confirmation: Mercury").Sync
Browser("Welcome: Mercury Tours").Page("Flight Confirmation: Mercury").
    WebElement("Flight Confirmation #").WaitProperty "visible",
    true, 10000
```

Comment Properties Dialog Box

This dialog box enables you to modify the text of a selected comment.



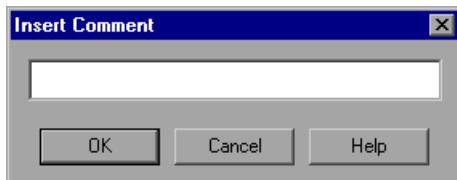
To access	In the Keyword View, right-click any step containing a comment and select Comment Properties .
Important information	<ul style="list-style-type: none"> ► If the Comment column is not visible, right-click any column header and select Comment. ► In the Keyword View, you also can modify the comment text directly in the Comment column. ► In the Expert View, you can manually edit the text to overwrite any existing comment.

User interface elements are described below (unlabeled elements are shown in angle brackets):

UI Elements	Description
<comment text area>	The comment text to modify.

Insert Comment Dialog Box

This dialog box enables you to insert a new comment for a step.



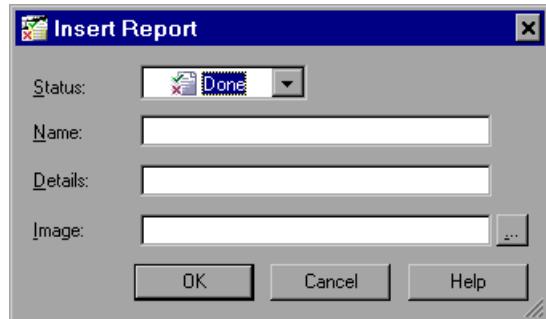
To access	In the Keyword View, right-click any step that does not yet contain a comment and select Insert Step > Comment .
Important information	<ul style="list-style-type: none"> ► If the Comment column is not visible, right-click any column header and select Comment. ► You can add comments directly in the Keyword View by selecting a step and typing the comment in the Comment column. ► You can modify comments at any time directly in the Keyword View, or by using the Comment Properties dialog box. ► 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. ► 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 "How to Use Actions in Your Test" on page 542.

User interface elements are described below (unlabeled elements are shown in angle brackets):

UI Elements	Description
<comment text area>	The comment text to enter.

Insert Report Dialog Box

This dialog box enables you to define a message that QuickTest sends to your run results.



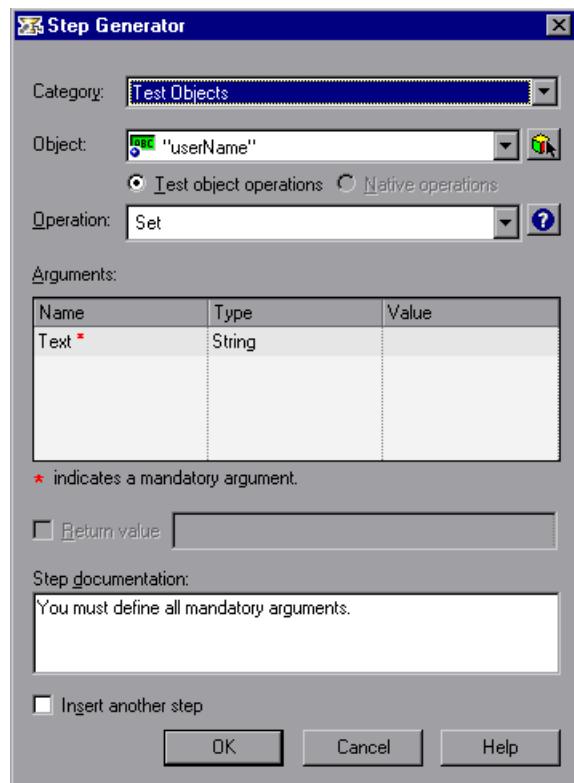
To access	In the Keyword View: ► Select a step and select Insert > Report . ► Right-click a step and select Insert Step > Report .
Important Information	When you add a report step, a Reporter.ReportEvent step is inserted.
See also	"Message Statements" on page 897

User interface elements are described below:

UI Elements	Description
Status	<p>The status that will result from this step from the Status list:</p> <ul style="list-style-type: none"> ➤ 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.
Name	<p>The name for the step, for example, Password edit box. This name is displayed in the Run Results tree as the node label for this step.</p>
Details	<p>The description of this step. For example, Password edit box does not exist. The description is displayed in the upper-right pane of the Run Results Viewer.</p>
Image	<p>The name of an image to include in the run results with this step. The image is displayed in the bottom-right pane of the Run Results Viewer. The image can be any of the supported file types, such as BMP, JPEG, GIF, TIF, or PNG.</p> <p>Notes:</p> <ul style="list-style-type: none"> ➤ You cannot specify an image stored in Quality Center. ➤ Including large images in the run results may impact performance. ➤ If an image is specified as a relative path, QuickTest will first search the Results folder for the image and then the search paths specified in the Folders pane of the Options dialog box.

Step Generator Dialog Box

This dialog box enables you to add steps that perform operations, using test object methods (for tests only), Utility object methods, or function calls.



To access	<p>This dialog box can be accessed from the following locations:</p> <ul style="list-style-type: none"> ➤ Keyword View ➤ Expert View ➤ Function Library ➤ Active Screen <p>To open the dialog box, select the location to insert the step, and do one of the following:</p> <ul style="list-style-type: none"> ➤ Right-click and select Insert Step > Step Generator. ➤ Press F7 (not available for Active Screen).
Important information	<ul style="list-style-type: none"> ➤ Although the Step Generator shows information regarding the currently selected step or function, selections that you make in the Step Generator add a new step or function to your test; they do not modify the existing step or function. ➤ In the Step Generator, if you add an operation that returns an object, and the assignment in the test is missing a Set statement, the run session will fail.
Relevant tasks	"How to Insert Steps Using the Step Generator" on page 900

General User Interface Elements

UI Elements	Description
Category	<p>The type of step to add. The following options are available:</p> <ul style="list-style-type: none"> ➤ Test Objects. Enables you to select a test object and operation for the step (for tests only). For more information, see "Test Object Category" on page 920. ➤ Utility Objects. Enables you to select a Utility object and operation for the step. For more information, see "Utility Object Category" on page 921. ➤ 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 "Function Category" on page 922.
Object	<p>The list of available objects. The list varies according to the type of object you select in the Category list box.</p>

UI Elements	Description
Operation	The list of available operations for the selected object type in alphabetical order.
Arguments	The list of arguments for the operation, if applicable to the selected operation.
Arguments > Name	The name of the argument for the selected operation.
Arguments > Type	The type of the argument for the selected operation.
Arguments > Value	<p>The value for the argument for the selected operation. The following types of the values are available:</p> <ul style="list-style-type: none"> ➤ 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 "Value Configuration Options Dialog Box" on page 872. ➤ 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.

UI Elements	Description
Return Value	The location for storing the return values of the operation, if applicable. For information on storing options, see "Storage Location Options Dialog Box" on page 924.
Step documentation (Keyword View)	<p>Summary information on the current step. The following options are available:</p> <ul style="list-style-type: none"> ▶ For the Test Object or Utility Object categories, 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. Note: If any mandatory and required argument values have not been defined for the operation, the Step documentation box displays a warning message. ▶ For Functions category, step documentation is available for user-defined functions, if you provided this information when defining them. For more information, see "How to Work with a User-Defined Function" on page 1039.
Generated Step (Expert View / Function Library)	<p>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.</p> <p>Example:</p> <div data-bbox="630 1107 1275 1254" style="border: 1px solid black; padding: 5px;"> <p>Generated step:</p> <pre>Browser("Welcome: Mercury Tours").Page("Welcome: Mercury Tours").WebEdit("userName").Set Text</pre> </div>
Insert another step	Enables you to insert the current step and continue adding steps at the same location. The OK button changes to Insert .

Test Object Category



Important information

- 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.
- After you select the operation for your test object, you can define the relevant argument values.
- 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.

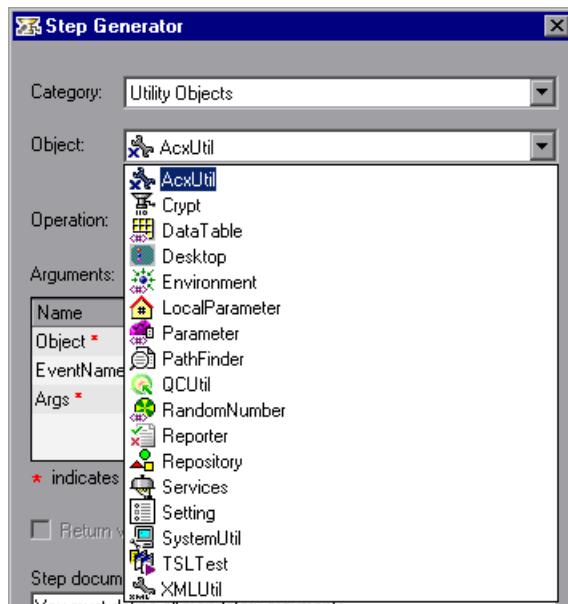
User interface elements are described below:

UI Elements	Description
Object	All the objects in the object repository that are at the same hierarchical level and location as the currently selected step. Note: The objects are listed by name in alphabetical order.
	Select Object. Enables you to select an object from the object repository or from your application. For more information, see "Select Test Object Dialog Box" on page 513.

UI Elements	Description
Test object operations	The QuickTest operations that can be performed on a test object.
Native operations	<p>The operations of the object in your application as defined by the object creator.</p> <p>Notes:</p> <ul style="list-style-type: none"> ► If QuickTest cannot retrieve native operations for the selected object, the Native operations option is not available. ► If you select a native operation, the Step Generator inserts a step using .Object syntax. For information on using the Object property, see "Native Properties and Operations" on page 962.

Utility Object Category

This option enables you to specify a utility object to insert to your test.

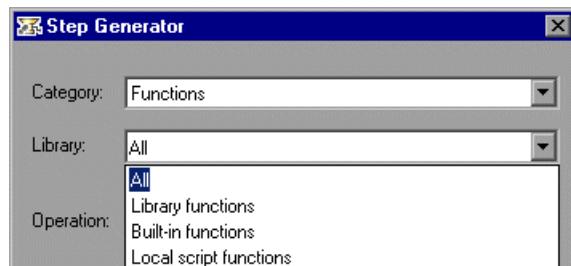


Important information	For more information on Utility objects, see the Utility Objects section of the <i>HP QuickTest Professional Object Model Reference</i> .
------------------------------	---

User interface elements are described below:

UI Elements	Description
Object	<p>The list of Utility objects that are available when you open the Step Generator from the Keyword View.</p> <p>Note: 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 loaded, the list may include additional Utility objects for those add-ins.</p>

Function Category



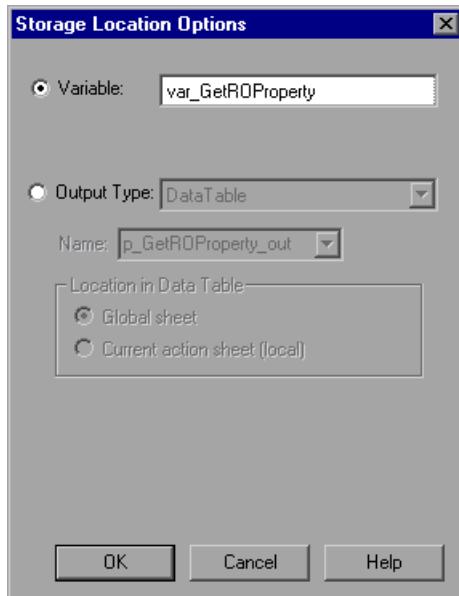
Important information	For detailed information on a selected built-in VBScript function, see Microsoft's VBScript Reference or the <i>HP QuickTest Professional Object Model Reference</i> .
------------------------------	--

User interface elements are described below:

UI Elements	Description
Library	<p>The list of function types. The following options are available:</p> <ul style="list-style-type: none">▶ 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 "Associated Function Libraries" on page 1015.▶ 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.

Storage Location Options Dialog Box

This 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. This dialog box also enables you to specify how and where to store the value for an output parameter for an action.



To access	<ul style="list-style-type: none"> ➤ In the Step Generator Dialog Box, click the displayed return value and then the output storage button . ➤ In the Action Call Properties dialog box, select an output parameter in the Parameter Values tab and click the output storage button 
See also	"Default Output Definitions" on page 786

User interface elements are described below:

UI Elements	Description
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 output parameter of the specified type.

Default Output Definitions for Action Parameter Values

When you select **Output Type** or 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 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.

Part V

Defining Functions and Other Programming Tasks

27

Working in the Expert View and Function Library Windows

This chapter includes:

Concepts

- The Expert View and Function Library Windows Overview on page 931
- Expert View and Keyword View - A Comparison on page 932
- Generating Statements in the Expert View or in a Function Library on page 934
- Bookmarks in an Action or Function Library on page 945
- Programmatic Descriptions on page 946
- Opening and Closing Applications Programmatically on page 959
- Comments, Control-Flow, and Other VBScript Statements on page 960
- Retrieving and Setting Identification Property Values on page 961
- Native Properties and Operations on page 962
- Running DOS Commands on page 964
- Choosing Which Steps to Report During the Run Session on page 964
- Windows API on page 964

Tasks

- How to Navigate in the Expert View and Function Libraries on page 965
- How to Enhance Your Tests and Function Libraries Using the Windows API on page 966

Reference

- Checkpoint and Output Statements on page 969
- Basic VBScript Syntax on page 970
- Report Modes on page 987
- Expert View and Function Library Window User Interface on page 988

Concepts

The Expert View and Function Library Windows Overview

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

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

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

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

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

You can add steps to your test or function library either manually or using the Step Generator. For more information on using the Step Generator, see "How to Insert Steps Using the Step Generator" on page 900.

You can print the test displayed in the Expert View or a function library at any time. You can also include additional information in the printout. For more information on printing a function library or printing from the Expert View, see "Print Dialog Box" on page 428.

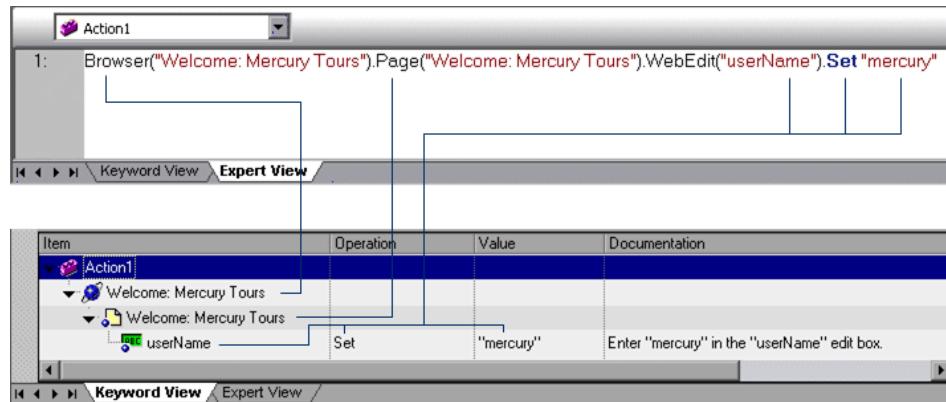
Expert View and Keyword View - A Comparison

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

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

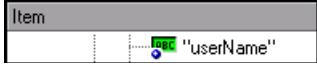
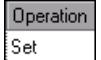
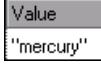
- In the Keyword View, QuickTest displays information about each step and shows the object hierarchy in an icon-based table. For more information, see Chapter 13, "Keyword View."
- In the Expert View, QuickTest displays each step as a VBScript line or statement. In object-based steps, the VBScript statement defines the object hierarchy.

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



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

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

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

For more details, see:

- "Checkpoint and Output Statements" on page 969
- "Parameter Indications in VBScript" on page 975
- "Generating Statements in the Expert View or in a Function Library" on page 934
- "Basic VBScript Syntax" on page 970
- "Automatic Completion of VBScript Syntax" on page 944

Generating Statements in the Expert View or in a Function Library

You can generate statements in the following ways:

- You can use the Step Generator to add steps that use methods and functions. For more information, see "How to Insert Steps Using the Step Generator" on page 900.
- You can manually insert VBScript statements that perform operations. QuickTest includes features that help you adhere to the correct syntax and select the relevant items for your statements.
- **Statement completion (IntelliSense).** This option, when enabled, helps you select the variable, test object, operation, property, or collection for your statement and view the relevant syntax as you type in the Expert View or a function library. For more information, see "Statement Completion (IntelliSense)" on page 934.
- **Auto-expand VBScript syntax.** When this option is enabled, QuickTest automatically adds the relevant syntax or blocks to your script, when you start to type a VBScript keyword in the Expert View or in a function library. For more information, see "Automatic Completion of VBScript Syntax" on page 944.

Statement Completion (IntelliSense)

When you type in the Expert View or a function library, IntelliSense (the statement completion feature included with QuickTest) enables you to select items for your statement from a drop-down list and view the relevant syntax. IntelliSense provides drop-down lists for the following types of items:

- Variables
- Test objects
- Operations
- Properties
- Collections
- Possible values for arguments (in some cases)

The **Statement Completion** option is enabled by default. You can disable or enable this option in the Editor Options dialog box. For more information, see Chapter 28, "Customizing the Expert View and Function Library Windows."

Tips:

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

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

- "Available Test Objects (Expert View Only)" on page 936
- "Available Operations" on page 936
- "Operation Syntax" on page 939
- "Possible Argument Values" on page 939
- "Available Constants and Local Variables" on page 940

For instructions on generating a statement using statement completion in the Expert View or a function library, see page 940.

Available Test Objects (Expert View Only)

If you type a test object class followed by an open parenthesis (, QuickTest displays a list of all test objects of this class in the object repository. If there is only one object of this class in the object repository, QuickTest automatically enters its name in quotes after the open parenthesis. For example, if you type Page(, QuickTest displays a list of all the **Page** test objects in the object repository.

Available Operations

If you type a period after a test object (or after the **Object** property) in a statement, QuickTest displays a list of the relevant test objects, operations, properties, collections, and registered functions that you can add after the item you typed.

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

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

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

Available Operations: Guidelines and Considerations

- If you type a period after an object that you created in your script (using the **CreateObject** method, for example), QuickTest displays the operations and properties that you can use for that object.
 - If you use the **Object** property in your statement and the object data is currently available in the Active Screen or the open application, QuickTest displays the native operations and properties of the object.
-

Note: When you record in a browser other than Internet Explorer, QuickTest captures the dynamic HTML from the relevant browser. However, the Active Screen uses an Internet Explorer engine and thus renders Web-based pages as static HTML with the Internet Explorer DOM exposed.

Therefore, if you insert the **Object** property for a Web object when only Active Screen object data is available (the object is not displayed in an open application), then QuickTest IntelliSense displays the available native operations and properties for the Internet Explorer DOM, even if you recorded the object in another browser.

For more information on the **Object** property, see "Native Properties and Operations" on page 962.

- If you type a period within a **With** statement, QuickTest displays a list of the operations and properties available for the relevant object.
-

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

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

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

The following examples illustrate this:

Example 1:

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

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

Example 2:

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

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

Operation Syntax

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

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

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

Possible Argument Values

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

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

Available Constants and Local Variables

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

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

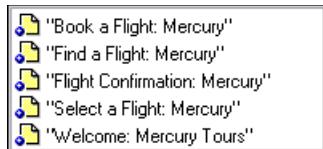
Example

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

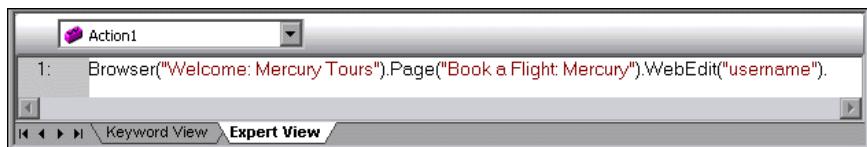
- 1 Confirm that the **Statement completion** option is selected (**Tools > View Options > General** tab).
- 2 Perform one of the following:
 - If you are working in a function library, skip to step 4
 - If you are working in the Expert View, type an object followed by an open parenthesis (



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



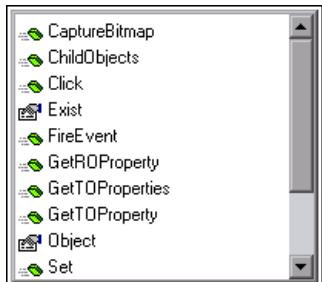
- 3** Double-click an object in the list or use the arrow keys to choose an object and press ENTER. QuickTest inserts the object into the statement.
- 4** Perform one of the following:
 - If you are working in the Expert View, type a period (.) after the object on which you want to perform the operation.



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

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

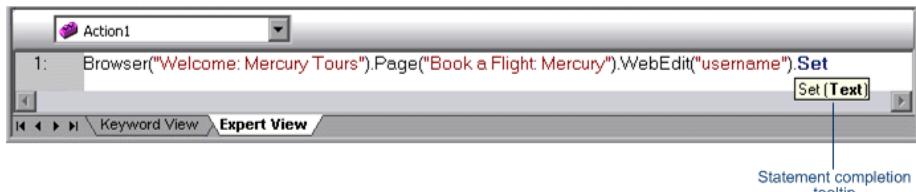
- 5 Type a period (.) after the object description, for example ("username"). QuickTest displays a list of the available operations and properties for the object.



Tips:

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

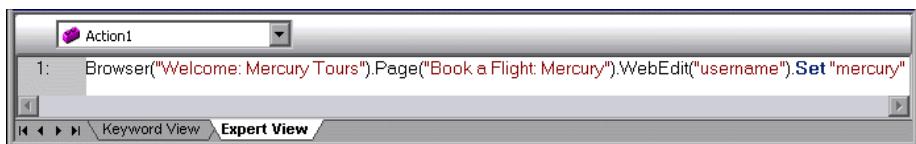
- 6 Double-click an operation in the list or use the arrow keys to choose an operation and press ENTER. QuickTest inserts the operation into the statement. If the operation contains arguments, QuickTest displays the syntax of the operation in a tooltip, as shown in this example from the Expert View.



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

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

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



Note: After you add a step in the Expert View, you can view the new step in the Keyword View. If the statement that you added in the Expert View contains syntax errors, QuickTest displays the errors in the Information pane when you select the Keyword View. For more information, see "Information Pane User Interface" on page 1360.

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

For more information on VBScript syntax, see "Basic VBScript Syntax" on page 970.

Automatic Completion of VBScript Syntax

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

```
If Then  
End If
```

The **Auto-expand VBScript syntax** option is enabled by default. You can disable or enable this option in the Editor Options dialog box. For more information, see "Expert View and Function Library Window Customization Options" on page 1002.

If you enter two characters that are the initial characters of multiple keywords, the Select a Keyword dialog box is displayed and you can select the keyword you want. For example, if you enter the letters `pr` and then enter a space, the Select a Keyword dialog box opens containing the keywords `private` and `property`. You can then select a keyword from the list and click **OK**. QuickTest automatically enters the relevant VBScript syntax or block in the script.

For details on the Select a Keyword dialog box, see "Select a Keyword Dialog Box" on page 998.

For details on VBScript syntax, see "Basic VBScript Syntax" on page 970.



Bookmarks in an Action or Function Library

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

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

```

1: Function check_data_validity( dateStr)
2:     Dim firstSlashPos, secondSlashPos
3:     Dim mmPart, ddPart, yyyyPart
4:     firstSlashPos = inStr( dateStr, "/")
5:     secondSlashPos = inStrRev( dateStr, "/")
6:     If ( firstSlashPos <> 3 or secondSlashPos <> 6) Then
7:         reporter.ReportEvent micFail, "Format check", "The
8:         check_data_validity = False
9:         Exit function
10:    End If
11:    mmPart = mid( dateStr, 1, 2)
12:    ddPart = mid( dateStr, firstSlashPos+1, 2)
13:    yyyyPart = mid( dateStr, secondSlashPos +1, 4)
14:    If mmPart > 12 Then
15:        reporter.ReportEvent micFail, "Format Check", "The
16:        check_data_validity = False

```

For details on using bookmarks, see "How to Navigate in the Expert View and Function Libraries" on page 965.

Programmatic Descriptions

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

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

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

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

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



Use-case scenario:

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

Programmatic Description Types

There are two types of programmatic descriptions:

- **Static.** You list the set of properties and values that describe the object directly in a VBScript statement. For details, see "Static Programmatic Descriptions" on page 948.
- **Dynamic.** You add a collection of properties and values to a Description object, and then enter the Description object name in the statement. For details, see "Dynamic Programmatic Descriptions" on page 951

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

This section also includes:

- "Static Programmatic Descriptions" on page 948
- "Dynamic Programmatic Descriptions" on page 951
- "Retrieving Child Objects" on page 954
- "Programmatic Description Checks" on page 956



Static Programmatic Descriptions

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

The general syntax is:

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

TestObject. The test object class.

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

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

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

Guidelines for Using Static Programmatic Descriptions

When working with static programmatic descriptions, be aware of the following guidelines:

- "Regular Expressions in Programmatic Descriptions" on page 949
- "Variables in Programmatic Descriptions" on page 949
- "Programmatic Descriptions for Parent Test Objects" on page 949
- "Reuse of Static Programmatic Descriptions" on page 950
- "Copying Programmatic Description Data from the Object Spy" on page 951

Regular Expressions in Programmatic Descriptions

QuickTest evaluates all property values in programmatic descriptions as regular expressions. Therefore, if you want to enter a value that contains a special regular expression character (such as *, ?, or +), use the \ (backslash) character to instruct QuickTest to treat the special characters as literal characters. For more information on regular expressions, see "Regular Expressions Overview" on page 863.

Variables in Programmatic Descriptions

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

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

Programmatic Descriptions for Parent Test Objects

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

Example:

- You can use the following statement, which uses object repository names for the parent objects and a programmatic description for the object on which the operation is performed:

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

- You can use the following statement since it uses programmatic descriptions throughout the entire test object hierarchy:

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

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

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

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

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

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

For more information on working with test objects, see Chapter 4, "Managing Test Objects in Object Repositories."

Reuse of Static Programmatic Descriptions

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

Example

Instead of entering:

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

You can enter:

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

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

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

For more information on the **With** statement, see "With Statement" on page 985.

Copying Programmatic Description Data from the Object Spy

You can use the **Copy Identification Properties to Clipboard** option in the Object Spy dialog box to copy all identification properties and values for a selected object to the Windows Clipboard. The copied values are formatted in standard programmatic description syntax with line breaks between each property-value pair. For example:

```
"Class Name:=Image",
"abs_x:=585",
"abs_y:=573",
"alt:=Specials",
....
```

You can paste the copied data to any document and then copy selected lines (remove the line breaks) into a programmatic description. For details on this option, see "Object Spy Dialog Box" on page 151.



Dynamic Programmatic Descriptions

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

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

Set MyDescription = Description.Create()

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

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

For example, instead of entering:

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

you can enter:

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

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

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

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

Considerations for Description Objects

- By default, the value of all **Property** objects added to a **Properties** collection are treated as regular expressions. Therefore, if you want to enter a value that contains a special regular expression character (such as *, ?, +), use the \ (backslash) character to instruct QuickTest to treat the special characters as literal characters. For more information on regular expressions, see "Regular Expressions Overview" on page 863.

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

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

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

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

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



Retrieving Child Objects

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

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

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

Set MySubSet=TestObject.ChildObjects(MyDescription)

Example

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

```
Set MyDescription = Description.Create()
MyDescription("html tag").Value = "INPUT"
MyDescription("type").Value = "checkbox"
Set Checkboxes =
Browser("Itinerary").Page("Itinerary").ChildObjects(MyDescription)
NoOfChildObjs = Checkboxes.Count
For Counter=0 to NoOfChildObjs-1
    Checkboxes(Counter).Set "ON"
Next
```

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



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

Using the Index Property in Programmatic Descriptions

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

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

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

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

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

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

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

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

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

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



Programmatic Description Checks

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

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

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

Example

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

Using the **GetROProperty** operation:

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

Using the **Exist** operation:

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

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

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

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

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

Note: For information on using the **GetROProperty** operation, see "Retrieving Native Properties" on page 963. For information on using **While...Wend** statements, see "While...Wend Statement" on page 984. For information on specific test objects, operations, and properties, see the *HP QuickTest Professional Object Model Reference*.

Opening and Closing Applications Programmatically

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

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

You can close most applications using the **Close** method. You can also use **SystemUtil** statements to close applications.

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

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

Note:

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

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

Comments, Control-Flow, and Other VBScript Statements

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

For information on how to use these programming concepts in the Keyword View, see Chapter 26, "User Interface-Based Programming Operations."

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

For more details, see:

- "Comments in VBScript" on page 974
- "Calculations in VBScript" on page 979
- "For...Next Statement" on page 982
- "For...Each Statement" on page 981
- "Do...Loop Statement" on page 981
- "While...Wend Statement" on page 984
- "If...Then...Else Statement" on page 983
- "With Statement" on page 985

Retrieving and Setting Identification Property Values

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

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

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

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

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

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

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

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

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

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

Tip: If you do not know the identification properties of objects in your application, you can view them using the Object Spy. For information on the Object Spy, see "Object Spy Dialog Box" on page 151.

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

Native Properties and Operations

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

You can view the native properties and their values, or the identification properties, of the run-time object associated with an object in your application, in the Object Spy dialog box. For details, see "Object Spy Dialog Box" on page 151.

You can use the statement completion feature with object properties to view a list of the available native operations and properties of an object. For more information on the statement completion option, see "Generating Statements in the Expert View or in a Function Library" on page 934.

Tip: If the object is a Web object, you can also reference its native properties in programmatic descriptions using the **attribute/property** notation. For details, see the *HP QuickTest Professional Add-ins Guide*.

This section also includes:

- "Retrieving Native Properties" on page 963
- "Activating Native Operations" on page 963

Retrieving Native Properties

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

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

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

Activating Native Operations

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

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

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

Running DOS Commands

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

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

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

Choosing Which Steps to Report During the Run Session

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

For details, see "Report Modes" on page 987.

Windows API

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

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

For details on how to use the Windows API, see "How to Enhance Your Tests and Function Libraries Using the Windows API" on page 966.

Tasks

How to Navigate in the Expert View and Function Libraries

The following steps describe how to navigate in the Expert View and function libraries.

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

- "Use the Go To dialog box" on page 965
- "Use bookmarks" on page 966
- "Find or find and replace strings" on page 966

Use the Go To dialog box

- 1 Click the **Expert View** tab or activate a function library.
- 2 Select **Edit > Go To**. The Go To dialog box opens. For details, see "Go To Dialog Box" on page 997.

Tip: By default, line numbers are displayed in the Expert View and in function libraries. If they are not displayed, you can select the **Show line numbers** option in the **Tools > View Options > General** tab. For more information on the Editor options, see Chapter 28, "Customizing the Expert View and Function Library Windows."

Use bookmarks

- 1 Click the **Expert View** tab or activate a function library.
- 2 Click in the line to which you want to assign a bookmark.
- 3 Select **Edit > Bookmarks**. The Bookmarks dialog box opens. For details, see "Bookmarks Dialog Box" on page 989.

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

For more details, see "Bookmarks in an Action or Function Library" on page 945.

Find or find and replace strings

For details on using the find dialog box, see "Find Dialog Box" on page 991.

For details on using the Find and Replace dialog box, see "Replace Dialog Box" on page 993.

How to Enhance Your Tests and Function Libraries Using the Windows API

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

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

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

- 5 Call the declared function, passing any required arguments, for example:

```
hwnd = extern.GetForegroundWindow().
```

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

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

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

For example, the **GetWindow** API function expects to receive a numerical value that represents the relationship between the specified window and the window whose handle is to be retrieved. In the MSDN documentation, you can find the constants: **GW_CHILD**, **GW_ENABLEDPOPUP**, **GW_HWNDFIRST**, **GW_HWNDLAST**, **GW_HWNDNEXT**, **GW_HWNDPREV** and **GW_HWNDPREV**.

If you open the **WINUSER.H** file, mentioned in the **GetWindow** documentation, you will find the following flag values:

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

Example

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

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

Reference

Checkpoint and Output Statements

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

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

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

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

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

Note:

- The details about a checkpoint are set in the relevant Checkpoint Properties dialog box. The details about an output value step are set in the relevant Output Value Properties dialog box. The statement displayed in the Expert View is a reference to the stored information. Therefore, you cannot insert a checkpoint or output value statement in the Expert View manually.
 - For more information on inserting and modifying checkpoints, see Chapter 15, "Checkpoints Overview." For more information on inserting and modifying output values, see Chapter 23, "Output Values."
-

Basic VBScript Syntax

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

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

Each VBScript statement has its own specific syntax rules. If you do not follow these rules, errors will be generated when you run the problematic step. Additionally, if you try to move to the Keyword View from the Expert View, QuickTest lists any syntax errors found in the document in the Information pane. You cannot switch to the Keyword View without fixing or eliminating the syntax errors. For more information, see "Information Pane User Interface" on page 1360.



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

General VBScript Syntax Rules and Guidelines

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

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

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

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

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

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

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

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

- **Variables.** You can specify variables to store strings, integers, arrays and objects. Using variables helps to make your script more readable and flexible. For more information, see "Variables in VBScript" on page 980.
- **Parentheses.** To achieve the desired result and to avoid errors, it is important that you use parentheses () correctly in your statements. For more information, see "Parentheses in VBScript" on page 977.
- **Indentation.** You can indent or outdent your script to reflect the logical structure and nesting of the statements. For more information, see "Formatting VBScript Text" on page 973.
- **Comments.** You can add comments to your statements using an apostrophe ('), either at the beginning of a separate line, or at the end of a statement. It is recommended that you add comments wherever possible, to make your scripts easier to understand and maintain. For more information, see "Formatting VBScript Text" on page 973, and "Comments in VBScript" on page 974.
- **Spaces.** You can add extra blank spaces to your script to improve clarity. These spaces are ignored by VBScript.

For more information on using specific VBScript statements to enhance your tests or function libraries, see "Comments, Control-Flow, and Other VBScript Statements" on page 960.



Handling VBScript Syntax Errors

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

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

Tips:



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

The Information pane lists the syntax errors found in your document, and enables you to locate each syntax error so that you can correct it. For details on the information pane, see "Information Pane User Interface" on page 1360.

 **Formatting VBScript Text**

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

Comments

Use comments to explain sections of a script. This improves readability and make tests and function libraries easier to maintain and update. For more information, see "Comments in VBScript" on page 974.

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

Tips:

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

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



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

Indentation

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

- **Indenting Statements.** You can indent your statements by selecting the text and choosing **Edit > Advanced > Indent** or by press the TAB key. The text is indented according to the tab spacing selected in the Editor Options dialog box, as described in "General Tab (Editor Options Dialog Box)" on page 1004.
-

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

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

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



Comments in VBScript

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

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

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

By default, comments are displayed in green in the Expert View and in function libraries. You can customize the appearance of comments in the Editor Options dialog box. For more information, see "Fonts and Colors Tab (Editor Options Dialog Box)" on page 1007.

Tips:

- You can comment a block of text by choosing **Edit > Advanced > Comment Block** or by clicking the **Comment Block** button.
 - To remove the comment, select **Edit > Advanced > Uncomment Block** or click the **Uncomment Block** button.
-



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



Parameter Indications in VBScript

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

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

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

Object_Hierarchy.Method DataTable (parameterID, sheetID)

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

Example

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

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

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

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

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

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

Welcome: Mercury Tours			
Welcome: Mercury Tours			
Find a Flight: Mercury			
toPort	Select	DataTable("departure", dtGlobalSheet)	Select the <the value of the 'departure

For more information on using and defining parameter values, see Chapter 22, "Parameterizing Values."

Parentheses in VBScript

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

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

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

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

Example

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

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

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

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

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

or

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

```
...
```

```
...
```

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

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

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

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

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

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



Calculations in VBScript

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

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

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

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

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

'Multiplies the number of passengers by 100

```
weight = passenger * 100
```

'Inserts the maximum weight into a message box.

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



Variables in VBScript

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

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

```
Set ObjectVar = ObjectHierarchy
```

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

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

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

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

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

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

Do...Loop Statement

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

```
Do [{while} {until} condition]
    statement
Loop
```

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

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

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

For...Each Statement

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

```
For Each item In array
    statement
Next
```

Item	Description
<i>item</i>	A variable representing the element in the array.

Item	Description
<i>array</i>	The name of the array.
<i>statement</i>	A statement, or series of statements, to be performed during the loop.

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

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

Note: During a run session, if a **For Each** statement iterates on the **ParameterDefinitions** collection, the run may fail if the collection was retrieved directly before using the **For Each** statement. To prevent this, use other VBScript loop statements, such as **For** or **While**.

For...Next Statement

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

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

Item	Description
<i>counter</i>	The variable used as a counter for the number of iterations.
<i>start</i>	The start number of the counter.
<i>end</i>	The last number of the counter.

Item	Description
<i>step</i>	The number to increment at the end of each loop. Default = 1. Optional.
<i>statement</i>	A statement, or series of statements, to be performed during the loop.

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

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

If...Then...Else Statement

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

```
If condition Then
    statement
Elseif condition2 Then
    statement
Else
    statement
End If
```

Item	Description
<i>condition</i>	Condition to be fulfilled.
<i>statement</i>	Statement to be perform.

Example

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

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

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

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

While...Wend Statement

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

```
While condition
    statement
Wend
```

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

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

```
passengers = Browser("Mercury Tours").Page("Find Flights").
    WebEdit("numpassengers").GetROProperty("value")
While passengers < 10
    passengers = passengers + 1
Wend
msgbox("The number of passengers in the party is " & passengers)
```

With Statement

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

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

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

```
With object
    statements
End With
```

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

Example

you could replace this script:

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

with the following:

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

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

Note: In addition to entering **With** statements manually, you can also instruct QuickTest to automatically generate **With** statements as you record or to generate **With** statements for an existing test. For more information, see "How to Generate With Statements for Your Test" on page 906.



Report Modes

For details on using report modes, see "Choosing Which Steps to Report During the Run Session" on page 964.

The following report modes are available:

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

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

```
Reporter.Filter = rfDisableAll
```

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

```
Reporter.Filter = rfEnableAll
```

- To instruct QuickTest to include only subsequent failed steps in the Run Results, enter:

```
Reporter.Filter = rfEnableErrorsOnly
```

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

```
Reporter.Filter = rfEnableErrorsAndWarnings
```

- To retrieve the current report mode, enter:

```
MyVar=Reporter.Filter
```

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

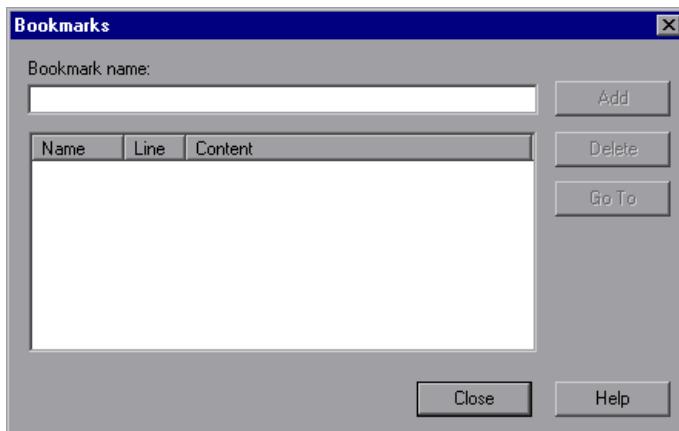
Expert View and Function Library Window User Interface

This section includes:

- Bookmarks Dialog Box on page 989
- Find Dialog Box on page 991
- Replace Dialog Box on page 993
- Regular Expressions in the Find and Replace Dialog Boxes on page 996
- Go To Dialog Box on page 997
- Select a Keyword Dialog Box on page 998

Bookmarks Dialog Box

This page dialog box enables you to add, delete, and go to bookmarks in the Expert View or a function library.



To access	1 Click the Expert View tab or activate a function library. 2 Click in the line to which you want to assign a bookmark. 3 Select Edit > Bookmarks .
Relevant tasks	"How to Navigate in the Expert View and Function Libraries" on page 965
See also	"Bookmarks in an Action or Function Library" on page 945

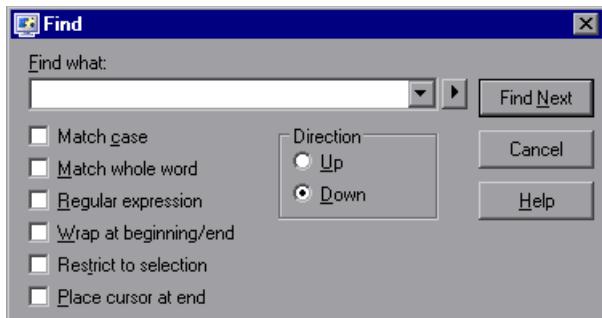
User interface elements are described below (unlabeled elements are shown in angle brackets):

UI Elements	Description
Bookmark name	The name for a new bookmark. Enter a unique name for the bookmark and click Add . The bookmark is added to the Bookmarks dialog box, together with the line number at which it is located and the textual content of the line. In addition, a bookmark icon  is added to the left of the selected line in the Expert View or function library.
< Bookmark list >	The list of bookmarks to navigate to. Select a bookmark and click the Go To button to jump to the bookmarked line.

Find Dialog Box

This dialog box enables you to search for strings in the current action in the Expert View or in a function library. You can also search for strings in the Edit HTML Source and Edit HTML Tags dialog boxes of Page checkpoints, and in the "With" Generation Results dialog box.

For more information on the With Generation Results dialog box, see "How to Generate With Statements for Your Test" on page 906. For more information on Page checkpoints, see the section on Page checkpoints in the *HP QuickTest Professional Add-ins Guide*.



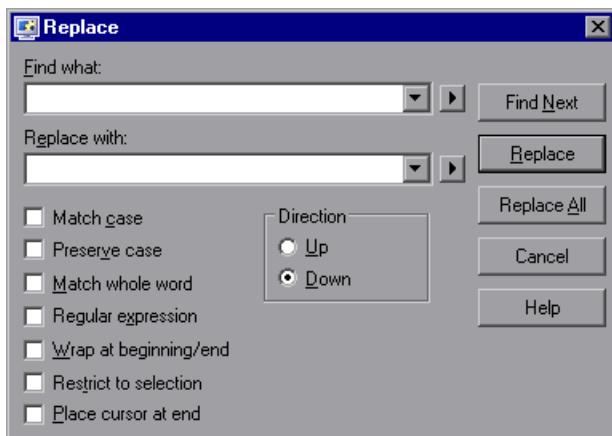
To access	<p>In the Expert View or function library, click the Find button .</p> <p>In the Expert View, you can also perform one of the following:</p> <ul style="list-style-type: none"> ▶ Select Edit > Advanced > Apply "With" to Script, and then press CTRL+F. ▶ In the Page Checkpoint Properties dialog box, click Edit HTML Source or Edit HTML Tags, and then right-click and select Find in the displayed dialog box.
Relevant tasks	<p>"How to Navigate in the Expert View and Function Libraries" on page 965</p>

User interface elements are described below (unlabeled elements are shown in angle brackets):

UI Elements	Description
Find what	The text string you want to locate. For information on searching for a regular expression, see < Regular Expression Arrow > below.
	< Regular Expression Arrow >. If you want to use regular expressions in the string you specify, click the arrow button  and select a regular expression from the list. This automatically inserts the regular expression in the Find what box at the cursor location and selects the Regular expression check box. For more information, see "Regular Expressions in the Find and Replace Dialog Boxes" on page 996.
Match case	Distinguishes between upper-case and lower-case characters in the search. When Match case is selected, QuickTest finds only those occurrences in which the capitalization matches the text you entered in the Find what box exactly.
Match whole word	Searches for occurrences that are only whole words and not part of longer words.
Regular expression	Treats the specified text string as a regular expression. This option is automatically selected when you select a regular expression from the list.
Wrap at beginning/end	Continues the search from the beginning or end of the action, dialog box, or function library text when either the beginning or end is reached, depending on the selected search direction.
Restrict to selection	Searches only within the selected part of the action, dialog box, or function library text.
Place cursor at end	Places the cursor at the end of the highlighted occurrence when the search string is located.
Direction	Up/Down. Specify the direction in which you want to search, from the current cursor location in the action, dialog box, or function library.

Replace Dialog Box

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



To access	<p>In the Expert View or function library, click the Replace button .</p> <p>In the Expert View, you can also:</p> <p>Click Edit HTML Source or Edit HTML Tags, and then right-click and select Replace in the displayed dialog box.</p>
Relevant tasks	<p>"How to Navigate in the Expert View and Function Libraries" on page 965</p>

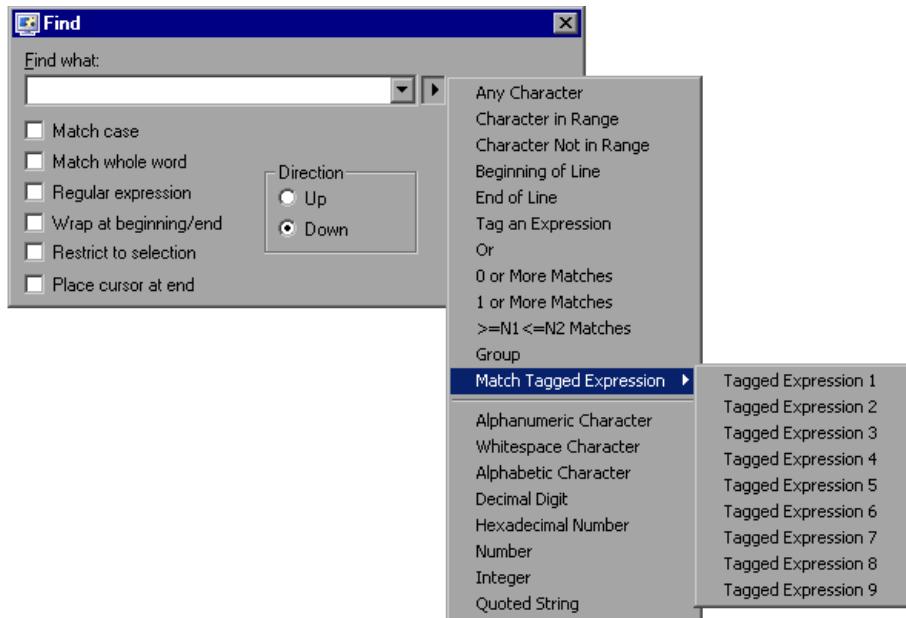
User interface elements are described below:

UI Elements	Description
Find what	The text string you want to locate. For information on searching for a regular expression, see < Regular Expression Arrow > below.
Replace with	The text string you want to use to replace the found text. For information on searching for a regular expression, see < Regular Expression Arrow > below.
	< Regular Expression Arrow >. If you want to use regular expressions in the Find what or Replace with string, click the arrow button  and select a regular expression from the list. This automatically inserts the regular expression in the Find what or Replace with box at the cursor location and selects the Regular expression check box. For more information, see "Regular Expressions in the Find and Replace Dialog Boxes" on page 996.
Match case	Distinguishes between upper-case and lower-case characters in the search. When Match case is selected, QuickTest finds only those occurrences in which the capitalization matches the text you entered in the Find what box exactly.
Preserve case	Checks each occurrence of the Find what string for all lowercase, all uppercase, sentence caps or mixed case. The Replace with string is converted to the same case as the occurrence found, except when the occurrence found is mixed case. In this case, the Replace with string is used without modification.
Match whole word	Searches for occurrences that are only whole words and not part of longer words.
Regular expression	Treats the specified text string as a regular expression. This option is automatically selected when you select a regular expression from the list.

UI Elements	Description
Wrap at beginning/end	Continues the search from the beginning or end of the action, dialog box, or function library text when either the beginning or end is reached, depending on the selected search direction.
Restrict to selection	Searches only within the selected part of the action, dialog box, or function library text.
Place cursor at end	Places the cursor at the end of the highlighted occurrence when the search string is located.
Direction	Up/Down. Specify the direction in which you want to search, from the current cursor location in the action, dialog box, or function library.

Regular Expressions in the Find and Replace Dialog Boxes

You can use regular expressions in the **Find what** and **Replace** with strings to enhance your search. For a general understanding of regular expressions, see "Regular Expressions Overview" on page 863. Note that there are differences in the expressions supported by the Find and Replace dialog boxes and the expressions supported in other parts of QuickTest.



To access	Click the arrow button  next to the Find what field in the Find or Replace dialog box.
Relevant tasks	"How to Navigate in the Expert View and Function Libraries" on page 965

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

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

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

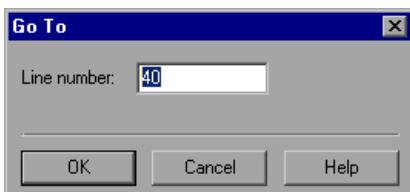
Select **Tag an Expression** from the regular expressions list to insert parentheses "()" to indicate a tagged expression in the search string.

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



Go To Dialog Box

This dialog box enables you to navigate to a specific line in an action or in a function library.



To access	1 Click the Expert View tab or activate a function library. 2 Select Edit > Go To .
Relevant tasks	"How to Navigate in the Expert View and Function Libraries" on page 965

User interface elements are described below:

UI Elements	Description
Line number	The line to which you want to navigate.

Select a Keyword Dialog Box

This dialog box enables you to select a keyword from a list of keywords for use with the Auto Expand VBScript syntax option.



To access	Enter two or more characters in the Expert View that are the initial characters of multiple keywords.
Important information	The Auto-expand VBScript syntax option (enabled by default) must be enabled to use the Select a Keyword dialog box. You can disable or enable this option in the Editor Options dialog box. For more information, see "General Tab (Editor Options Dialog Box)" on page 1004.
See also	"Automatic Completion of VBScript Syntax" on page 944

User interface elements are described below (unlabeled elements are shown in angle brackets):

UI Elements	Description
<Keyword list>	A list of the keywords whose first two characters match the first two characters you entered in the Keyword View.

28

Customizing the Expert View and Function Library Windows

This chapter includes:

Concepts

- Expert View and Function Library Window Customization Options on page 1002

Reference

- General Tab (Editor Options Dialog Box) on page 1004
- Fonts and Colors Tab (Editor Options Dialog Box) on page 1007
- Key Binding Tab (Editor Options Dialog Box) on page 1009

Concepts

Expert View and Function Library Window Customization Options

QuickTest includes a powerful editor for scripts in the Expert View and functions libraries in the Function Library window. You can modify and customize many aspects of the editor through the Editor Options dialog box. Any changes you make are applied globally to the Expert View and to all function library windows.

The Editor Options dialog box consists of three tabs:

- **General.** Customizes the way in which the Expert View and function library windows behave when you edit a script or function library. You can show or hide character symbols, and choose to display line numbers. For more information on using the Expert View, see Chapter 27, "Working in the Expert View and Function Library Windows." For more information on working with function libraries, see Chapter 29, "User-Defined Functions and Function Libraries."
- **Fonts and Colors.** Customizes script and function library element appearance. Changes the color of different elements, including comments, strings, QuickTest reserved words, operators, and numbers. Each element of QuickTest tests and function libraries can be displayed in a different color. You can also specify the font style and size to use for all elements. You can create your own personalized color scheme for each element.
- **Key Binding.** Personalizes default keyboard shortcuts you use for editing. QuickTest includes keyboard shortcuts that let you move the cursor, delete characters, and cut, copy, or paste information to and from the Clipboard. You can replace these shortcuts with your preferred shortcuts.

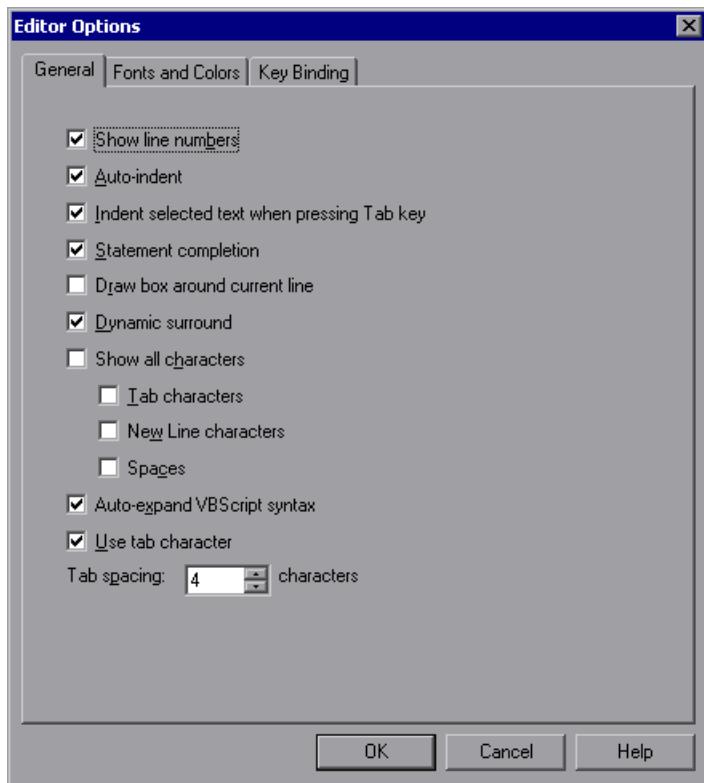
You can also modify the way your script or function library is printed using options in the Print dialog box. For more information, see "Print Dialog Box" on page 428 and "How to Manage Function Libraries" on page 1028.

For more information on using the Expert View, see Chapter 27, "Working in the Expert View and Function Library Windows." For more information on working with function libraries, see Chapter 29, "User-Defined Functions and Function Libraries."

Reference

General Tab (Editor Options Dialog Box)

This tab enables you to customize how scripts and function libraries are displayed in the Expert View and function library windows.



To access	When the Expert View or a function library window is active, select Tools > View Options > General tab.
See also	<ul style="list-style-type: none"> ➤ "Expert View and Function Library Window Customization Options" on page 1002 ➤ "Fonts and Colors Tab (Editor Options Dialog Box)" on page 1007 ➤ "Key Binding Tab (Editor Options Dialog Box)" on page 1009

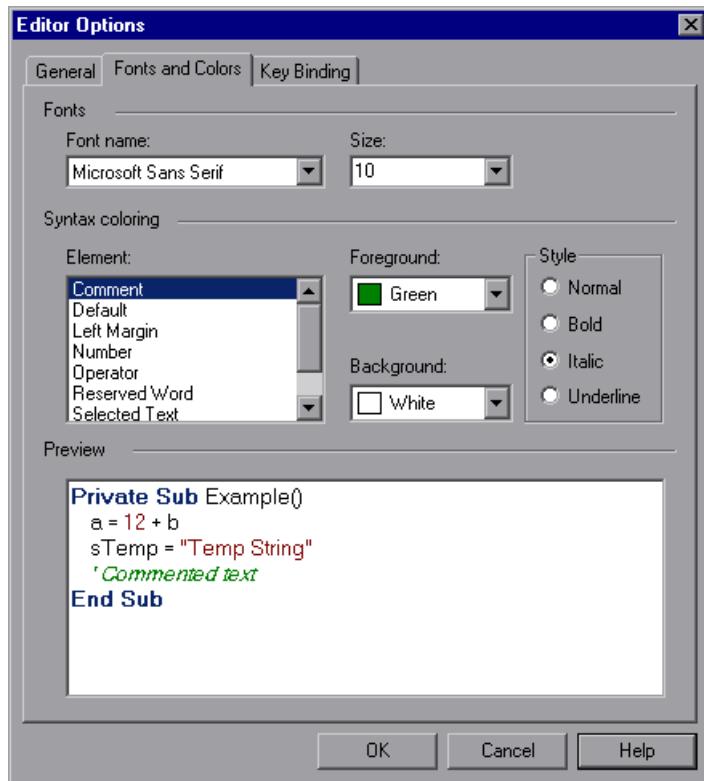
User interface elements are described below:

UI Elements	Description
Show line numbers	Displays a line number to the left of each line in the script or function.
Auto-indent	Causes lines following an indented line to automatically begin at the same point as the previous line. You can press the HOME key on your keyboard to move the cursor back to the left margin.
Indent selected text when pressing Tab key	Indents selected text when pressing the TAB key. When this option is not enabled, pressing the Tab key replaces the selected text with a single Tab character.
Statement completion	<p>Enables Intellisense, the statement completion feature included with QuickTest. When you type in the Expert View or a function library, IntelliSense enables you to select items for your statement from a drop-down list and view the relevant syntax.</p> <p>For more information on using the statement completion (IntelliSense) feature, see "Statement Completion (IntelliSense)" on page 934.</p>
Draw box around current line	Displays a box around the line of the test in which the cursor is currently located.

UI Elements	Description
Dynamic surround	Surrounds existing lines of code with a block structure, enabling you to dynamically expand (or collapse) block statements. For example, when you add a surrounding statement (such as if/while) before existing code, you can use the arrow keys to expand the block to include subsequent lines. These lines are then automatically indented to the correct levels.
Show all characters	Displays all TAB, NEW LINE, and SPACE character symbols. You can also select to display only some of these characters by selecting or clearing the relevant check boxes.
Auto-expand VBScript syntax	<p>Automatically recognizes the first two characters of keywords and adds the relevant VBScript syntax or blocks to the script, when you type the relevant keyword.</p> <p>For example, if you enter the letters if and then enter a space at the beginning of a line, QuickTest automatically enters:</p> <pre>If Then End If</pre>
Use tab character/ Tab spacing	Inserts a TAB character when the TAB key on the keyboard is used. When this option is not enabled, the specified number of space characters is inserted when you press the TAB key.

Fonts and Colors Tab (Editor Options Dialog Box)

This tab enables you to display the comments, strings, QuickTest and VBScript reserved words, operators, and numbers in tests and function libraries, in different colors.



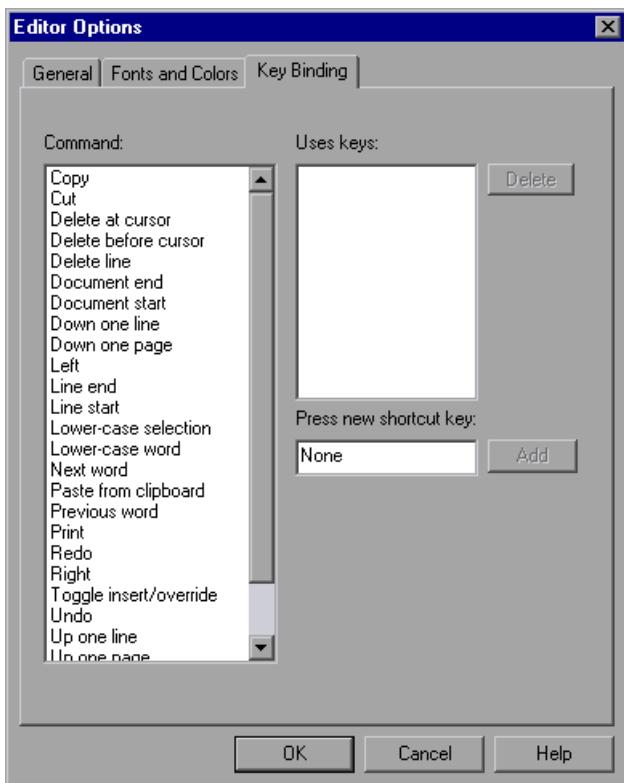
To access	When the Expert View or a function library window is active, select Tools > View Options > Fonts and Colors tab.
See also	<ul style="list-style-type: none"> ➤ "Expert View and Function Library Window Customization Options" on page 1002 ➤ "General Tab (Editor Options Dialog Box)" on page 1004 ➤ "Key Binding Tab (Editor Options Dialog Box)" on page 1009

User interface elements are described below:

UI Elements	Description
Fonts	Select the Font name and Size that you want to use to display all elements. By default, the editor uses the Microsoft Sans Serif font, which is a Unicode font. Note: When testing in a Unicode environment, you must select a Unicode-compatible font. Otherwise, elements in your test or function library may not be correctly displayed in the Expert View or function library windows. However, the test or function library will still run in the same way, regardless of the font you choose. If you are working in an environment that is not Unicode-compatible, you may prefer to choose a fixed-width font, such as Courier, to ensure better character alignment.
Syntax coloring	Select an Element whose appearance you want to customize in QuickTest tests and function libraries. Choose a Foreground color and a Background color. Choose a font Style for the element.
Preview	An example of your change.

Key Binding Tab (Editor Options Dialog Box)

This tab enables you to personalize the default keyboard shortcuts you use for editing. QuickTest includes keyboard shortcuts that let you move the cursor, delete characters, and cut, copy, or paste information to and from the Clipboard. You can replace these shortcuts with your preferred shortcuts.



To access	When the Expert View or a function library window is active, select Tools > View Options > Key Binding tab.
Important information	The default QuickTest menu shortcut keys override any key bindings that you may define. For example, if you define the Paste command key binding to be CTRL+P , it will be overridden by the default QuickTest shortcut key for opening the Print dialog box (corresponding to the File > Print option). For a list of QuickTest menu shortcut keys, see the menu commands available from "QuickTest Commands" on page 82.
See also	<ul style="list-style-type: none"> ▶ "Expert View and Function Library Window Customization Options" on page 1002 ▶ "General Tab (Editor Options Dialog Box)" on page 1004 ▶ "Fonts and Colors Tab (Editor Options Dialog Box)" on page 1007

User interface elements are described below:

UI Elements	Description
Command	The list of commands to which a keyboard shortcut can be assigned.
Uses keys	The list of keyboard shortcuts for the selected command. To delete a key sequence from the list, select the command in the Command list, then highlight the keys in the Uses keys list, and click Delete .
Press new shortcut key	The new shortcut key for the selected command. Click in the Press new shortcut key box and then press the keys you want to use for the selected command and click Add . If the key combination you specify is not supported, or if it is already defined for another command, a message displays below the shortcut key box.

29

User-Defined Functions and Function Libraries

Note: The terms function, method, and operation are used interchangeably in this chapter.

This chapter includes:

Concepts

- Function Library Overview on page 1013
- Associated Function Libraries on page 1015
- User-Defined Functions on page 1017
- User-Defined Function Storage and Access on page 1019
- User-Defined Function Registration on page 1020
- Executing Externally-Defined Functions from Your Test on page 1026

Tasks

- How to Manage Function Libraries on page 1028
- How to Edit a Function Library on page 1033
- How to Manage Function Library Associations on page 1036
- How to Work with a User-Defined Function on page 1039

- How to Create and Register a User-Defined Function Using the Function Definition Generator on page 1042
- How to Execute an Externally-Defined Function from Your Test on page 1048

Reference

- Function Definition Generator Dialog Box on page 1050

Troubleshooting and Limitations - Function Libraries on page 1059

Concepts

Function Library Overview

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

A **function library** is a separate QuickTest document that contains Visual Basic script. Any text file written in standard VBScript syntax can be used as a function library.

Your function libraries can contain:

- **Function definitions (function signature and code).** You can call these functions from other functions or from an action in your test after you associate the function library with the test.
- **VBScript statements.** These are statements that are not contained within function definitions (for example, **RegisterUserFunc** statements). QuickTest runs all of these statements when it loads the function library.

Loading Function Libraries

At the beginning of a run session, QuickTest loads all of the function libraries associated with your test.

In addition, you can use the **LoadFunctionLibrary** statement to dynamically load a function library during a run session.

Editing Function Libraries

When you edit function libraries you can do any of the following:

- Use QuickTest to modify and debug any existing function libraries (such as **.vbs**, **.txt**, or **.qfl** files). For information on using VBScript, see "Handling VBScript Syntax Errors" on page 972 and "Basic VBScript Syntax" on page 970.

- Open and work on one or several function libraries at the same time, as each function library opens in a separate window.
- Choose to open a function library in edit mode or read-only mode:
 - **Edit mode.** Enables you to view and modify the function library. While the function library is open on your computer, other users can view the file in read-only mode, but they cannot modify it.
 - **Read-only mode.** Enables you to view the function library but not modify it. By default, when you open a function library that is currently open on another computer, it opens in read-only mode. You can also choose to open a function library in read-only mode if you want to review it, but you do not want to prevent another user from modifying it.
- Navigate directly from a function call in your document to its function definition in the source document. The function definition can be located either in the same document (test or function library) or in another function library that is associated with your test.

If the document containing the function definition is already open, QuickTest activates the window (brings the window into focus). If the document is closed, QuickTest opens it in read-only mode. For task details, see "Navigate to the definition of a specific function" on page 1031.

- Close a function library after you finish editing it, leaving your QuickTest session open.
- Save the function library to your Quality Center project or to the file system. By default, QuickTest saves a function library with a **.qfl** extension, unless you specify a different extension, such as **.vbs** or **.txt**.

You can also save a function library as an attachment to a test for storage purposes only. To insert function calls from this function library into a test, you must first associate the function library with the test. For more information, see "How to Manage Function Library Associations" on page 1036.

- Use the tools that QuickTest provides for editing and debugging any function library, even if it was created using an external editor. For example, QuickTest can check the syntax of your functions, and the function library window provides the same editing features that are available in the Expert View. For more information on the options available in the Expert View, see Chapter 27, "Working in the Expert View and Function Library Windows."

For task details, see "How to Manage Function Libraries" on page 1028.

Associated Function Libraries

After you create your function libraries, you can associate them with your test. This enables you to insert a call to a public function or subroutine in the associated function library from that test. (Public functions stored in function libraries can be called from any associated test, whereas private functions can be called only from within the same function library.)

You can:

- "Specify default function libraries for all new tests." on page 1015
- "Edit the list of associated function libraries for an existing test." on page 1016
- "Load a function library dynamically during a run session." on page 1016
- "Call a function defined outside of the associated function libraries" on page 1016

Specify default function libraries for all new tests.

You do this in the Test Settings dialog box (**File > Settings > Resources** node). For details, see "Resources Pane (Test Settings Dialog Box)" on page 1475.

After a test is created, the list of default function libraries is integrated into the test. Therefore any changes to the default function libraries list in the Test Settings dialog box do not affect existing tests.

Edit the list of associated function libraries for an existing test.

You do this in the Resources pane or the Test Settings dialog box. For details, see "Resources Pane" on page 1385, and "Resources Pane (Test Settings Dialog Box)" on page 1475.

Load a function library dynamically during a run session.

You do this using the **LoadFunctionLibrary** statement. For details, see the **Utility** section of the *HP QuickTest Professional Object Model Reference*.

Call a function defined outside of the associated function libraries

In addition to the functions available in the associated function libraries, you can also call a function contained in any function library (or VBscript file) directly from any action using the **ExecuteFile** function. You can also insert **ExecuteFile** statements within an associated function library. For more information, see "Executing Externally-Defined Functions from Your Test" on page 1026.

For task details, see "How to Manage Function Library Associations" on page 1036.

Working with Associated Function Libraries in Quality Center

You can associate a function library with your test, regardless of whether the function library is stored in the file system or your Quality Center project. However, if you are planning to use the function library in a business process test or if you are working with the Resources and Dependencies model, you must save it in your Quality Center project.

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

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

User-Defined Functions

If you have segments of code that you need to use several times in your tests, you may want to create a user-defined function. You create the functions in VBScript. For details on using VBScript, see "Handling VBScript Syntax Errors" on page 972 and "Basic VBScript Syntax" on page 970.

A user-defined function encapsulates an activity (or a group of steps that require programming) into a keyword (also called an operation). By using user-defined functions, your tests are shorter, and easier to design, read, and maintain. You can then call user-defined functions from an action by inserting the relevant keywords (or operations) into that action.

A user-defined function is automatically defined as a global function. You can call global functions by selecting them from the lists displayed in the following locations:

- The **Operation** box, when the **Functions** category is selected in the Step Generator
- The **Operation** column, when the **Operation** item is selected from the **Item** list in the Keyword View
- The Expert View, when using IntelliSense

You can also register a user-defined function as a method for a QuickTest test object class. A registered method can either override the functionality of an existing test object method for the duration of a run session, or be registered as a new method for a test object class. You can call test object method by selecting it from the list displayed in the following locations:

- The **Operation** box, when the relevant test object class is selected in the Step Generator
- The **Operation** column, when a test object of the relevant class is selected from the **Item** list in the Keyword View
- The Expert View, when using IntelliSense and typing the name of a test object of the relevant class

For more information on registering user-defined functions, see "User-Defined Function Registration" on page 1020 and "How to Create and Register a User-Defined Function Using the Function Definition Generator" on page 1042.

Before you can call registered or global function from a test, you must add it to the test script or to a function library that is associated with the test.

User-Defined Function Names

When deciding the name for your function, consider the following:

- During run-time, QuickTest first searches the test for the specified function and then searches the function libraries in the order in which they are listed in the Resources pane. If QuickTest finds more than one function that matches the function name in a specific test or function library, it uses the last function it finds in that test or function library. If QuickTest finds two functions with the same name in two different function libraries, it uses the function from the function library that has the higher priority. To avoid confusion, it is recommended that you verify that within the resources associated with a test, each function has a unique name.
- When you create a user-defined function, do not give it the same name as a built-in function (for example, **GetLastError**, **MsgBox**, or **Print**). Built-in functions take priority over user-defined functions, so if you call a user-defined function that has the same name as a built-in function, the built-in function is called instead. For a list of built-in functions, see the **Built-in functions** list in the Step Generator (**Insert > Step Generator**).



User-Defined Function Storage and Access

Using QuickTest, you can define and store your user-defined functions either in a function library (saved as a **.qfl** file, by default) or directly in an action within a test. For details, see "How to Manage Function Libraries" on page 1028.

When you store a public function in a function library and associate the function library with a test, the test can call the public functions in that function library. For more information, see "Associated Function Libraries" on page 1015.

You can access the functions that are stored in an associated function library from the Step Generator, and the Available Keywords pane, and you can enter these functions manually in the Expert View.

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

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

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

If the same function name exists locally within your action and within an associated function library, QuickTest uses the function defined in the action.

User-Defined Function Registration

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

When you register a function to a test object class, you can register the function as a new operation for the test object class, or you can choose to override the functionality of an existing operation. You can unregister the function to disable new operations or to return existing operations to their original QuickTest behavior.

A registered method applies only to the test or function library in which you register it. In addition, QuickTest clears all function registrations at the beginning of each run session. If you call an external action that registers a method (and does not unregister it at the end of the action), the method registration remains in effect for the remainder of the test that called the action.

After you register a function to a test object class, it can be called as a method of that test object class (in addition to being available as a global function). The function is displayed as an operation in the Step Generator when a test object of that class is selected, and in the Keyword View **Operation** list when a test object of that class is selected from the **Item** list, as well as in IntelliSense and in the general **Operation** list in the Step Generator.

When you register a function to a test object class, you can optionally define it as the default operation for that test object class. This instructs QuickTest to display the function in the **Operation** column in the Keyword View, by default, when you choose a test object from the associated class in the **Item** list, and in the Step Generator, when you choose a test object of that class from the **Object** list. It also enables you to select the function from IntelliSense.

This section also includes:

- "Preparing the User-Defined Function for Registration" on page 1022
- "Registering User-Defined Functions as Test Object Methods" on page 1022
- "Unregistering User-Defined Test Object Methods" on page 1023
- "Running an Overriding User-Defined Test Object Method" on page 1024



Preparing the User-Defined Function for Registration

When you run a statement containing a registered method, QuickTest sends the test object as the first argument. For this reason, your user-defined function must have at least one argument. Your user-defined function can have any number of arguments, or it can have only the test object argument.

If you register a user-defined function to override an existing test object method, then after the test object argument, the function must have the same number of arguments as the method it overrides.

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



Registering User-Defined Functions as Test Object Methods

To register a user-defined function as a test object method, you enter a **RegisterUserFunc** statement in your test or function library. The **RegisterUserFunc** statement specifies the test object class, the name of your function, and the name of the test object method that should call your function.

In this statement, you can also instruct QuickTest to use the function as the default operation for the test object class.

You can register the same function to more than one test object class, using the same operation name for different test object classes, or different names.

After the **RegisterUserFunc** statement runs, your method becomes a recognized method of the specified test object class for the remainder of the test, or until you unregister the method.

When QuickTest loads a function library it runs all the statements in the function library. Therefore, if the function you are registering is defined in a function library, it is recommended to include the **RegisterUserFunc** statement in the function library as well so that the method will be immediately available for use in any test using that function library.

For task details, see "Register the function to a test object class - optional" on page 1040.



Unregistering User-Defined Test Object Methods

When you register a method using a **RegisterUserFunc** statement, your method becomes a recognized method of the specified test object class for the remainder of the test, or until you unregister the method.

If your method overrides a QuickTest method, unregistering the method resets the method to its normal behavior. Unregistering other methods removes them from the list of methods supported by the test object class.

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

If you register a method within a reusable action, you should unregister the method at the end of the action (and then re-register it at the beginning of the next action if necessary), so that tests calling your action will not be affected by the method registration.

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

For task details, see "Unregister the function - optional" on page 1042.

Unregistering Functions That Were Registered More Than Once

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

Example:

Suppose you enter the following statements:

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

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

Running an Overriding User-Defined Test Object Method

You can register a user-defined function to (temporarily) override the functionality of an existing test object method for a test object class.

When a user-defined function runs instead of the test object method it overrides, if it calls any overridden test object methods, the standard functionality of those methods is used.

When you call the user-defined function directly, if it calls any overridden test object methods, their overriding user-defined functions are used.

Examples:

The following scenarios demonstrate different situations that are affected by this functionality.

A Registered User Function That Calls the Test Object Method It Overrides

Suppose you want to report the current value of a Web edit box to the run results before you set a new value for it. You can override the standard QuickTest **Set** method with a function that retrieves the current value of an edit box, reports that value to the run results, and then sets the new value of the edit box using the standard **Set** method.

The function (and its registering line) would look something like this:

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

RegisterUserFunc "WebEdit", "Set", "MySet"
```

When a test step uses the **WebEdit.Set** method, the overriding **MySet** function runs, and in turn, calls the original QuickTest WebEdit **Set** method.

However, when a test step uses the **MySet** function, the function runs and calls the overridden **WebEdit.Set** method, running the **MySet** function once more. This time, **MySet** calls the original QuickTest WebEdit **Set** method.

A Registered User Function That Calls a Test Object Method Overridden by Another Function

Suppose you want to override the VbButton's standard **Click** method to always perform a double click. In addition, you want to override the standard QuickTest **DblClick** method with a function that retrieves the text of the button and reports it to the run results before double-clicking the button.

The functions (and their registering lines) would look something like this:

```
Function MyDblClick (obj, x, y, button)
    dim button_name
    button_name = obj.GetROProperty("text")
    Reporter.ReportEvent micDone, "Clicking", button_name
    obj.DblClick x, y, button
End Function
RegisterUserFunc "VbButton", "DblClick", "MyDblClick"

Function MyClick (obj, x, y, button)
    obj.DblClick x, y, button
End Function
RegisterUserFunc "VbButton", "Click", "MyClick"
```

When a test step uses the **VbButton.Click** method, the overriding **MyClick** function runs. In this situation, **MyClick** will then run the original QuickTest VbButton **DblClick** method.

When a test step uses the **MyClick** function, the function runs and calls the overridden **VbButton.DblClick** method, running **MyDblClick**. **MyDblClick** reports the button text to the run results and then calls the original QuickTest VbButton **DblClick** method.

To ensure that the **MyClick** function always runs the overridden behavior for **DblClick** method, you could call **MyDblClick** directly within **MyClick**.

Executing Externally-Defined Functions from Your Test

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

When you run an **ExecuteFile** statement within an action, you can call the functions in the file only from the current action. Functions in a VBScript file are available to your entire test, only if they are stored in a function library that is associated with your test. For more information, see "Associated Function Libraries" on page 1015.

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

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

For task details, see "How to Execute an Externally-Defined Function from Your Test" on page 1048.

Tasks

How to Manage Function Libraries

This task describes the different activities you can perform to manage function libraries in QuickTest. If a function is stored in a function library, you must associate the function library with a test before you can call the function from the test.

This task includes the following steps:

- "Create a function library" on page 1028
- "Open a function library" on page 1029
- "Edit a function library" on page 1029
- "Enable editing for a read only function library" on page 1029
- "Debug a function library" on page 1030
- "Navigate between open QuickTest documents" on page 1031
- "Navigate to the definition of a specific function" on page 1031
- "Associate a function library with a test" on page 1031
- "Save a function library" on page 1032
- "Print a function library" on page 1032
- "Close a function library" on page 1033

Create a function library

Click the **New** button down arrow and select **Function Library**.

You can also:

- Create a new function library file from the Test Resources module in Quality Center. For details, see the Quality Center or HP ALM user guide.
- Create function libraries outside of QuickTest in any editor and save them with a **.qfl** extension.

Open a function library

Click the **Open** button down arrow and select **Function Library**. The Open Function Library dialog box opens, enabling you to open a function library from the file system or from a Quality Center project.

You can choose to open a function library in edit mode or read-only mode. For details, see "Open <Resource> Dialog Box" on page 406.

Tips:

- You can open a function library only if you have read or read-write permissions for the file.
 - If the function library was recently created or opened, you can select it from the recent files list in the **File** menu.
 - If the function library is associated with the open test, you can also open it as follows:
 - In the Resources pane, double-click the function library, or right-click the function library and select **Open Function Library**.
 - In the Available Keywords panes, double-click the function library, or right-click the function library and select **Open Resource**.
 - Select **Resources > Associated Function Libraries**. (If you select a function library that is stored in a Quality Center project, QuickTest must be connected to that project to open the associated function library.)
-

Edit a function library

For details, see "How to Edit a Function Library" on page 1033.

Enable editing for a read only function library



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

Note:

- ▶ You cannot enable editing if the function library is locked by another user or checked in to a Quality Center project.
 - ▶ During a debug session, all documents (such as tests and function libraries) are read-only. To edit a document during a debug session, you must first stop the debug session.
-

Debug a function library

- 1** Associate the function library with a test.
- 2** In your test, insert a call to at least one of the functions defined in the function library.
- 3** Run the test, suspend the run session while in the context of your function library and debug the function library.

For details, see "Debugging Tests and Function Libraries" on page 1205.

Note:

- ▶ During a debug session, all documents are read-only and cannot be edited. To edit a document during a debug session, you must first stop the debug session.
 - ▶ You cannot debug a file that is called using an **ExecuteFile** statement, or any of the functions contained in the file. In addition, when debugging a test that contains an **ExecuteFile** statement, the execution marker may not be correctly displayed.
-

Navigate between open QuickTest documents

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

Tips:



- If not all tabs are displayed due to lack of space, use the left and right scroll arrows in the Document pane to display the required document's tab.
 - To scroll between open QuickTest documents, Press **CTRL+TAB** on your keyboard.
-

Navigate to the definition of a specific function

You can navigate directly from a function call to the function's definition.

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

QuickTest activates the relevant document (if the function definition is located in another function library) and positions the cursor at the beginning of the function definition. If the document is closed, QuickTest opens it in read-only mode.

Associate a function library with a test

For details, see "How to Manage Function Library Associations" on page 1036.

Save a function library

Make sure that the function library you want to save is the active document (You can click the function library's tab to bring it into focus), and click the **Save** button.



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

Save the function library to the file system or to your Quality Center project, or save it as an attachment to a test in a Quality Center project, by selecting the relevant location from the sidebar of the dialog box. For details, see "Save <Resource> Dialog Box" on page 417.

Tips:

- ▶ When you modify a function library, an asterisk (*) is displayed in the title bar until the function library is saved.
 - ▶ You can right-click a function library document's tab and select **Save**.
 - ▶ To save multiple documents, select **Window > Windows**. In the Windows Dialog Box (described on page 1304), select the documents you want to save and click the **Save** button. QuickTest prompts you for the save location for any new files that have not yet been saved.
 - ▶ To save all open documents, select **File > Save All**. QuickTest prompts you to specify a location in which to save any new files that have not yet been saved.
-

Print a function library



Click the **Print** button. The Print Dialog Box (described on page 428) opens, enabling you to print the function library and optional additional information.

Close a function library



Make sure that the function library you want to save is the active document (you can click the function library's tab to bring it into focus) and Click the **Close** button in the top right corner of the function library window.

Tips:

- When you modify a function library, an asterisk (*) is displayed in the title bar until the function library is saved.
 - If you don't want to save it, you can right-click a function library document's tab and select **Close**.
 - To close multiple documents, select **Window > Windows**. In the Windows Dialog Box (described on page 1304), select the documents you want to close and click the **Close Window(s)** button. QuickTest prompts you for the save location for any new files that have not yet been saved.
 - To close all open function libraries, **File > Close All Function Libraries**, or **Window > Close All Function Libraries**.
-



How to Edit a Function Library

The following steps describe how to edit a function library using the QuickTest editing features that are available in the Expert View:

- "Add steps manually" on page 1034
 - "Add steps using the Step Generator" on page 1034
 - "Drag and drop a function" on page 1035
 - "Check the syntax of the code in your function library" on page 1035
-

Note: This task is part of a higher-level task. For details, see "How to Manage Function Libraries" on page 1028.

Add steps manually

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

QuickTest applies the same formatting to your function library as it does to content in the Expert View. For details on modifying this formatting, see "Customizing the Expert View and Function Library Windows" on page 1001.

IntelliSense is available for all functions defined in your action or for public functions defined in associated function libraries.

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

Add steps using the Step Generator

The Step Generator dialog box (described in the *HP QuickTest Professional User Guide*) enables you to add steps that contain **reserved objects** (the objects that QuickTest supplies for enhancement purposes, such as utility objects), VBScript functions (such as **MsgBox**), utility statements (such as **Wait**), and user-defined functions that are defined in the same function library.

Drag and drop a function

You can drag and drop a function (or part of it) within the same document, or from one document to another.

To drag and drop a function from one document to another:

- 1 Separate the tabbed documents into separate document panes by clicking the **Restore Down** button (located below the QuickTest window's **Restore Down / Maximize** button).
- 2 Select the relevant lines and drag and drop them from one document to another.

Check the syntax of the code in your function library



Click the **Check Syntax** button. QuickTest checks the syntax of all the code in your function library. This may include function definitions and other VBScript statements.

Tips:

- For information on using VBScript, see "Basic VBScript Syntax" on page 970.
 - To check the syntax for all function libraries associated with your test, click the **Check Syntax** button in the Resources pane of the Test Settings dialog box (**File > Settings > Resources** node). For more information, see "Resources Pane (Test Settings Dialog Box)" on page 1475.
-

How to Manage Function Library Associations

The following steps describe the different ways that you can associate a function library with an open test or modify existing associations:

- "Associate a function library with a test using the Resources pane" on page 1036
- "Associate the currently active function library with a test" on page 1037
- "Associate a function library with a test using the Test Settings dialog box" on page 1037
- "Modify the priority of an associated function library" on page 1038
- "Remove an associated function library" on page 1038

Note: This task is part of a higher-level task. For details, see "How to Manage Function Libraries" on page 1028.

Associate a function library with a test using the Resources pane

In the Resources pane, right-click the **Associated Function Libraries** node in the tree and select **Associate Function Library**. The Open Function Library dialog box opens. For details, see "Open <Resource> Dialog Box" on page 406.

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

Associate the currently active function library with a test

- 1 Make sure that the test with which you want to associate the function library is open in QuickTest.
- 2 Create or open a function library in QuickTest. (Before continuing to the next step, make sure that the function library you want to associate with the test is the active document—you can click the function library's tab to bring it into focus.) For more information, see "How to Manage Function Libraries" on page 1028.
- 3 Save the function library either in your Quality Center project or in the file system. For more information, see "Save a function library" on page 1032.
- 4 In QuickTest, select **File > Associate Library '<Function Library>' with '<Test>**', or right-click in the function library and select **Associate Library '<Function Library>' with '<Test>'**. QuickTest associates the function library with the open test.

Associate a function library with a test using the Test Settings dialog box



- 1 In the Test Settings dialog box (**File > Settings**), click the **Resources** node in the navigation bar.
- 2 In the **Associated function libraries** list, click the **Add** button. QuickTest displays a browse button enabling you to browse to a function library in the file system. If you are connected to a Quality Center project, QuickTest also adds [QualityCenter] to the file path, indicating that you can browse to a function library either in your Quality Center project or in the file system.



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

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

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



Modify the priority of an associated function library

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

For more information, see "Resources Pane (Test Settings Dialog Box)" on page 1475.

Remove an associated function library

Perform one of the following:



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

For more information, see "Resources Pane (Test Settings Dialog Box)" on page 1475.

How to Work with a User-Defined Function

This task describes how to create and work with a user-defined function that you store in a function library and register to a test object class.

This task includes the following steps:

- "Prerequisites - Open the function library or test" on page 1039
- "Create the function" on page 1039
- "Register the function to a test object class - optional" on page 1040
- "Associate the function library with a test" on page 1041
- "Call the function" on page 1041
- "Unregister the function - optional" on page 1042

1 Prerequisites - Open the function library or test

Create a new function library or test, open an existing one, or click on the tab of an open document to bring it into focus.

If you insert the function in a function library, the function will be accessible to any associated test. If you insert the function directly in a test in the Expert View, it can be called only from within the specific action.

2 Create the function

You can define functions manually or using the Function Definition Generator, which creates the basic function definition for you automatically.

Even if you prefer to define functions manually, you may still want to use the Function Definition Generator to view the syntax required to add header information, register a function to a test object class, or set the function as the default method for that test object class. For an in-depth view of the required syntax, you can define a function using the Function Definition Generator and experiment with the various options.

For details, see "Function Definition Generator Dialog Box" on page 1050.

When writing the code for your function, consider the issues listed in "Troubleshooting and Limitations - Function Libraries" on page 1059. For details on using VBScript, see the Microsoft VBScript Language Reference (**Help > QuickTest Professional Help > VBScript Reference > VBScript**).

Note:

- If you want to register the function to a test object class, define it as a public function, and make sure that it expects the test object as the first argument.
 - If you want to override an existing test object method, make sure that after the test object argument, your function accepts the same number of arguments as the method it overrides.
-

3 Register the function to a test object class - optional

You can register your function as a new method for the test object class, or you can register it using an existing method name to (temporarily) override the existing functionality of the specified method.

You can perform this step manually, or using the Function Definition Generator Dialog Box (described on page 1050):

- Add a **RegisterUserFunc** statement in your test or function library. The name of the test object operation you register cannot contain spaces. In this statement, you can also instruct QuickTest to use the function as the default operation for the test object class.

Example:

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

After this statement runs (in your action or function library), the **MySet** method (operation) is added to the WebEdit test object class using the **MySetFunc** user-defined function, and defined to be the default operation (as specified in the last argument of the statement).

If you choose the WebEdit test object from the **Item** list in the Keyword View, the **MySet** operation will be selected automatically in the **Operation** column. It will also be displayed in the **Operation** list together with other registered and out of the box operations for the WebEdit test object.

For syntax and other examples, see the Utility Statements section (in the Utility Objects section) of the *HP QuickTest Professional Object Model Reference*.

- If you use the Function Definition Generator dialog box to create your function definition, a **RegisterUserFunc** statement is automatically added immediately after the definition if you select the **Register to a test object** option.

You can add additional **RegisterUserFunc** statements manually to register the function to additional test object classes.

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

4 Associate the function library with a test

If you inserted the code in a function library, you must associate the function library with a test to enable access to the user-defined functions.

For details, see "Associate a function library with a test" on page 1031.

5 Call the function

In your test or function library, create steps that call your user-defined function as a global function, or run it by calling the test object method to which it is registered.

6 Unregister the function - optional

If you do not want your function to remain registered until the end of the run session, add an **UnRegisterUserFunc** statement in your test or function library. For syntax and an example, see the **Utility Statements** section (in the **Utility Objects** section) of the *HP QuickTest Professional Object Model Reference*.

Tip: If you register a method within a reusable action, you should unregister the method at the end of the action (and then re-register it at the beginning of the next action if necessary), so that tests calling your action will not be affected by the method registration.

How to Create and Register a User-Defined Function Using the Function Definition Generator

This task provides a general description of each step required to create a user-defined function and register it to a test object class using the Function Definition Generator Dialog Box (described on page 1050). This task also includes examples for some of the steps.

Note: This task is part of a higher-level task. For details, see "How to Work with a User-Defined Function" on page 1039.

This task includes the following steps:

- "Prerequisite - Open the function library or test" on page 1043
- "Open the Function Definition Generator" on page 1043
- "Specify the details for the function definition" on page 1043
- "Register the function to a test object class - optional" on page 1044

- "Add arguments to the function - optional" on page 1045
- "Add documentation details to the function - optional" on page 1045
- "Preview the function definition before finalizing it" on page 1046
- "Insert the function in your active document" on page 1046
- "Add the content (code) of the function" on page 1046
- "Results" on page 1047

1 Prerequisite - Open the function library or test

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

2 Open the Function Definition Generator

Select **Insert > Function Definition Generator**. The Function Definition Generator dialog box opens.

3 Specify the details for the function definition

Specify the function's name, type, and scope in the Function Definition Area.

Example:

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

4 Register the function to a test object class - optional

If you defined a public function and you want to register the function as a test object operation, do the following in the Registration Area (described on page 1054):

- a Select the **Register to a test object** check box.
- b Select the test object from the list of available objects.

Example:

For the sample **VerifyProperty** function, you might want to register it to the **Link** test object.

- c Enter the name of a new operation that you want to add to the test object class, or select an existing operation to specify the operation that you want to override its standard functionality. The name of the method cannot contain spaces.

Example:

For the sample **VerifyProperty** function, you might want to define a new **VerifyProperty** operation.

- d Optionally, specify that this operation is the default operation for test objects of this type.

Tip: If you choose not to register your function at this time, you can manually register it later by adding a **RegisterUserFunc** statement as described in "Register the function to a test object class - optional" on page 1040. You can also add additional **RegisterUserFunc** statements, to register the function to additional test object classes.

5 Add arguments to the function - optional

Specify any arguments that are required for your function to run correctly. For details, see "Arguments Area" on page 1055.

Example:

For the **VerifyProperty** function, registered to a test object class in the example for the previous step, you may want to assign the arguments `prop_name` (the name of the property to check) and `expected_value` (the expected value of the property), in addition to the first argument, `test_object`.

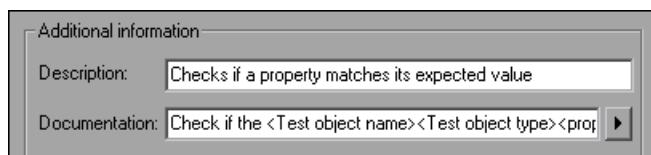
6 Add documentation details to the function - optional

Define **Description** and **Documentation** strings for the test object operation. For details, see "Additional Information Area" on page 1056.

Example:

- For the sample **VerifyProperty** function, you may want to provide the following description: Checks whether a property value matches the actual value.
- If you were checking a link to "HP" from a search engine, you might define the following documentation using the Function Definition Generator:

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



In the Keyword View, after you create a step that calls the **VerifyProperty** operation and choose values for the arguments, the above documentation might appear as follows: Check if the "Management Software" link "text" value matches the expected value: "Business Technology Optimization (BTO) Software".

7 Preview the function definition before finalizing it

As you add information to the Function Definition Generator, the **Preview** area displays the emerging function definition. You can review your function and make any changes, as needed, in the various areas of the dialog box. For details, see "Bottom Area" on page 1058.

Example:

After defining the **VerifyProperty** function as described in the previous steps, the **Preview** area displays the following code:

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

8 Insert the function in your active document

- a** To generate an additional function definition after inserting this one, select **Insert another function definition**.
- b** Click **OK**. QuickTest inserts the generated VBScript code in the active document.

9 Add the content (code) of the function

To finalize the function, add content to the function code, as required, replacing the TODO comment.

When writing the code for your function, consider the issues listed in "Troubleshooting and Limitations - Function Libraries" on page 1059. For details on using VBScript, see the Microsoft VBScript Language Reference (**Help > QuickTest Professional Help > VBScript Reference > VBScript**).

Example:

For example, if you want the function to verify whether the expected value of a property matches the actual property value of a specific test object, you might add the following to the body of the function:

```

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

```

10 Results

- If you inserted the function in a function library, the function will be accessible to any associated test. If you insert the function directly in a test in the Expert View, it can be called only from within the specific action.

To associate the function library with additional tests, see "How to Manage Function Library Associations" on page 1036.

- If you registered the function to a test object, the function is displayed as an operation in the Step Generator when a test object of that class is selected, and in the Keyword View **Operation** list when a test object of that class is selected from the **Item** list, as well as in IntelliSense and in the general **Operation** list in the Step Generator.

You can add additional **RegisterUserFunc** statements, to register the function to additional test object classes. For details, see "Register the function to a test object class - optional" on page 1040.

- ▶ If you specified that the function is the default operation for the test object class, it is displayed in the **Operation** column in the Keyword View, by default, when you choose a test object from the associated class in the **Item** list, and in the Step Generator, when you choose a test object of that class from the **Object** list. It also enables you to select the function from IntelliSense.
- ▶ If you defined a description and a **Documentation** string for the test object operation, they are displayed in the Keyword View and the Step Generator.

How to Execute an Externally-Defined Function from Your Test

This task describes how to call a function that is not stored in your test or in an associated function library.

1 Prerequisites

- a Create a VBScript file using standard VBScript syntax. For more information, see the Microsoft VBScript Language Reference (**Help > QuickTest Professional Help > VBScript Reference > VBScript**).
- b Store the file in any folder that you can access from the computer running your test.

2 Add an **ExecuteFile** statement to an action in your test

Use the following syntax:

ExecuteFile *FileName*

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

The **ExecuteFile** statement utilizes the VBScript **ExecuteGlobal** statement. For details, see the Microsoft VBScript Language Reference (select **Help > QuickTest Professional Help > VBScript Reference > VBScript**).

Tip: To include the same **ExecuteFile** statement in every action you create, you can add the statement to an action template. For more information, see "Create an action template" on page 543.

3 Results

You can now use the functions, subroutines, and so forth, from the specified VBScript file as necessary in the action that contains the **ExecuteFile** statement.

Reference

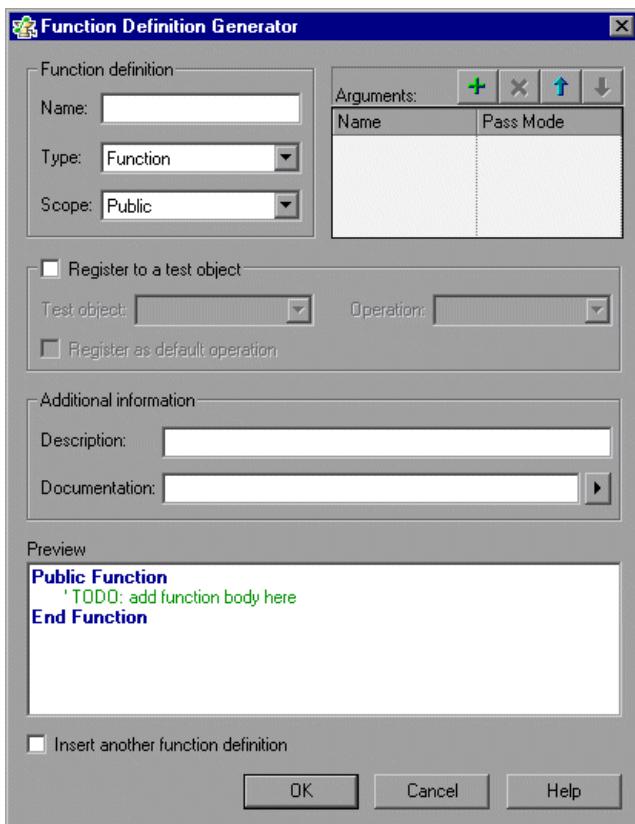
Function Definition Generator Dialog Box

This dialog box enables you to:

- Generate definitions for new user-defined functions.
- Add header information to the function you create.
- Register the functions to a test object class, if needed.

You fill in the required information and the Function Definition Generator creates the basic function definition for you. You complete the function by adding its content (code).

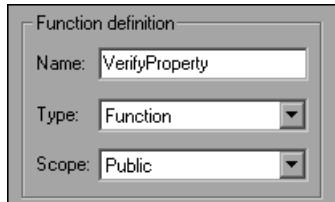
For a task that describes the order of working with this dialog box and provides examples, see "How to Create and Register a User-Defined Function Using the Function Definition Generator" on page 1042.



To access	<p>Select Insert > Function Definition Generator</p> <p>Tip: You can customize the Insert toolbar to display the Function Definition Generator button  to access this dialog box more directly.</p>
------------------	--

Important information	Before you open the dialog box, make sure that the function library or test (Expert View) in which you want to insert the function definition is the active document. (You can click the document's tab to bring it into focus.) This is because the Function Definition Generator inserts the function in the currently active document when you click OK .
Relevant tasks	<ul style="list-style-type: none">▶ "How to Create and Register a User-Defined Function Using the Function Definition Generator" on page 1042▶ "How to Work with a User-Defined Function" on page 1039
See also	<ul style="list-style-type: none">▶ "User-Defined Functions" on page 1017▶ "User-Defined Function Storage and Access" on page 1019▶ "User-Defined Function Registration" on page 1020

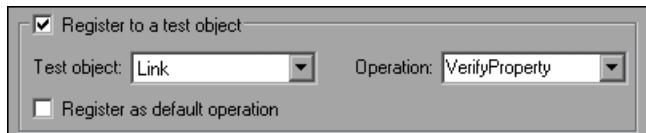
Function Definition Area



User interface elements are described below:

UI Elements	Description
Name	<p>A name for the new function.</p> <p>The name should clearly indicate what the operation does so that it can be easily selected from the Step Generator or the Keyword View.</p> <p>For a list of naming conventions, see "Naming Conventions" on page 1779.</p> <p>Note:</p> <ul style="list-style-type: none"> ➤ Do not use any of the built-in function names (for example, GetLastError, MsgBox, or Print). For a list of built-in functions, see the Built-in functions list in the Step Generator (Insert > Step Generator). ➤ Try to give each function a name that is unique within the resources associated with a specific test. <p>For more details, see "User-Defined Function Names" on page 1019.</p>
Type	<p>The type of function.</p> <p>Possible values:</p> <ul style="list-style-type: none"> ➤ Function ➤ Sub (subroutine)
Scope	<p>The scope of the function.</p> <p>Possible values:</p> <ul style="list-style-type: none"> ➤ Public. The function can be called by any test that is associated with this function library. ➤ Private. The function can be called only from elsewhere in the same function library. <p>Default value: Public</p> <p>Notes:</p> <ul style="list-style-type: none"> ➤ Only public functions can be registered to a test object class. ➤ If you create a user-defined function manually and do not define the scope as Public or Private, it will be treated as a public function, by default.

Registration Area

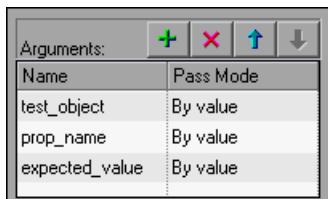


User interface elements are described below:

UI Elements	Description
Register to a test object	<p>Indicates whether to register this function as an operation for a QuickTest test object class.</p> <p>All user-defined functions are available as global operations. If you register the function to a test object class, it can also be called by test objects of that class, and is displayed in the list of available operations for such test objects. For more details, see "User-Defined Function Registration" on page 1020.</p> <p>Note: If you select this option, then when the Function Definition Generator creates your function definition, it also automatically adds a RegisterUserFunc statement with the correct argument values immediately after the definition.</p>
Test object	<p>The test object class (type) to which you want to register the function.</p> <p>Available: When Register to a test object is selected</p>
Operation	<p>The operation name to use for this function.</p> <p>You can select an existing test object operation to override its functionality, or enter a name for a new operation to add to the test object class.</p> <p>Do not include spaces in the test object operation name.</p> <p>Available: When Register to a test object is selected</p>

UI Elements	Description
Register as default operation	<p>Indicates whether the function should be the default operation for the test object class.</p> <p>This instructs QuickTest to display the function in the Operation column, by default, when you or the Subject Matter Expert choose a test object of the associated class in the Item list. It also enables you to select the function from IntelliSense.</p> <p>Note: If you select this option, then when the Function Definition Generator creates your function definition, it specifies the value True as the fourth argument of the RegisterUserFunc statement.</p> <p>Available: When Register to a test object is selected</p>

Arguments Area

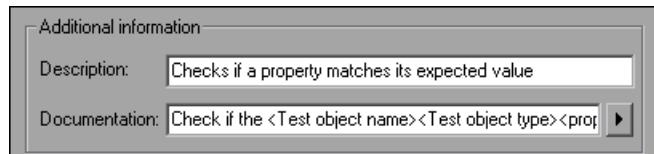


Important information	<p>When calling a function that is registered to a test object class, QuickTest passes the test object as the first argument. Therefore:</p> <ul style="list-style-type: none"> ▶ If you select the Register to a test object check box, the Function Definition Generator automatically adds the argument, test_object, as the first argument in this area. ▶ If you clear the Register to a test object check box, the default test_object argument is automatically removed from this area (unless you renamed it).
------------------------------	--

The user interface elements in this area are described below:

UI Elements	Description
	<p>A toolbar that enables you to add, remove, or re-arrange arguments in the list. You can add as many arguments as you want to the list.</p> <p>Caution:</p> <ul style="list-style-type: none"> ➤ If you are registering the function to a test object class, do not remove the test_object argument, change its location in the list, or modify its Pass Mode. ➤ If you are registering the function to override an existing test object method, then after the test object argument, the function must have the same number of arguments as the method it overrides.
Name	<p>The argument name.</p> <p>The name should clearly indicate the value that needs to be entered for the argument. For a list of naming conventions, see "Naming Conventions" on page 1779.</p>
Pass Mode	<p>Specifies whether the argument will be passed to the function By value or By reference during run-time.</p>

Additional Information Area

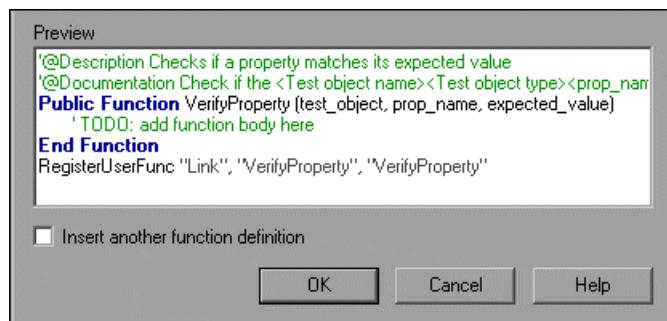


The Function Definition Generator can add header information to your user-defined function definitions, based on the information you add in this area.

User interface elements are described below:

UI Elements	Description
Description	<p>The function's description.</p> <p>If you register the function as a test object operation, the description is displayed as a tooltip in QuickTest when the cursor is positioned over the operation in the Step Generator, in the Keyword View, and when using IntelliSense.</p> <p>Keep the description text brief and clear.</p>
Documentation	<p>A sentence (in the imperative form) that specifies exactly what a step using your function does.</p> <p>If you register the function as a test object operation, the text that you add here is displayed in the Step documentation box of the Step Generator and in the Documentation column in the Keyword View. Therefore, the sentence must be clear and understandable.</p>
	<p>Insert Documentation Element. Displays a list that contains the function's arguments, and the items test object name and test object type, enabling you to include these items in the Documentation text.</p> <p>If you use the argument or test object items in the Documentation text, they are replaced dynamically by the relevant test object names and types or argument values when you create a step that uses the operation.</p> <p>Note: The Test object name and Test object type items are available in the list only if you selected the Register to a test object check box.</p>

Bottom Area



User interface elements are described below:

UI Elements	Description
Preview	<p>Displays the VBScript code that the Function Definition Generator will add to your active document, based on the information that you enter in the dialog box.</p> <p>The code is displayed in read-only format and includes:</p> <ul style="list-style-type: none"> ➤ An empty function definition ➤ Header information for the function documentation (if defined) ➤ A RegisterUserFunc statement (if you selected Register to a test object) <p>The function definition is displayed dynamically, as you enter the information. You can review your function and make any changes, as needed, in the various areas of the dialog box.</p>
Insert another function definition	<p>Specifies whether the dialog box should remain open after you click OK, enabling you to define an additional function.</p>
OK	<p>Inserts the generated VBScript code in the active document and clears the data from the dialog box.</p>

Troubleshooting and Limitations - Function Libraries

This section describes troubleshooting and limitations for user-defined functions and function libraries.

- If you define a VBScript class, it can be called only within the QuickTest action or function library in which you defined it.
- You can use the **RegisterUserFunc** statement to register a user-defined function that will override an existing test object method. You can also register a user-defined function to override a test object method that was created using a QuickTest Professional Extensibility SDK. If you override this type of test object method, the user-defined function must not (recursively) call the test object method that it overrides.
- By default, steps that use user-defined functions are not included in the run results after a run session. If you want the function to appear in the run results, you must add a **Reporter.ReportEvent** statement to the function code. For example, you may want to provide additional information or to modify the test status. For details on the **Reporter.ReportEvent** statement, see the Utility Objects section of the *HP QuickTest Professional Object Model Reference*.

If a step within your user-defined function calls a standard QuickTest test object method, this step will appear in the run results after the run session. However, you can still add a **Reporter.ReportEvent** statement to the function code to provide additional information and to modify the test status, if required.

- If you use a partial run or debug option, such as **Run from step** or **Debug from step**, to begin running a test from a point after method registration was performed in a test step (and not in a function library), QuickTest does not recognize the method registration because it occurred prior to the beginning of the current run session.
- If you delete a function in use from an associated function library, the test step using the function will display the  icon. In subsequent run sessions for the test, an error will occur when the step using the non-existent function is reached.

- To use an **Option Explicit** statement in a function library associated with your test, you must include it in all of the function libraries associated with the test. If you include an **Option Explicit** statement in only some of the associated function libraries, QuickTest ignores all of the **Option Explicit** statements in all function libraries. You can use **Option Explicit** statements directly in your action scripts without any restrictions.
- Each function library must have unique variables in its global scope. If you have two associated function libraries that define the same variable in the global scope using a Dim statement or define two constants with the same name, the second definition causes a syntax error. If you need to use more than one variable with the same name in the global scope, include a Dim statement only in the last function library (since function libraries are loaded in the reverse order).
- Function libraries that run in the same run session must not contain different definitions for the same class. Make sure that each class is defined in only one location.
- If another user modifies a function library that is referenced by a test, or if you modify the function library using an external editor (not QuickTest), the changes will take effect only after the test is reopened.

Part VI

Running and Analyzing Tests

30

QuickTest Run Sessions

This chapter includes:

Concepts

- [Run Sessions - Overview on page 1064](#)
- [Optional Steps on page 1066](#)

Tasks

- [How to Run Your Test on page 1067](#)
- [How to Set Optional Steps on page 1071](#)

Reference

- [Default Optional Steps on page 1072](#)
- [Run Dialog Box: Results Location Tab \(For Tests Stored in the File System\) on page 1073](#)
- [Run Dialog Box: Results Location Tab \(For Tests Stored in Quality Center\) on page 1075](#)
- [Run Dialog Box: Input Parameters Tab on page 1078](#)
- [Test Batch Runner on page 1080](#)

[Troubleshooting and Limitations - Run Sessions on page 1082](#)

Concepts

Run Sessions - Overview

When you run a test, QuickTest performs the steps it contains. If you have defined test parameters, QuickTest prompts you to enter values for them. When the run session is complete, QuickTest displays a report detailing the results. For details on viewing run results, see "Run Results Viewer" on page 1085.

Run Preferences

QuickTest always runs a test from the first step, unless you specify otherwise. You can:

- Run the entire test from the beginning.
- Run only a part of a test using the **Run from Step**, **Run from Action**, **Run to Step** or **Run to Action** options. These features are useful if you want to check a specific section of the test or to confirm that a certain part of your test runs smoothly, without running the test from the beginning or to the end. The **Run to Step** and **Run to Action** options are also useful if you want to open your application to a specific location to add steps or debug your test.
- Debug a section of a test using the **Debug from step** or **Debug from Action** option. (Make sure that the application is open to the relevant location before using this option.) For details, see "Running to a Step and Debugging from a Step" on page 1209.
- Run an iteration of a single action using the **Run Current Action** option. Even though this option runs only one iteration, if the action contains nested actions, QuickTest runs the nested actions for the defined number of iterations.
- Run only one iteration of your entire test by selecting **Run one iteration only** from the Run pane in the Test Settings dialog box. For details, see "Run Pane (Test Settings Dialog Box)" on page 1471.

- Designate certain steps as **optional**, to enable QuickTest to bypass them instead of aborting the run if these steps do not succeed. For details, see "Default Optional Steps" on page 1072.
- Update your test to change the test object descriptions, expected checkpoint values, and/or the Active Screen images and values. For details, see "Maintaining and Updating Tests" on page 1239.
- Run tests on objects with dynamic descriptions. For details, see Chapter 4, "Managing Test Objects in Object Repositories."
- Set up a batch of tests and run them sequentially, using the QuickTest Test Batch Runner. For details, see "Test Batch Runner" on page 1080.

Global Data Table Parameters

If your test contains a global Data Table parameter, QuickTest runs the test once for each row in the Data Table. If your test contains a Data Table parameter for the current action data sheet, QuickTest runs the action once for each row of data in that action data sheet. You can also specify whether to run the first iteration or all iterations, for the entire test or for a specific action in the test; or to run the iterations for a specified range of data sets. For details on test iterations, see "Run Pane (Test Settings Dialog Box)" on page 1471. For details on Data Table parameters see, "Data Table Parameters" on page 731.

 **Optional Steps**

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

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

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

During a recording session, QuickTest automatically marks steps that open certain dialog boxes as optional. (For a list of these dialog boxes, see "Default Optional Steps" on page 1072.)

You can also manually designate steps as optional. For example, you can add conditional statements or use recovery scenarios to automatically click a button, press ENTER, or enter login information in a step. For details, see "How to Insert Conditional Statements from the Keyword View" on page 901 and "Recovery Scenarios" on page 1523.

Tasks

How to Run Your Test

This task describes the various ways in which you can run a test.

This task includes the following steps:

- "Prerequisites" on page 1067
- "Run an entire test" on page 1067
- "Run to a selected step or action" on page 1068c
- "Run a single action or run a test or action from a selected step" on page 1069
- "Interrupt a run session" on page 1070
- "Results" on page 1070

Prerequisites

- 1 Ensure that any required QuickTest add-ins are loaded. For details, see "How to Start QuickTest" on page 78.
- 2 Open the test you want to run. For details, see "Open Test Dialog Box" on page 401.

Run an entire test

- 1 Open the Run dialog box in one of the following ways:

- Click **Run**.
- Select **Automation > Run**.

The Run dialog box opens.

- 2 In the Run dialog box, choose where to save the run session results, and define any input parameters you want to use, as described in "Run Dialog Box: Results Location Tab (For Tests Stored in the File System)" on page 1073, and "Run Dialog Box: Input Parameters Tab" on page 1078.

Note: When running part of a test within the scope of an action, you need to specify the action's parameters, not the test parameters, in the Input Parameters tab of the Run dialog box. For details, see "Parameters Tab (Action Properties Dialog Box)" on page 563.

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

Run to a selected step or action

- 1 Do one of the following:
 - Select **Debug > Run to Step**.
 - Right-click a step and select **Run to Step**.
 - Right-click an action in the Test Flow pane and select **Run to Action**.
 - 2 In the Run dialog box, choose where to save the run session results, and define any input parameters you want to use, as described in "Run Dialog Box: Results Location Tab (For Tests Stored in the File System)" on page 1073, and "Run Dialog Box: Input Parameters Tab" on page 1078.
-

Note: When running part of a test within the scope of an action, you need to specify the action's parameters, not the test parameters, in the Input Parameters tab of the Run dialog box. For details, see "Parameters Tab (Action Properties Dialog Box)" on page 563.

QuickTest runs from the beginning of the test and pauses at the selected step.

Run a single action or run a test or action from a selected step

- 1 Make sure your application is in a state matching the action or step you want to run.
- 2 Select the action or step where you want to start running the test:
 - In the Test Flow pane, select the action.
 - In the Keyword View, highlight a step or action row.
 - In the Expert View, place your cursor in a line of VBScript.

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

- 3 Do one of the following:
 - Select **Automation > Run from Step**.
 - Select **Automation > Run Current Action**.
 - Right-click the step and select **Run from Step**.
 - Right-click the action in the Test Flow pane and select **Run from Action**.

The Run dialog box opens.

- 4 In the Run dialog box, choose where to save the run session results, and define any input parameters you want to use, as described in "Run Dialog Box: Results Location Tab (For Tests Stored in the File System)" on page 1073, and "Run Dialog Box: Input Parameters Tab" on page 1078.

Note: When running part of a test within the scope of an action, you need to specify the action's parameters, not the test parameters, in the Input Parameters tab of the Run dialog box. For details, see "Parameters Tab (Action Properties Dialog Box)" on page 563.

Interrupt a run session

Do one of the following:



- Click the **Pause** button in the Debug toolbar or select **Debug > Pause**. The run pauses. (To resume running a paused run session, click the **Run** button or select **Automation > Run**.)
- Click the **Stop** button, select **Automation > Stop**, or press the STOP command shortcut key. (To define a Stop command shortcut key, see "Run Pane (Options Dialog Box)" on page 1447.)
- Perform a file operation (for example, open a different test or create a new test).

Results

By default, when the run session ends, the Run Results Viewer opens. For details, see "Run Results Viewer Overview" on page 1086. If you run part of a test, the Run Results summary displays a note indicating that the test was run using the **Run from Step** or **Run Current Action** option.

Note: If you cleared the **View results when run session ends** check box in the Run pane of the Options dialog box, the Run Results Viewer does not open at the end of the run session. For details, see "Run Pane (Options Dialog Box)" on page 1447.

How to Set Optional Steps

This task describes how to set an optional step.

Do one of the following:

- In the Keyword View, right-click the step and select **Optional Step**. The Optional Step icon  is added next to the selected step.

>Welcome: Mercury Tours			
_ userName	Set	"Amy"	Enter "Amy" in the "userName" edit box.
_ password	SetSecure	"425e5cd87021ce00d5476d94a8e44...	Enter the encrypted string "425e5cd87021ce00d5476d94a8e44...
_ Sign-In	Click	10,11	Click the "Sign-In" image.
_ AutoComplete			
_ ? Yes	Click		Click the "Yes" button.
_ Sign-on: Mercury Tours	Sync		Wait for the Web page to synchronize before continuing the run.

- In the Expert View, add **OptionalStep** to the beginning of the VBScript statement. For example:

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

For information on working in the Expert View, see Chapter 27, "Working in the Expert View and Function Library Windows."

Reference

Default Optional Steps

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

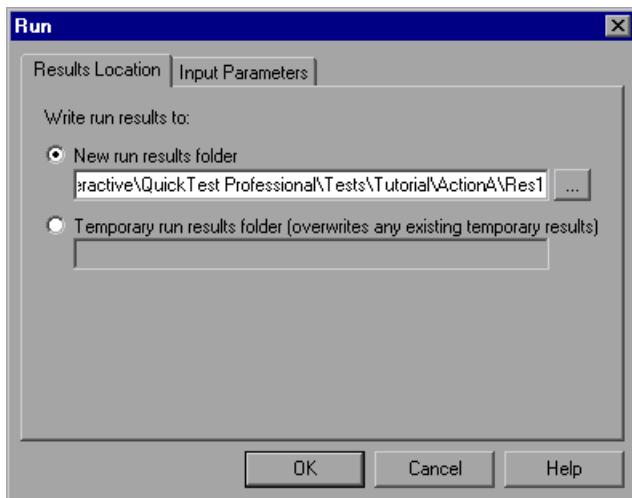
Dialog Box / Message Box Title Bar
AutoComplete
File Download
Internet Explorer
Netscape
Enter Network Password
Error
Security Alert
Security Information
Security Warning
Username and Password Required

For an overview and task details, see "Optional Steps" on page 1066 and "How to Set Optional Steps" on page 1071.

Run Dialog Box: Results Location Tab (For Tests Stored in the File System)

This tab enables you to specify the location in which you want to save your run session results when running tests stored in the file system.

For details on the Run dialog box that opens for Tests stored in Quality Center, see "Run Dialog Box: Results Location Tab (For Tests Stored in Quality Center)" on page 1075.



To access	Start a run session in any mode.
Important information	If you are running a test from a Quality Center project, the Project name , Run name , Test set , and Instance options are displayed instead of the New run results folder option.
Relevant tasks	<ul style="list-style-type: none"> ▶ "How to Run Your Test" on page 1067 ▶ "How to Use Maintenance Run Mode to Update Your Test When Your Application Changes" on page 1248 ▶ "Running to a Step and Debugging from a Step" on page 1209
See also	"Run Dialog Box: Input Parameters Tab" on page 1078

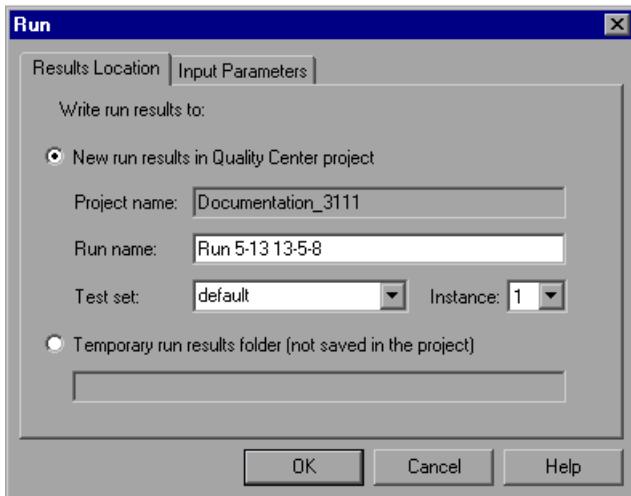
User interface elements are described below:

UI Elements	Description
New run results folder	<p>Displays the default path and folder name in which the results are saved. A new folder is created for each run. By default, the results for a QuickTest test are stored in the test folder.</p> <p>You can accept the default settings, or enter a new path by typing it in the text box or clicking the browse button to locate a different folder. The folder must be new, empty, or contain only QuickTest test files.</p>
Temporary run results folder	<p>Saves the run results in a temporary folder. This option overwrites any results previously saved in this folder.</p> <p>Note:</p> <ul style="list-style-type: none">➤ QuickTest stores temporary results for all tests in %TMP%\TempResults (which is usually <System Drive>\Documents and Settings\<user name>\Local Settings\Temp\ TempResults).➤ The path in the text box of the Temporary run results folder option cannot be changed. Additionally, if you save results to an existing results folder, the contents of the folder are deleted when the run session starts.

Run Dialog Box: Results Location Tab (For Tests Stored in Quality Center)

The tab in this dialog box enables you to specify where to save run results when running tests stored in Quality Center.

For details on the Run dialog box that opens for Tests stored in the File System, see "Run Dialog Box: Results Location Tab (For Tests Stored in Quality Center)" on page 1075.



To access	Start a run session in any mode.
Important information	<p>As QuickTest runs a test, it highlights each step in the Keyword View.</p> <p>When the test stops running, the Run Results Viewer opens unless you have cleared the View results when test run ends check box in the Run pane of the Options dialog box. For details, see Chapter 45, "Global Testing Options."</p> <p>When the test stops running, Uploading is displayed in the status bar. The Run Results Viewer opens when the uploading process is completed.</p> <p>Note: You can report defects to a Quality Center project either automatically as they occur, or manually directly from the QuickTest Run Results Viewer. For details, see "How to Manually Submit Defects to Quality Center" on page 1098 and "How to Automatically Submit Defects to a Quality Center Project" on page 1100.</p>
See also	"Run Dialog Box: Input Parameters Tab" on page 1078

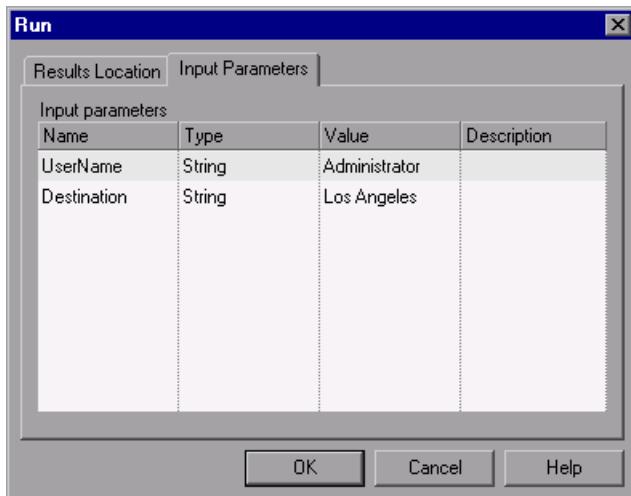
User interface elements are described below:

UI Elements	Description
New run results folder	<ul style="list-style-type: none"> ➤ Project name. Displays the Quality Center project to which you are currently connected. ➤ Run name. The name of the run. You can accept the automatically generated name or enter a different one. ➤ Test set. A group of tests selected to achieve specific testing goals. For example, you can create a test set that tests the user interface of the application or the application's performance under stress. (You define test sets when working in the Quality Center test run mode. For details, see your Quality Center documentation.) ➤ Instance. The instance of the test in the test set. If there is more than one instance, specify the instance of the test for which you want to save the results.

UI Elements	Description
Temporary run results folder	<p>Saves the run results in a temporary folder. This option overwrites any results previously saved in this folder.</p> <p>Note:</p> <ul style="list-style-type: none">▶ The path in the text box of the Temporary run results folder option cannot be changed. Additionally, if you save results to an existing results folder, the contents of the folder are deleted when the run session starts.▶ QuickTest stores temporary results in %TMP%\TempResults (which is usually <System Drive>\Documents and Settings\<user name>\Local Settings\Temp\TempResults)

Run Dialog Box: Input Parameters Tab

This tab enables you to specify the run-time values of input parameters to be used during the run session.



To access	Start a run session in any mode.
Important information	<ul style="list-style-type: none">➤ The Input Parameters tab displays the input parameters that were defined for the test (using the File > Settings > Parameters node).➤ When running part of a test within the scope of an action (using the Automation > Run from Step or Automation > Run Current Action options), you need to specify the action's parameters and not the test parameters.

Relevant tasks	<ul style="list-style-type: none">▶ "How to Run Your Test" on page 1067▶ "How to Use Maintenance Run Mode to Update Your Test When Your Application Changes" on page 1248▶ "Running to a Step and Debugging from a Step" on page 1209
See also	<ul style="list-style-type: none">▶ "Run Dialog Box: Results Location Tab (For Tests Stored in the File System)" on page 1073▶ "Run Dialog Box: Results Location Tab (For Tests Stored in Quality Center)" on page 1075

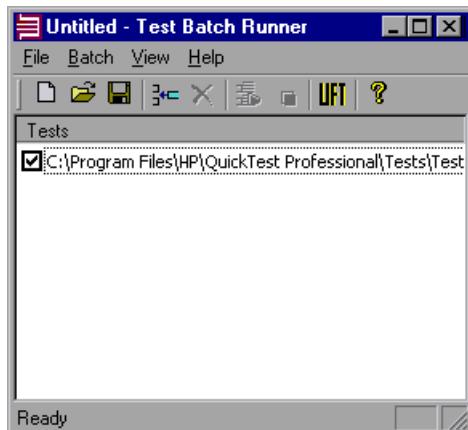
User interface elements are described below:

UI Elements	Description
Input parameters	<p>The input parameters that were defined for the test (using the File > Settings > Parameters node).</p> <p>To set the value of a parameter to be used during the run session:</p> <p>Click in the Value field for the specific parameter and enter the value, or select a value from the list. If you do not enter a value, QuickTest uses the default value from the Test Settings dialog box during the run session.</p>

Test Batch Runner

This tool enables you to run several tests in succession. The results for each test are stored in their default location.

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



To access	From the Start menu, select Programs > HP QuickTest Professional > Tools > Test Batch Runner .
Important information	<ul style="list-style-type: none"> ➤ To enable Test Batch Runner to run tests, you must select Allow other HP products to run tests and components in the Run pane of the Options dialog box. For more information, see "Run Pane (Options Dialog Box)" on page 1447. ➤ Test Batch Runner can be used only with tests located in the file system. If you want to include tests saved in Quality Center in the batch run, you must first save the tests in the file system.

User interface elements are described below:

UI Elements	Description
	<p>Add. Click Add (or select Batch > Add) to open the Open Test dialog box.</p> <p>Select a test you want to include in the test batch list and click Open to add the test to the list.</p> <p>Tip: To insert a test at another point in the list, select the test that is to precede the test you would like to add. When you add the test, it is added above the selected test.</p>
	<p>Remove. Click Remove (or select Batch > Remove) to remove a test from the list.</p> <p>Tip: If you want to include a test in the list, but you do not want the test to be run during the next batch run, clear the check box next to the test name.</p>
	<p>Save. Click Save (or select File > Save) and enter a name for the list. The file extension is .mtb.</p>
	<p>Run. Click Run (or select Batch > Run) when you are ready to run your test batch.</p> <p>If QuickTest is not already open, it opens and the tests run sequence begins. After the batch run is complete, you can view the results for each test in its default run results folder (<test folder>\res#\report). For details, see "Run Results Viewer Overview" on page 1086.</p>
	<p>Stop. Click Stop (or select Batch > Stop) to stop the test batch run.</p>

UI Elements	Description
	<p>Use HP Unified Functional Testing license. Instructs QuickTest to use an HP Unified Functional Testing license during a batch run. Use this button only if at least one of the tests in the batch contains a call to a Service Test test.</p> <p>How does it work?</p> <p>When you click Run (or select Batch > Run), the Test Batch Runner instructs QuickTest to use an HP Unified Functional Testing license. If it cannot use this license type, an error message opens informing you that if you run a test batch that contains a call to a Service Test test, that test will fail, and the batch will stop running.</p> <p>Important: If you click this button and QuickTest begins using an HP Unified Functional Testing license, QuickTest will continue to use this license for the duration of the batch run session. To clear the license, you must close QuickTest.</p>

Troubleshooting and Limitations - Run Sessions

This section describes troubleshooting and limitations for:

QuickTest run sessions

- If you want to run QuickTest in a minimized RDP (remote desktop protocol) session, and you are using an RDP 6.0 or later client, you can enable it by setting a registry value on the remote computer:
 - a If it does not exist, create the **RemoteDesktop_SuppressWhenMinimized** registry value (DWORD type) in one of the following registry paths on the remote computer:
 - HKEY_CURRENT_USER\Software\Microsoft\Terminal Server Client\RemoteDesktop_SuppressWhenMinimized
 - or
 - HKEY_LOCAL_MACHINE\Software\Microsoft\Terminal Server Client\RemoteDesktop_SuppressWhenMinimized
 - b Set the data for this value to 2.

- When running QuickTest on a remote machine using a Remote Desktop Connection session (RDC) or using Citrix, if the computer on which the application is being tested is logged off or locked, the following problems may occur:
 - The test run session may fail
 - Steps that contain keyboard or focus operations may fail
 - The Run Results still image capture and/or the Screen Recorder may display a black screen
 - Steps for which the device level replay is configured to use the mouse (instead of browser events) to run mouse operations may fail. (You set the device level replay using a **Setting.WebPackage("ReplayType")** statement or by setting the **Replay type** option in the Advanced Web Options dialog box.)

Workaround: If you are using Citrix or a Remote Desktop Connection session to run a test, make sure that the computer on which the application is being tested is not logged off or locked.

Learning objects, and recording and running steps

- QuickTest cannot record or run steps if it has limited access to the processes of the application you are testing.

Workarounds:

- Make sure that the application you are testing is started by the same Windows user as QuickTest.
- Make sure that neither you nor the tested application actively prevent QuickTest from accessing the application's processes.
- When the title of a window changes during recording, QuickTest may fail to recognize objects within that window while running the test.

Workaround: Remove the text property from the window's test object description in the Object Repository dialog box.

31

Run Results Viewer

This chapter includes:

Concepts

- Run Results Viewer Overview on page 1086
- Run Results File Location on page 1091

Tasks

- How to Install the Run Results Viewer as a Standalone Application on page 1092
- How to Open Run Results on page 1093
- How to Navigate the Run Results Tree on page 1094
- How to Customize the Run Results Viewer on page 1096
- How to Jump to a Step in QuickTest on page 1097
- How to Manually Submit Defects to Quality Center on page 1098
- How to Automatically Submit Defects to a Quality Center Project on page 1100
- How to Export Run Results on page 1102
- How to Play a Screen Recorder Movie in the HP Micro Player on page 1104
- How to Delete Run Results on page 1105

Reference

- Run Results Viewer User Interface on page 1107
- Run Results Viewer Commands on page 1109
- Run Results Viewer Panes on page 1113
- Run Results Viewer Dialog Boxes on page 1140
- Run Results Deletion Tool on page 1160

Troubleshooting and Limitations - Viewing Run Results on page 1168

Concepts

Run Results Viewer Overview

After running a test, you can view the run results in the HP Run Results Viewer. The Run Results Viewer contains multiple panes, each of which displays specific types of information. The run results tree pane displays a hierarchical representation of the run results. The remaining panes provide details about a selected node or step, the data used for a particular step, optional screen captures or images, optional system information, and so on. For an overview of the various panes, see "Run Results Viewer User Interface" on page 1107.

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

The Run Results Viewer contains a description of the steps performed during the run session.

- For a test that does not contain Data Table input parameters, the Run Results Viewer shows a single test iteration.
- If the test contains Data Table input parameters, and the test settings are configured to run multiple iterations, the Run Results Viewer displays details for each iteration of the test run. The results are grouped by the actions in the test.

You set the test to run for one or all iterations in the Run pane of the Test Settings dialog box. For details, see "Run Pane (Test Settings Dialog Box)" on page 1471.

QuickTest/Service Test Integration

If you run a QuickTest test that contains a call to a Service Test test, or vice versa, you can view the results for all steps performed in the main test and in the called test.

Viewing Partial Results

In addition to viewing the results for a run session after a run is complete (including runs that crash prior to completion), you can view the results *during* a run session by opening the **results.xml** file for that run. This enables you to view partial results (up to the step for which the results are opened). For example, you may want to view the results for a specific iteration before the run continues to the next iteration. One way to do this is to insert a step that opens a message box, as this stops the run until you close the message box.

Example: MsgBox "Open the following file:" & Reporter.ReportPath

Note: To view the partial results in the Run Results Viewer, you need to open the results file from another computer.

Installing the Run Results Viewer

The Run Results Viewer is installed automatically together with QuickTest and Service Test.

You can also install the Run Results Viewer as a standalone application. This enables you to share the results of your tests with business analysts and developers who do not have QuickTest or Service Test installed on their computers.

This section also includes: "Run Results XML File" on page 1088

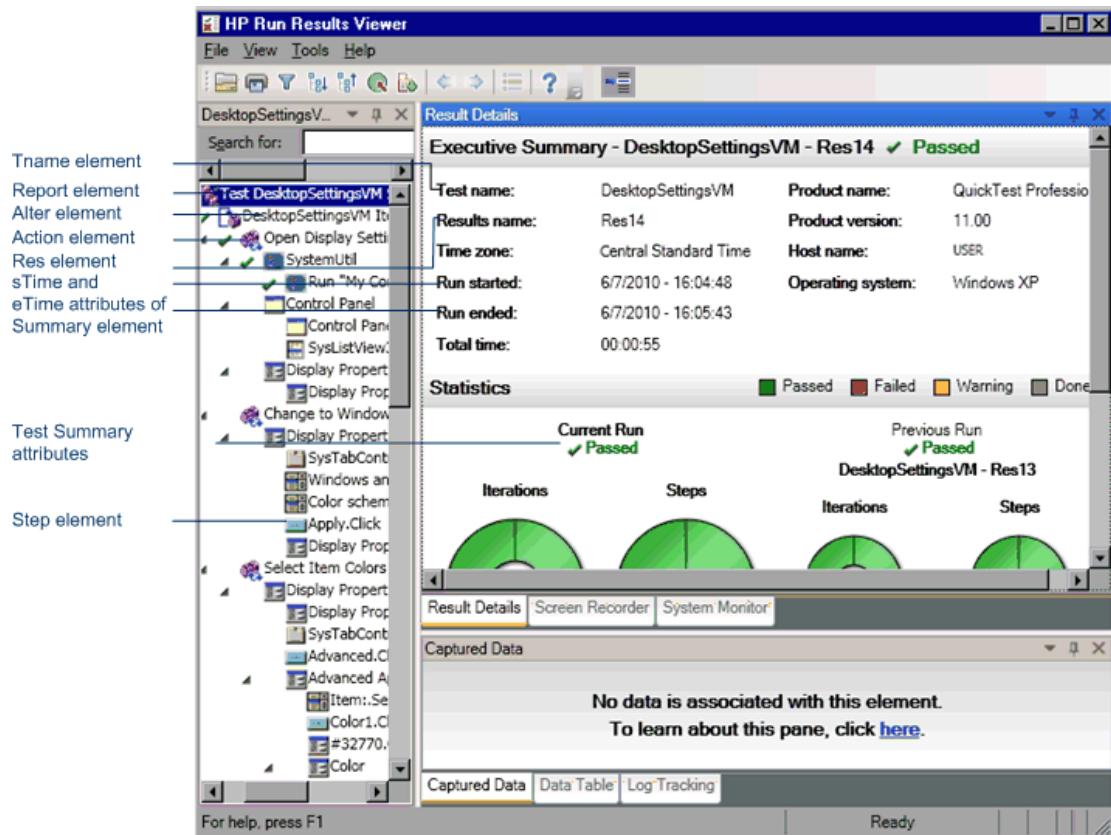


Run Results XML File

The results of each run session are saved in a single **.xml** file (called **results.xml**). This **.xml** file stores information on each of the run result nodes in the display. The information in these nodes is used to dynamically create **.htm** files that are shown in the Result Details pane in the Run Results Viewer.

Each node in the run results tree is an element in the **results.xml** file. In addition, there are different elements that represent different types of information displayed in the run results. You can take run result information from the **.xml** file and use XSL to display the information you require in a customized format (either when printing from within the Run Results Viewer, when displaying run results in your own customized results viewer, or when exporting the run results to an HTML file).

The diagram below for QuickTest, shows the correlation between some of the elements in the **.xml** file and the items they represent in the run results.



XSL provides you with the tools to describe exactly which run result information to display and exactly where and how to display, print or export it. You can also modify the **.css** file referenced by the **.xsl** file, to change the appearance of the report (for example, fonts, colors, and so forth).

For example, in the **results.xml** file, one element tag contains the name of an action, and another element tag contains information on the time at which the run session is performed. Using XSL, you could tell your customized run results viewer that the action name should be displayed in a specific place on the page and in a bold green font, and that the time information should not be displayed at all.

You may find it easier to modify the existing **.xsl** and **.css** files provided with the Run Results Viewer application, instead of creating your own customized files from scratch. The files are located in the **HP\Run Results Viewer\dat** folder, and are named as follows:

- **PShort.xsl**. Specifies the content of the run results report printed, or exported to an HTML file, when you select the **Short** option in the Print or Export to HTML File dialog boxes.
- **PDetails.xsl**. Specifies the content of the run results report printed, or exported to an HTML file, when you select the **Detailed** option in the Print or Export to HTML File dialog boxes.
- **PResults.css**. Specifies the appearance of the run results print preview. This file is referenced by the above **.xsl** files.

For more information on printing run results using a customized **.xsl** file, see "Print Dialog Box (Run Results Viewer)" on page 1151.

For more information on exporting the run results to a file using a customized **.xsl** file, see "Export Run Results Dialog Box (Run Results Viewer)" on page 1141.

For information on the structure of the XML schema, and a description of the elements and attributes you can use to customize the run results reports, see the XML Report Help (**Help > QuickTest Professional Help > QuickTest Advanced References > QuickTest Run Results Schema**).

Run Results File Location

Tests saved in the file system. By default, the results of a QuickTest test that is saved in the file system are stored in the test folder. When you run your test, you can specify a different location to store the results, using the Results Location tab of the Run dialog box. Specifying your own location for the results file can make it easier for you to locate the results file in the file system. For details, see "Run Dialog Box: Results Location Tab (For Tests Stored in the File System)" on page 1073.

Tests saved in Quality Center. Run results are stored in the test folder in Quality Center. You cannot change the location of the run session results. For details, see "Run Dialog Box: Results Location Tab (For Tests Stored in Quality Center)" on page 1075.

Tasks

How to Install the Run Results Viewer as a Standalone Application

The Run Results Viewer is installed by default together with QuickTest and Service Test. This task describes how to install the Run Results Viewer as a standalone application. For example, business analysts and developers that do not have QuickTest or Service Test installed on their computers can install the Run Results Viewer locally as a standalone application. You can then share the results of your tests with them.

1 Install the prerequisite applications, if any.

Insert the QuickTest Professional or Service Test installation DVD into a DVD drive and browse to and double-click

RunResultsViewer\EN\setup.exe. Setup checks your computer for the required prerequisites, and enables you to install them, if needed. Follow the on-screen instructions. After the prerequisites are installed, you may need to restart the computer.

2 Install the HP Run Results Viewer.

Insert the QuickTest Professional or Service Test installation DVD into a DVD drive and browse to and double-click

RunResultsViewer\EN\setup.exe. Follow the on-screen instructions.

The Run Results Viewer is installed and can be opened from **Start > Programs > HP Run Results Viewer > HP Run Results Viewer.**

How to Open Run Results

The following steps describe how to open specific run results in the Run Results Viewer.

- "Open the Run Results Viewer" on page 1093
- "Connect to your Quality Center project - optional" on page 1093
- "View saved results" on page 1093

Open the Run Results Viewer

Open the Run Results Viewer in one of the following ways:

- In QuickTest, select **Automation > Results** or click the **Results** button .
- From the **Start** menu, select **Program Files > HP Run Results Viewer > HP Run Results Viewer**.
- Run a test. By default, the results are displayed in the Run Results Viewer at the end of the run session. (You can change the default setting in the Options dialog box. For details, see "Run Pane (Options Dialog Box)" on page 1447.)

Note: This section describes how to open tests run in QuickTest. For other types of tests, see your testing product's documentation.

Connect to your Quality Center project - optional

If your run results are saved in Quality Center, connect to your Quality Center project before opening the results file. For details, see "HP ALM Connection Dialog Box (Run Results Viewer)" on page 1155.

View saved results

When you open the Run Results Viewer manually, the Open Run Results dialog box opens automatically, enabling you to select results to display.



If the Run Results Viewer opened automatically, click the **Open** button or select **File > Open**. Browse to the relevant results. For details, see "Open Run Results Dialog Box" on page 1149.

How to Navigate the Run Results Tree

This task describes how to collapse or expand a branch in the run results tree to select the level of detail that the tree displays.

When you open run results in the Run Results Viewer for the first time, the tree expands one level at a time. If the child branches under a parent branch were previously expanded, that state is maintained when you expand or collapse the parent branch.

To expand a specific branch, do any of the following:

- Double-click the branch.
- Select the branch and click the arrow to the left of the branch icon.
- Press the plus key (+) on your keyboard number pad.

The tree displays the details for the branch, and the expand sign changes to collapse.

To expand a branch and all branches below it:

- Select the branch and press the asterisk (*) key on your keyboard number pad.
- Right-click a branch and select **Expand All**.

To expand all of the branches in the run results tree, do any of the following:

- Right-click the top level branch and select **Expand All**.
- Select **View > Expand All**.
- Click the **Expand All** button.
- Select the top level of the tree and press the asterisk (*) key on your keyboard number pad.



To collapse a specific node, do any of the following:

- Double-click the node.
- Right-click a node and select **Collapse All**.
- Select it and click the arrow to the left of the node icon.
- Press the minus key (-) on your keyboard number pad.

The node's child nodes disappear from the tree.

To collapse all of the nodes in the tree:

- Right-click the top level branch and select **Collapse All**.
- Select **View > Collapse All**.
- Click the **Collapse All** button.

**To move between previously selected nodes within the run results tree:**

Click the **Go to Previous Node** or **Go to Next Node** buttons.

To find specific steps within the Run Results:

Use the **Search** box (located above the run results tree), for example:



You can search for text, status, and/or types of nodes. For details, see "Run Results Tree Pane and Search Box" on page 1114.

To filter the tree to display only nodes that match certain criteria:

Use the **Filter dialog box (View > Filters)**. For details, see "Filter Dialog Box (Run Results Viewer)" on page 1146.

How to Customize the Run Results Viewer

The following steps describe how to customize the layout of the Run Results Viewer.

- "Move, float, and dock panes" on page 1096
- "Show and hide panes" on page 1096
- "Restore the default layout of the panes" on page 1096

Move, float, and dock panes

You can move the panes to suit your personal preferences by dragging the title bar or tab of the pane you want to move and dropping it in the required location.

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

Floating panes are displayed on top of all other windows. They can be dragged to any position on your screen, even outside of the Run Results Viewer.

Show and hide panes

- **To close panes that are not needed:** Click the X in the top-right corner of a pane.
- **To show panes that are closed:** Select **View > <Name of pane>**.

Restore the default layout of the panes

Select **View > Restore Layout**.

How to Jump to a Step in QuickTest

You can view the step in QuickTest that corresponds to a node in the Run Results tree for any node that has a corresponding step in a QuickTest test.

Note: This feature is disabled for a variety of settings. For details, see "Guidelines for using the Jump to Step in QuickTest command" on page 1097.

To view the step in the test that corresponds to a node:

- 1 Make sure that QuickTest is open to the test whose results are displayed in the Run Results Viewer.
- 2 Select a node in the run results tree.
- 3 Perform one of the following:
 - a Click the **Jump to Step in QuickTest** button from the Run Results toolbar.
 - b Right-click and select **Jump to Step in QuickTest** from the context-sensitive menu.
 - c Select **View > Jump to Step in QuickTest**.
- 4 The QuickTest window is activated and the step is highlighted.



Guidelines for using the Jump to Step in QuickTest command

- The test must be saved before the run session.
- The run results must be from QuickTest Professional 10.00 or later.

This feature is disabled for:

- Any testing document other than a QuickTest test.
- The Action, Iteration, and Test Summary nodes.
- Any step that is part of an action that was run using the **LoadAndRunAction** statement. For more information, see the **Utility** section of the *HP QuickTest Professional Object Model Reference*.
- Any step performed by a recovery scenario.
- Tests that were run in **Fast** mode. For details on this setting, see "Run Pane (Options Dialog Box)" on page 1447.
- Any step run from the Watch or Command tabs of the Debug Viewer.

How to Manually Submit Defects to Quality Center

This task describes how to manually add defects to a Quality Center project.

This task includes the following steps:

- "Prerequisites" on page 1099
- "Connect to a Quality Center project" on page 1099
- "Open the New Defect dialog box" on page 1099
- "Modify the defect information if needed and submit it" on page 1099
- "Results" on page 1099

1 Prerequisites

Ensure that the Quality Center client is installed on your computer. (Enter the Quality Center Server URL in a browser and ensure that the Login screen is displayed.)

2 Connect to a Quality Center project



Select **Tools > ALM/QC Connection** or click the **ALM/QC Connection** button and connect to a Quality Center project. For details, see "HP ALM Connection Dialog Box (Run Results Viewer)" on page 1155.

Note: If you do not connect to a Quality Center project before proceeding to the next step, you are prompted to connect before continuing.

3 Open the New Defect dialog box



Select **Tools > Add Defect** or click the **Add Defect** button to open the New Defect dialog box in the specified Quality Center project. The New Defect dialog box opens.

4 Modify the defect information if needed and submit it

Basic information on the test and any checkpoints (if applicable) is included in the description, but you can modify the defect if needed:

Operating system :	Windows XP
Test path :	C:\Program Files\HP\QuickTest Professional\Tests\Tutorial\Recording on PREDATOR
The CheckPoint 'Flight Details' Failed	

Tip: You can attach movies (.fbr files) to defects in Quality Center. Quality Center users who have the QuickTest Add-in for ALM/QC installed can view the movies from Quality Center.

5 Results

The defect is added to the Quality Center project's defect database.

How to Automatically Submit Defects to a Quality Center Project

This task describes how to set the Run options in QuickTest to automatically submit defects to your Quality Center project for each failed step in your test.

This task includes the following steps:

- "Prerequisites" on page 1100
- "Modify the Run settings in the Options dialog box" on page 1101
- "Results" on page 1102

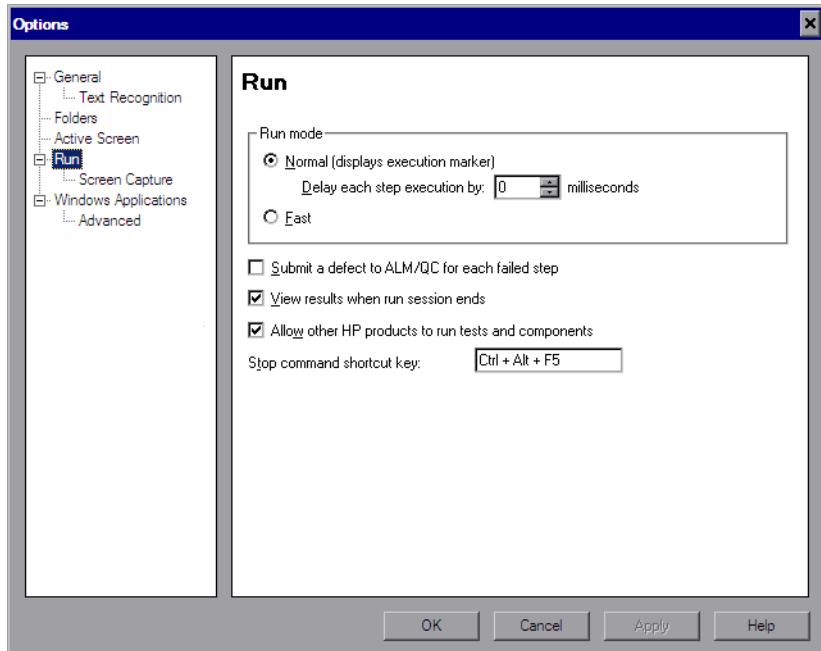
1 Prerequisites

- In QuickTest, make sure you are connected to the relevant Quality Center project prior to the run session (**File > ALM/QC Connection**).
- The run results must be stored in this Quality Center project.

2 Modify the Run settings in the Options dialog box



- a Select **Tools > Options** or click the **Options** button. The Options dialog box opens.
- b Click the **Run** node.



- c Select the **Submit a defect to Quality Center for each failed step** check box.
- d Click **OK** to close the Options dialog box.

3 Results

A sample of the information that is submitted to Quality Center for each defect is shown below:

```
This defect was added automatically by QuickTest Professional

Standard Checkpoint "Flight Details_4" failed

Test name: Recording
Test location: C:\Program Files\HP\QuickTest Professional
\Tests\Tutorial\Recording on BINDER
Action name: Action1

Operating system : Windows XP
Host: BINDER

Additional Information:
Verification type: String Content.
Settings: Exact match - ON; Ignore space - ON; Match case - OFF.
Results: Checked 28 cells;
Succeeded: 27;
Failed: 1
```

How to Export Run Results

This task describes how to export run results to a file. For details on what is included when you export run results, see "Export Run Results Dialog Box (Run Results Viewer)" on page 1141.

This task includes the following steps:

- "Open the results in the Run Results Viewer" on page 1103
- "Specify the export settings" on page 1103
- "Save the file" on page 1103
- "Results" on page 1104

1 Open the results in the Run Results Viewer

For details, see "Open Run Results Dialog Box" on page 1149.

2 Specify the export settings

Select **File > Export To File**. The Export Run Results dialog box opens. For details on the various settings, see "Export Run Results Dialog Box (Run Results Viewer)" on page 1141.

3 Save the file

Click **Export**. The Save As dialog box opens. Specify the file name and path, and select the required file type.

Report type	Save as type
Step details	► HTML (*.htm, *.html) (default) ► PDF (*.pdf) ► DOC (*.doc) (Available if Microsoft Word is installed)
Data Table	Excel (*.xls)
Log Tracking	XML (*.xml)
Screen Recorder	FlashBack (*.fbr)
System Monitor	► Text (*.csv, *.txt) (default) ► Excel (*.xls) ► XML (*.xml) ► HTML (*.htm, *.html) Note: Only the system monitor data is exported, not the graph.

4 Results

When you click **Save**, the file is exported in the specified format to the designated location.

Note: You can view .fbr files in the HP Micro Recorder (as described in "Viewing Screen Recorder Movie Files in the HP Micro Player" on page 1136). You can also attach .fbr files to defects in Quality Center. Quality Center users who have the QuickTest Add-in for ALM/QC installed can view the movies from Quality Center.

How to Play a Screen Recorder Movie in the HP Micro Player

Note: QuickTest must be installed on the computer on which you want to use the HP Micro Player.

1 Perform one of the following:

- Double-click any .fbr file in Windows Explorer.
- Select **Start > Programs > HP QuickTest Professional > Tools > HP Micro Player** and then select **File > Open** in the Micro Player to select any .fbr file.

The movie opens in the HP Micro Player and begins playing.

2 Use the controls at the top of the window to access a particular location in the movie or to modify the volume settings.

How to Delete Run Results

This task describes how to use the Run Results Deletion Tool to remove unwanted or obsolete run results from your system, according to specific criteria that you define. For example, you may want to always delete run results older than a certain date or over a minimum file size. This enables you to free up valuable disk space.

You can use this tool with a Windows-style user interface, or you can use the Windows command line to run the tool in the background (silently) to directly delete results that meet criteria that you specify.

This task includes the following steps:

- "Prerequisites" on page 1105
- "Delete run results using the Run Results Deletion Tool" on page 1105
- "Delete run results using the Windows command line" on page 1106
- "Results" on page 1106

Prerequisites

To delete run results from a Quality Center project, you must first:

- Make sure that you have **Delete Run** permission for this Quality Center project.
- Connect to the Quality Center project. For details see, "HP ALM Connection Dialog Box (Run Results Viewer)" on page 1155.

For details, see "Run Results Deletion Tool" on page 1160.

Delete run results using the Run Results Deletion Tool

For details, see "Run Results Deletion Tool" on page 1160.

Delete run results using the Windows command line

Open a Windows command prompt and enter <QuickTest installation path>\bin\TestResultsDeletionTool.exe. Then enter a space and enter the command line options you want to use. This enables you to specify the criteria for the run results that you want to delete.

For a list of command line options, see "Command Line Options for the Run Results Deletion Tool" on page 1162.

Results

The selected run results are deleted from the file system and/or the Quality Center project.

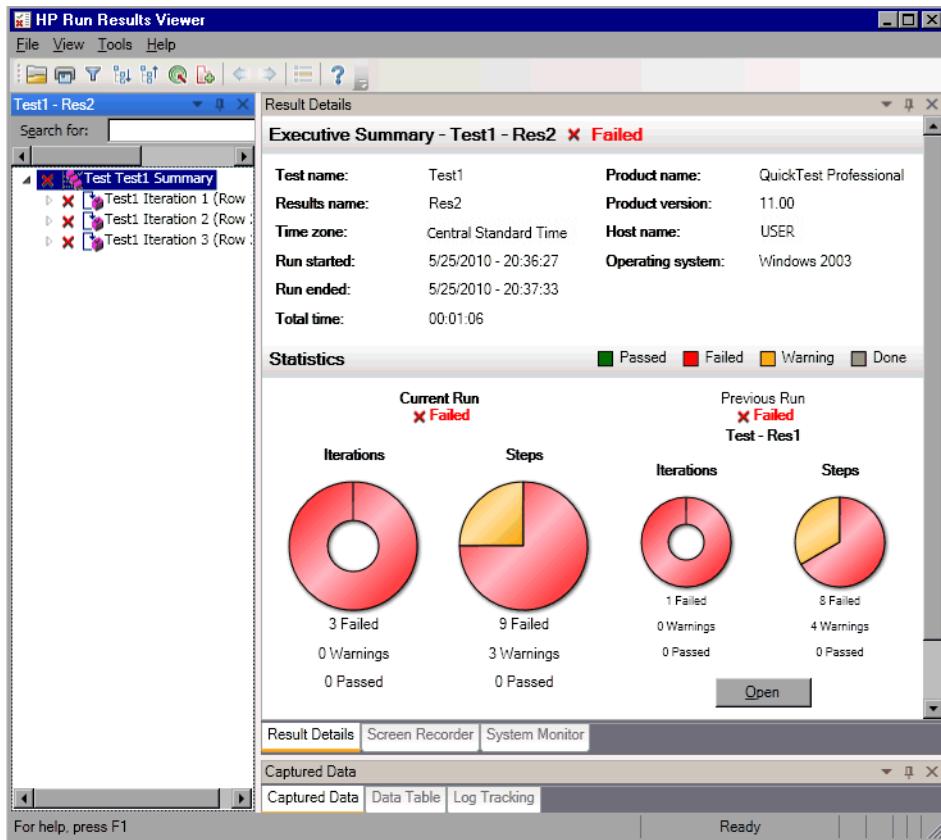
Reference

Run Results Viewer User Interface

This window enables you to view the results of a run session.

QuickTest Test. The following example shows the Executive Summary results of a test with three iterations. Notice that the results for a test are organized by the test's actions.

In the **Statistics** area, you can see how many iterations passed, contained warnings, or failed, and, when previous runs exist, you can compare the current results with the previous results. You can also access the previous run results by clicking the **Open** button.



To access	Open the Run Results Viewer, as described in "How to Open Run Results" on page 1093.
------------------	--

By default, the left pane (dockable) contains the run results tree. The right side of the window contains two rows of additional dockable panes. These user interface elements are described below (unlabeled elements are shown in angle brackets):

UI Elements	Description
<Run Results Viewer menu bar and toolbar>	See "Run Results Viewer Commands" on page 1109.
Executive Summary pane	<ul style="list-style-type: none"> ➤ A high-level results overview report (general information, pass/fail status, statistics, link to previous run results (if any), notes, and so on) displayed in the Result Details pane when the topmost node is selected. For details, see "Executive Summary" on page 1122. ➤ Quality Center information for your test (if the test was run from Quality Center or if a test that is stored in Quality Center is run from QuickTest and you choose to store the results in Quality Center)
Run Results Tree pane	<ul style="list-style-type: none"> ➤ A graphical representation of the results in an expandable tree ➤ A search box ➤ Displays the test steps, specifying exactly where application failures occurred
Result Details pane	Detailed explanations of each step and checkpoint pass or failure, at each stage of the test
Captured Data pane	<ul style="list-style-type: none"> ➤ A still image of the state of your application at a particular step ➤ For Quick Test, additional information, such as a bitmap checkpoint image. <p>For details, see "Captured Data Pane (Run Results Viewer)" on page 1125.</p>
Data Table pane	The data used in all iterations

UI Elements	Description
Screen Recorder pane	A movie clip of the state of your application at a particular step or of the entire test
System Monitor pane	Any system counters that were monitored for your test
Log Tracking pane	Any log messages that were received for your test
<status bar>	<p>Displays:</p> <ul style="list-style-type: none"> ➤ A description of any highlighted menu command (Available only if the menu command is enabled) ➤ The status of the currently selected command ➤ Connection information (when connected to a Quality Center project) ➤ A filter indication icon (when the results are filtered)

Run Results Viewer Commands

The Run Results Viewer menu bar and toolbar contain commands to help you view run session results.

Button	Command	Shortcut Key	Description
	File > Open	CTRL+O	<p>Opens the Open Run Results dialog box, enabling you to open saved run results from the file system or from Quality Center.</p> <p>For details, see "How to Open Run Results" on page 1093.</p>
	File > Print	CTRL+P	<p>Opens the Print dialog box, enabling you to print the results of the run session.</p> <p>For details, see "Print Dialog Box (Run Results Viewer)" on page 1151.</p>

Button	Command	Shortcut Key	Description
--	File > Print Preview	CTRL+F2	Opens the Print Preview dialog box, enabling you to preview the results of the run session prior to printing. For details, see "Print Preview Dialog Box (Run Results Viewer)" on page 1153.
--	File > Export To File	--	Opens the Export Run Results dialog box, enabling you to save various parts of the results as external files. For details, see "Export Run Results Dialog Box (Run Results Viewer)" on page 1141.
--	File > Remove Movie from Results	--	Enables you to remove a stored movie from the results of a test. This reduces the size of the run results file.
--	File > Recent Files	--	Lists the recently viewed files.
--	File > Exit	--	Closes the Run Results Viewer session.
--	View > Run Results Viewer Toolbar	--	Shows or hides the Run Results Viewer toolbar.
--	View > Status Bar	--	Shows or hides the status bar, which indicates: <ul style="list-style-type: none">➤ A hint about the currently selected command➤ The status of the Run Results Viewer➤ The Quality Center server name and project to which the Run Results Viewer is connected➤ Whether the results are currently filtered (displays a Filter icon when a filter is applied)

Button	Command	Shortcut Key	Description
--	View > Result Details	--	Opens the Results Details pane if it is closed or brings it into focus. For details, see "Result Details Pane (Run Results Viewer)" on page 1120.
--	View > Screen Recorder	--	Opens the Screen Recorder pane if it is closed or brings it into focus. For details, see "Screen Recorder Pane (Run Results Viewer)" on page 1134.
--	View > System Monitor	--	Opens the System Monitor pane if it is closed or brings it into focus. For details, see "System Monitor Pane (Run Results Viewer)" on page 1137.
--	View > Captured Data	--	Opens the Captured Data pane if it is closed or brings it into focus. For details, see "Captured Data Pane (Run Results Viewer)" on page 1125.
--	View > Data Table	--	Opens the Data Table pane if it is closed or brings it into focus. For details, see "Data Table Pane (Run Results Viewer)" on page 1131.
--	View > Log Tracking	--	Opens the Log Tracking pane if it is closed or brings it into focus. For details, see "Log Tracking Pane (Run Results Viewer)" on page 1132.
--	View > Restore Layout	--	Restores the default layout of the Run Results Viewer.
	View > Filters	CTRL+T	Opens the Filters dialog box, enabling you to filter the information displayed. For more information, see "Filter Dialog Box (Run Results Viewer)" on page 1146.

Button	Command	Shortcut Key	Description
	View > Collapse All	--	<p>Collapses all of the branches in the run results tree.</p> <p>Also available as a context-menu option to collapse all of the nodes under the selected node.</p>
	View > Collapse All	--	<p>Collapses all of the branches in the run results tree.</p> <p>Also available as a context-menu option to collapse all of the nodes under the selected node.</p>
	View > Go to Previous Node	--	Moves the cursor to the previously selected node in the tree.
	View > Go to Next Node	--	Moves the cursor to the node you selected in the tree prior to clicking the Go to Previous Node button.
	View > Jump to Step in Test	CTRL+J	<p>Activates the QuickTest window and highlights the step in the test corresponding to the selected node in the Run Results tree.</p> <p>For details, see "How to Jump to a Step in QuickTest" on page 1097.</p> <p>Note: Disabled for the Action, Iteration, and Summary nodes.</p>
	Tools > Add Defect	--	Enables you to add a defect to your Quality Center project. If you are not currently connected to Quality Center, opens the HP ALM Connection dialog box. For more information, see "How to Manually Submit Defects to Quality Center" on page 1098.

Button	Command	Shortcut Key	Description
	Tools > ALM/QC Connection	--	Opens the HP ALM Connection dialog box, enabling you to connect to a Quality Center project. For more information, see "HP ALM Connection Dialog Box (Run Results Viewer)" on page 1155.
	Go to Previous Node	BACKSPACE	Moves the cursor to the previously selected node in the run results tree. For more information, see "How to Navigate the Run Results Tree" on page 1094.
	Go to Next Node	ALT+RIGHT ARROW	Moves the cursor to the node you selected in the run results tree prior to clicking the Go to Previous Node button. For more information, see "How to Navigate the Run Results Tree" on page 1094.
	Help > Help Topics	--	Opens the <i>HP Run Results Viewer Help</i> .
--	Help > About Run Results Viewer	--	Displays version information about the HP Run Results Viewer.

Run Results Viewer Panes

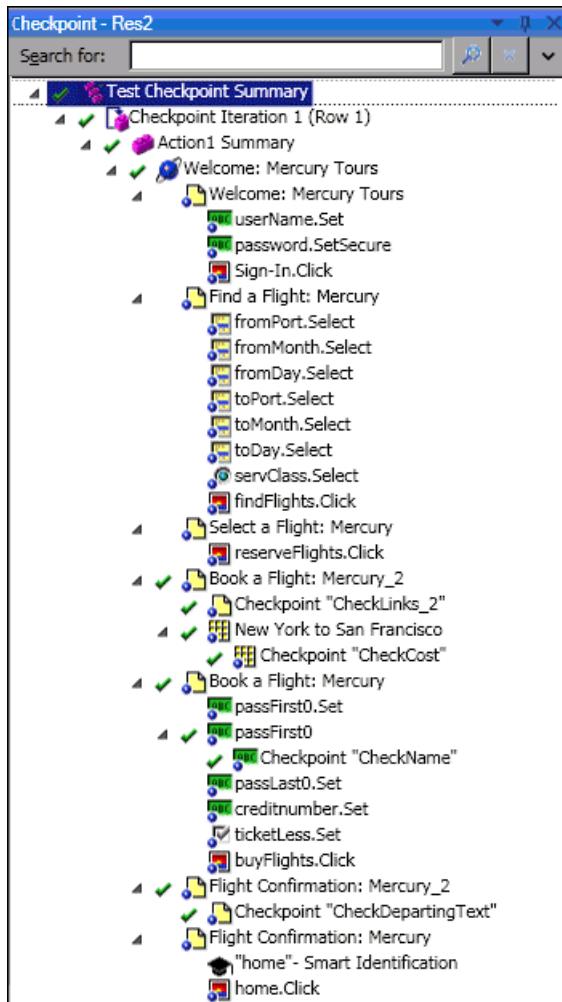
This section includes:

- "Run Results Tree Pane and Search Box" on page 1114
- "Result Details Pane (Run Results Viewer)" on page 1120
- "Captured Data Pane (Run Results Viewer)" on page 1125
- "Data Table Pane (Run Results Viewer)" on page 1131
- "Log Tracking Pane (Run Results Viewer)" on page 1132
- "Screen Recorder Pane (Run Results Viewer)" on page 1134
- "System Monitor Pane (Run Results Viewer)" on page 1137

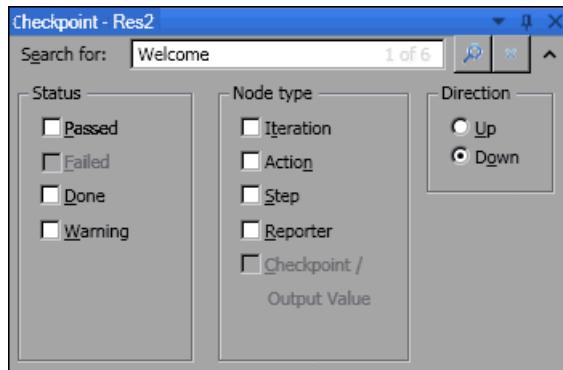
Run Results Tree Pane and Search Box

This pane displays the **run results tree**—a hierarchical representation of the run results. The **Search** box is located above the tree.

The following image shows an example of the run results tree with a collapsed **Search** box.



The following image shows an example of an expanded **Search** box. Six instances of the searched for text, Welcome, were found.



To access	Open the Run Results Viewer, as described in "How to Open Run Results" on page 1093. The Run Results Tree pane is displayed by default on the left side of the Run Results Viewer. It cannot be hidden. The Search box is located directly above the tree and can be expanded by clicking the Expand button.
Important information	Click a node in the tree to view its details in the Result Details pane. Other panes also display information for the test or highlighted step, if available. You can collapse or expand a node in the run results tree to change the level of detail that the tree displays. You can also use the Filter commands to control what is displayed in the run results tree. For details, see "Filter Dialog Box (Run Results Viewer)" on page 1146.

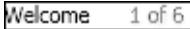
Run Results Tree

Some of the Run Results Tree icons are described below:

UI Elements	Description
	<p>Indicates a step that succeeded.</p> <p>Note: If a test does not contain checkpoints, no icon is displayed.</p>
	<p>Indicates a step that failed. Note that this causes all parent steps (up to the root action or test) to fail as well.</p>
	<p>Indicates an information step. This does not affect the pass/fail status of the step.</p>
	<p>Indicates a warning, meaning that the step did not succeed, but it did not cause the action or test to fail.</p>
	<p>Indicates a step that failed unexpectedly, such as when an object is not found for a checkpoint.</p>
	<p>Indicates that the Smart Identification mechanism successfully found the object.</p>
	<p>Indicates that a recovery scenario was activated.</p>
	<p>Indicates that the run session was stopped before it ended.</p>
	<p>Square brackets around a test object name indicate that the test object was created dynamically during the run session. A dynamic test object is created either using programmatic descriptions or by using an object returned by a ChildObjects method, and is not saved in the object repository.</p>
	<p>Displays the Maintenance Mode Update Result, which is a table that describes the Action taken by Maintenance Run Wizard on a failed step and its Details. Displayed only for tests run in Maintenance Run Mode. For more information on Maintenance Run Mode, see "Maintenance Run Mode" on page 1242.</p>

Search Box

The Search box user interface elements are described below:

UI Elements	Description
	<p>Text box in which you can optionally enter text for which to search.</p> <p>If the specified text is found in one or more tree nodes, the text area indicates this, as shown below:</p>  <p>In this example, 1 of 6 indicates that there are six nodes displaying the text, Welcome, and the first matching node is highlighted in the tree.</p>
	<p>Search. Finds the next instance that matches the criteria you specified in the Search box. Click this button to jump to each node that matches your search criteria.</p>
	<p>Cancel. Clears the Search for text box.</p>
	<p>Expand or Collapse. Shows or hides the lower part of the Search box.</p>

UI Elements	Description
Status	<p>The status to search for. (Optional)</p> <p>Possible values:</p> <ul style="list-style-type: none"> ➤ Passed. Searches for steps that passed and match your other selection criteria. ➤ Failed. Searches for steps that failed and match your other selection criteria. ➤ Done. Searches for steps with the status Done (steps that were performed successfully but did not receive a pass, fail, or warning status) that match your other selection criteria. ➤ Warning. Searches for steps with the status Warning (steps that did not pass, but did not cause the test to fail) that match your other selection criteria. <p>Note: If the tree does not contain any steps that match a particular status, that option is grayed out in the Search box.</p>
Node type	<p>The type of node to search for (together with your other search criteria). (Optional)</p> <p>Possible values:</p> <ul style="list-style-type: none"> ➤ Iteration. Searches for Iteration nodes that match your other selection criteria. ➤ Action. Searches for action nodes that match your other selection criteria. ➤ Step. Searches for steps that match your other selection criteria. ➤ Reporter. Searches for Reporter steps that match your other selection criteria. <p>Note: This is not relevant for Reporter.ReportNote steps, which are displayed in the Executive Summary page and not in the run results tree.</p> <ul style="list-style-type: none"> ➤ Checkpoint/Output Value. Searches for checkpoint and output value steps that match your other selection criteria. <p>Note: If the tree does not contain a particular node type, that option is grayed out in the Search box.</p>

UI Elements	Description
Direction	The direction to search in the tree. Possible values: ➤ Up ➤ Down

Result Details Pane (Run Results Viewer)

This pane displays the details for an individual iteration, an action, or a step that is currently selected in the run results tree.

Example of Executive Summary:

The Executive Summary is displayed when the topmost node in the run results tree is selected.

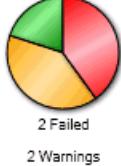
Result Details

Executive Summary - DesktopSettingsVM - Res3 ✓ Passed

Test name:	DesktopSettingsVM	Product name:	QuickTest Professional
Results name:	Res3	Product version:	11.00
Time zone:	Eastern Standard Time	Host name:	VRND2
Run started:	1/20/2010 - 12:00:20	Operating system:	Windows XP
Run ended:	1/20/2010 - 12:01:34	Server name:	http://lab:8080/qcbin
Total time:	00:01:14	Project name:	DEFAULT.FT_11_01

Statistics

Current Run ✓ Passed Previous Run DesktopSettingsVM - Res2 ✗ Failed

Iterations	Steps	Iterations	Steps
			
0 Failed 0 Warnings 1 Passed	0 Failed 0 Warnings 1 Passed	1 Failed 0 Warnings 0 Passed	2 Failed 2 Warnings 1 Passed

Notes

This test was run using a wireless connection.

Example of Result Details:

Result details are displayed when any node (other than the topmost node) is selected in the run results tree.



To access	<p>Open the Run Results Viewer, as described in "How to Open Run Results" on page 1093, and do the following:</p> <p>1 Select a node in the run results tree:</p> <ul style="list-style-type: none">➤ To open the Executive Summary page, select the topmost node in the tree.➤ To open the Result Details for a step, select the relevant node in the tree. <p>2 Select the Result Details tab. (This assumes that the default layout is displayed.)</p> <p>Tip: If the Result Details pane is hidden, select View > Result Details to show it.</p>
------------------	---

Important information	<p>By default, when the Run Results Viewer opens after a run session, an Executive Summary is displayed in the Result Details pane. This summary displays run session information about the test. It also includes run Statistics and Notes (if any were included).</p> <p>For any other node, the details in the Result Details pane are specific for the step selected in the run results tree. For example, the details may include input or output parameters, or may indicate that the session ran in Update Run Mode.</p>
------------------------------	--

User interface elements are described below (unlabeled elements are shown in angle brackets):

Executive Summary Page

UI Elements	Description
Executive Summary	<p>Includes:</p> <ul style="list-style-type: none"> ▶ the test name and result details, and configuration details, if any ▶ time-related information for the run ▶ the product from which the test was run ▶ Quality Center server and project, if QuickTest was connected to a Quality Center project during the run <p>Note: If a test that is stored in Quality Center is run from QuickTest, but you choose to store the results in a temporary location, the Test set and Test instance fields are not displayed in the results.</p> <ul style="list-style-type: none"> ▶ input and output parameters, if any ▶ additional information (for example, if the test ran in update mode)
Statistics	<p>Provides graphical, status-related statistics for the current run and the previous run (if any). If the test was run previously, you can click Open to open the previous run results in a new Run Results Viewer window.</p>

UI Elements	Description
Notes	Displays any test-related notes, if any were added to the test using the Reporter.ReportNote Utility statement. For details, see the Utility section of the <i>HP QuickTest Professional Object Model Reference</i> .
Parameters	Displays any test input and/or output parameters. For details, see "Parameterized Values in the Run Results" on page 1188.

Result Details for Step

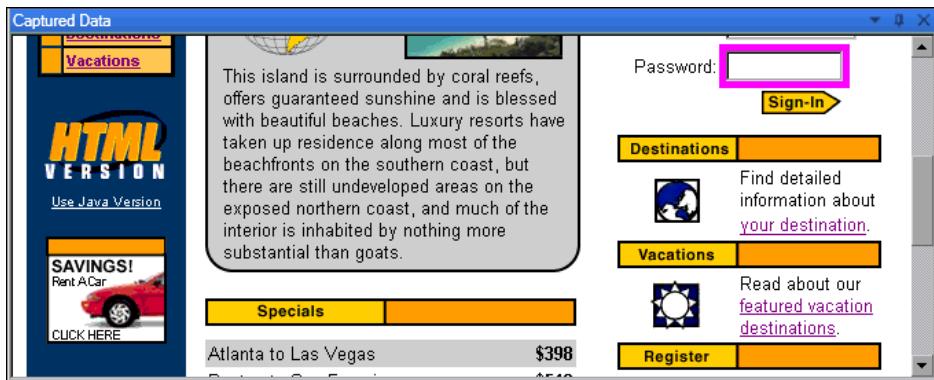
UI Elements	Description
<step name>	The name of the step.
<step status>	<p>The status of the step. Possible values:</p> <ul style="list-style-type: none"> ➤ Done. Relevant for iterations, actions, and steps that ran successfully, but do not contain checkpoints. ➤ Failed. Relevant for iterations, actions, and steps that contain checkpoints. ➤ Passed. Relevant for iterations, actions, and steps that contain checkpoints. ➤ Warning. Relevant for steps that were not successful, but did not cause the test to stop running. <p>Note: A test, iteration, or action containing a step marked Warning may still be labeled Passed or Done.</p>

UI Elements	Description																															
<step details>	<p>Details about the step, such as the object on which the step was performed, the timestamp, the step results, and so on. The information in this area changes according to the type of step.</p> <p>Example 1: Step performed on a Page test object</p> <table border="1" data-bbox="505 376 1023 480"> <thead> <tr> <th data-bbox="573 385 645 402">Object</th><th data-bbox="707 385 851 402">Details Result</th><th data-bbox="912 385 970 402">Time</th></tr> </thead> <tbody> <tr> <td data-bbox="544 419 688 471">Sign-on: Mercury Tours</td><td data-bbox="721 433 764 451">Page</td><td data-bbox="783 433 826 451">Done</td><td data-bbox="845 433 1009 451">1/21/2010 - 17:55:41</td></tr> </tbody> </table> <p>Example 2: Output value step</p> <table border="1" data-bbox="505 537 1023 883"> <thead> <tr> <th colspan="4" data-bbox="512 546 613 589">userName Results</th></tr> <tr> <th data-bbox="512 606 584 649">Property Name</th><th data-bbox="591 606 707 649">Captured Value</th><th data-bbox="714 606 779 649">Type</th><th data-bbox="800 606 865 623">Name</th></tr> </thead> <tbody> <tr> <td data-bbox="512 675 584 692">html tag</td><td data-bbox="591 675 663 692">INPUT</td><td data-bbox="714 675 807 718">Repository Parameter</td><td data-bbox="829 675 1009 692">userName_html_tag_out</td></tr> <tr> <td data-bbox="512 727 584 744">name</td><td data-bbox="591 727 663 744">userName</td><td data-bbox="714 727 807 770">Repository Parameter</td><td data-bbox="829 727 995 744">userName_name_out</td></tr> <tr> <td data-bbox="512 779 584 796">type</td><td data-bbox="591 779 635 796">text</td><td data-bbox="714 779 807 822">Repository Parameter</td><td data-bbox="829 779 980 796">userName_type_out</td></tr> <tr> <td data-bbox="512 831 584 848">value</td><td data-bbox="591 831 635 848">yael</td><td data-bbox="714 831 807 874">Repository Parameter</td><td data-bbox="829 831 995 848">userName_value_out</td></tr> </tbody> </table>	Object	Details Result	Time	Sign-on: Mercury Tours	Page	Done	1/21/2010 - 17:55:41	userName Results				Property Name	Captured Value	Type	Name	html tag	INPUT	Repository Parameter	userName_html_tag_out	name	userName	Repository Parameter	userName_name_out	type	text	Repository Parameter	userName_type_out	value	yael	Repository Parameter	userName_value_out
Object	Details Result	Time																														
Sign-on: Mercury Tours	Page	Done	1/21/2010 - 17:55:41																													
userName Results																																
Property Name	Captured Value	Type	Name																													
html tag	INPUT	Repository Parameter	userName_html_tag_out																													
name	userName	Repository Parameter	userName_name_out																													
type	text	Repository Parameter	userName_type_out																													
value	yael	Repository Parameter	userName_value_out																													

Captured Data Pane (Run Results Viewer)

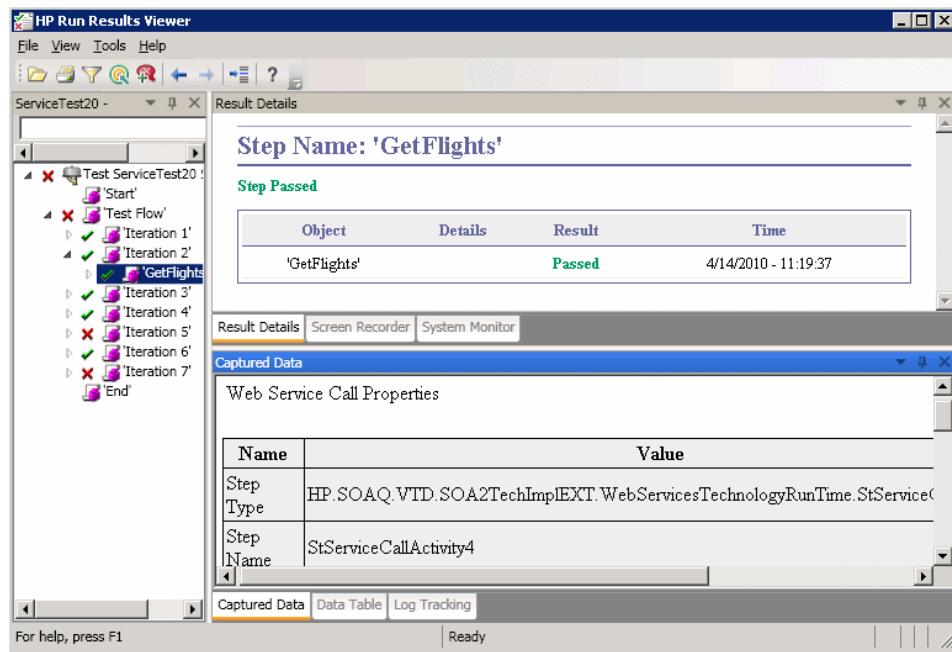
This pane may display a still image of your application for the highlighted step, a bitmap checkpoint image, or other data, for example, if the step was performed for a Service Test step.

The following image shows an example of the Captured Data pane with a still image of an application taken during a run session.



Chapter 31 • Run Results Viewer

The following image shows the Captured Data pane with Web Service Call properties.



To access	Open the Run Results Viewer, as described in "Run Results Viewer User Interface" on page 1107. Select View > Captured Data or click the Captured Data tab.
Important information	<ul style="list-style-type: none"> ➤ Screen captures for QuickTest steps. By default, QuickTest saves a still image of your application for failed steps. When you select a failed step in the run results tree and select the Captured Data pane, the pane displays a screen capture of your application corresponding to the highlighted step in the run results tree. If the highlighted step does not contain an error, no screen capture is displayed. You instruct QuickTest to include still images of your application in the run results by setting the Save still image captures to results option in the Run > Screen Capture pane of the Options dialog box described on page 1447. ➤ Programmatically adding information to the results: You can also programmatically add an image to the Captured Data pane using the ReportEvent method of the Reporter utility object. For details, see the Utility section of the <i>HP QuickTest Professional Object Model Reference</i>. ➤ Captured Data pane in Service Test step. The Captured Data pane contains most of the information that is relevant for Service Test steps. For details on the content that can be displayed in this pane, see "Captured Data Pane Contents in Service Test Steps" on page 1128.
Relevant tasks	"How to Navigate the Run Results Tree" on page 1094

Captured Data Pane Contents in Service Test Steps

The contents of the Captured Data pane differs depending on the level you select in the run results tree.

- **Start, End.** General information about the Start and End activities.
- **Parent step.** Information about the loop to which the test steps belong, such as **Main Loop**.
- **Iteration.** Information about the iteration to which the activities belong.
- **Activity.** Captured data from the activity.
 - For service type activities, this level shows the Request and Response data for the operation or method.
 - For the **Report Message** activity, this level displays the custom message defined in the activity's properties.
- **Checkpoints.** Data about the checkpoints such as the Expected and Actual values, the method of evaluation (Equals, Does Not Equal, and so on) and status.

Request and Response

The **Activity** level captured data contains a table showing the request and response messages.

The following example shows the Request and Response data for the **GetFlights** operation from the sample Web Service.

The screenshot shows the 'Captured Data' window with the title 'Web Service Call HTTP Snapshot'. It is divided into two main sections: 'Request' and 'Response'.

Request:

- HTTP Header:**

```
SOAPAction: HP.SOAQ.Sample&App/IHPFlight_Ser
Content-Type: text/xml; charset=utf-8
Host: vndoc03:24240
Content-Length: 220
Expect: 100-continue
```
- SOAP:**

```
<Envelope xmlns="http://schemas.xmlsoap.org/soap/>
  <Header />
  <Body>
    <GetFlights xmlns="HP.SOAQ.SampleApp">
      <DepartureCity>Frankfurt</DepartureCity>
      <ArrivalCity>Portland</ArrivalCity>
    </GetFlights>
  </Body>
```

Response:

- HTTP Header:**

```
Connection: close
Content-Length: 14816
Content-Type: text/xml; charset=utf-8
Date: Wed, 14 Apr 2010 08:19:37 GMT
Server: Microsoft-HTTPAPI/1.0
```
- SOAP:**

```
<s:body>
  <GetFlightsResponse xmlns="HP.SOAQ.Sample.
    <GetFlightsResult xmlns:i="http://www.w3.org/2001/XMLSchema-instance">
      <Flight>
        <Airlines>AF</Airlines>
        <ArrivalCity>Portland</ArrivalCity>
        <ArrivalTime>10:02 AM</ArrivalTime>
        <DepartureCity>Frankfurt</DepartureC
```

At the bottom of the window, there are three tabs: 'Captured Data' (which is selected), 'Data Table', and 'Log Tracking'.

Data Driven and Parameterized Properties

If you parameterized or applied data driving to an activity's properties, the viewer lists the actual values used during the test run, per iteration.

For built-in activities, the **Step Properties** table shows the values used during the test run.

Name	Value
Type	HP.SOAQ.VTD.SO2TechImplEXT.VTDBasicActivity
Name	ConcatenateStringsActivity4
Prefix	'Hello '
Suffix	'World'
Result	'Hello World'
DisplayName	'ConcatenateStringsActivity4'

For service requests, you can observe the actual values in the **Request/Response** table.

Request	
HTTP Header	
SOAPAction:	HP.SOAQ.SampleApp/IHPFlight_Ser
Content-Type:	text/xml; charset=utf-8
Host:	vmdoc03:24240
Content-Length:	220
Expect:	100-continue
SOAP	
<Envelope xmlns="http://schemas.xmlsoap.org/soap/>	
<Header />	
<Body>	
<GetFlights xmlns="HP.SOAQ.SampleApp">	
<DepartureCity>Frankfurt</DepartureCity>	
<ArrivalCity>Portland</ArrivalCity>	
</GetFlights>	
</Body>	

Data Table Pane (Run Results Viewer)

This pane contains the runtime version of the data table associated with your QuickTest test or your HP ALM configuration. It displays the values used to run a test or configuration that contains Data Table parameters, as well as any output values retrieved from a QuickTest test or an HP ALM configuration during a run session.

If you select a node in the run results tree that represents a step using data table values, the relevant row is highlighted in this pane. In the following example, the data table contains parameterized flight departure and arrival values.

Data Table		
	Departure	Arrival
1	Frankfurt	Amsterdam
2	New York	Boston
3	London	Seattle

Action1

To access	Open the Run Results Viewer, as described in "How to Open Run Results" on page 1093. Select View > Data Table or click the Data Table tab.
Important information	For QuickTest tests. This pane may display one data table or several data tables divided by tabs, for example, if your test uses data table parameters from the global sheet and individual action sheets.
See also	For details on the run-time data table, see Chapter 38, "Data Table Pane."

User interface elements are described below (unlabeled elements are shown in angle brackets):

UI Elements	Description
<Row>	Represents the set of values that QuickTest submitted for the parameterized arguments during a single iteration of the action (QuickTest tests only), test, or configuration.
<Column>	Represents the list of values for a single parameterized argument. The column header is the parameter name.



Log Tracking Pane (Run Results Viewer)

This pane displays a complete list of log messages that QuickTest received from your application during the run session.

In addition to viewing the log messages, when you select a message, you can see its details in the Result Details pane (described on page 1120).

Tip: You can print or export the log tracking details to a file to show to a developer, or you can provide the developer with the standalone Run Results Viewer installation so that the developer can view and analyze the results directly on his or her own computer. For details, see "How to Install the Run Results Viewer as a Standalone Application" on page 1092.

ID	Timestamp	Level	Logger	Thread	Message
15	2010-01-14 11:18:11.108	INFO	Log4NetPiano.Log4PianoForm	1	Message: INFO
16	2010-01-14 11:18:12.452	INFO	Log4NetPiano.Log4PianoForm	1	Message: INFO
17	2010-01-14 11:18:13.795	INFO	Log4NetPiano.Log4PianoForm	1	Message: INFO
18	2010-01-14 11:18:15.155	INFO	Log4NetPiano.Log4PianoForm	1	Message: INFO
19	2010-01-14 11:18:16.498	INFO	Log4NetPiano.Log4PianoForm	1	Message: INFO
20	2010-01-14 11:18:17.342	WARN	Log4NetPiano.Log4PianoForm	1	Message: WARN
21	2010-01-14 11:18:18.186	DEBUG	Log4NetPiano.Log4PianoForm	1	Message: DEBUG
22	2010-01-14 11:18:19.608	DEBUG	Log4NetPiano.Log4PianoForm	1	Message: DEBUG
23	2010-01-14 11:18:20.967	DEBUG	Log4NetPiano.Log4PianoForm	1	Message: DEBUG
24	2010-01-14 11:18:22.327	DEBUG	Log4NetPiano.Log4PianoForm	1	Message: DEBUG
25	2010-01-14 11:18:23.670	DEBUG	Log4NetPiano.Log4PianoForm	1	Message: DEBUG
26	2010-01-14 11:18:25.014	DEBUG	Log4NetPiano.Log4PianoForm	1	Message: DEBUG
27	2010-01-14 11:18:26.358	DEBUG	Log4NetPiano.Log4PianoForm	1	Message: DEBUG
28	2010-01-14 11:18:27.702	DEBUG	Log4NetPiano.Log4PianoForm	1	Message: DEBUG

To access	Open the Run Results Viewer, as described in "How to Open Run Results" on page 1093. Select View > Log Tracking.
Important information	Japanese characters - known Log4Net issue. If a log message contains Japanese characters, these characters are displayed as question marks (?) in the Message column of the Run Results Viewer's Log Tracking pane. This is due to a known bug in log4net.Layout.XmlLayoutSchemaLog4j - version 1.2.10. For details, see https://issues.apache.org/jira/browse/LOG4NET-229 .

See also	<ul style="list-style-type: none">▶ "Find Dialog Box (Log Tracking Pane - Run Results Viewer)" on page 1144▶ "Log Tracking" on page 1461
-----------------	---

User interface elements are described below:

UI Elements	Description
Find	Opens the Find dialog box, enabling you to search the log messages by message, level, and character case.
ID	The message number.
Timestamp	The date and time (in milliseconds).
Level	The severity level for the log message. Possible level values: <ul style="list-style-type: none">▶ TRACE▶ DEBUG▶ INFO▶ WARN▶ ERROR▶ FATAL
Logger	The name of the logger.
Thread	The thread that initiated the log request.
Message	The text of the log message.

Screen Recorder Pane (Run Results Viewer)

This pane enables you to view a movie of a run session. You can view the entire movie, or you can view a frame for particular segment (by selecting a node in the run results tree or by clicking on a specific point in the slider).



To access	Open the Run Results Viewer, as described in "How to Open Run Results" on page 1093. Select View > Screen Recorder or click the Screen Recorder tab.
-----------	--

Important information	<ul style="list-style-type: none"> ➤ Use of multiple monitors: The Screen Recorder records a movie of the operations performed on your primary monitor. Therefore, if you are working with multiple monitors, make sure that your application is fully visible on your primary monitor when recording or running a test. ➤ Prevent QuickTest from obscuring your application. The Screen Recorder saves a movie of your entire desktop. You can prevent the QuickTest window from partially obscuring your application while capturing the movie by minimizing QuickTest during the run session. For information on how to minimize QuickTest during run sessions, see "QuickTest Window Layout Customization in Different Modes" on page 1280. ➤ View full screen. You can double-click the Screen Recorder pane to display the Screen Recorder in full-screen mode and hide the run results tree. Double-clicking again restores the Screen Recorder to its previous size and displays the run results tree. When the Screen Recorder is expanded, the playback controls at the top of the Screen Recorder automatically hide after approximately three seconds with no mouse activity, or when you click anywhere on the Screen Recorder. They reappear when you move the mouse again.
------------------------------	---

User interface elements are described below:

UI Elements	Description
	First Frame. Click to view the first frame in the movie.
	Play/Pause. Click to play or pause the movie. When you click Pause , the relevant tree node is highlighted.
	Stop. Click to stop the movie.
	Last Frame. Click to view the last frame in the movie.

UI Elements	Description
	Slider. Drag the bar to view a particular frame in the movie. When you click on the bar, the relevant node is highlighted in the tree. When you play the movie, the slider automatically moves to the position of the currently displayed frame.
	Volume Control. Drag the bar to the right or left to control the movie sound.
	Mute. Click to turn off the movie sound, if any.

Setting Movie Recording Options

You can customize the whether QuickTest captures movies and the criteria used to save them, using the **Save movies to results** option in the **Run > Screen Capture** pane of the Options dialog box. For details, see "Run Pane (Options Dialog Box)" on page 1447.

Exporting Movies of Your Run Session

You can export a captured Screen Recorder movie as an **.fbr** file. You can view **.fbr** files in the HP Micro Recorder (as described in "Viewing Screen Recorder Movie Files in the HP Micro Player" on page 1136).

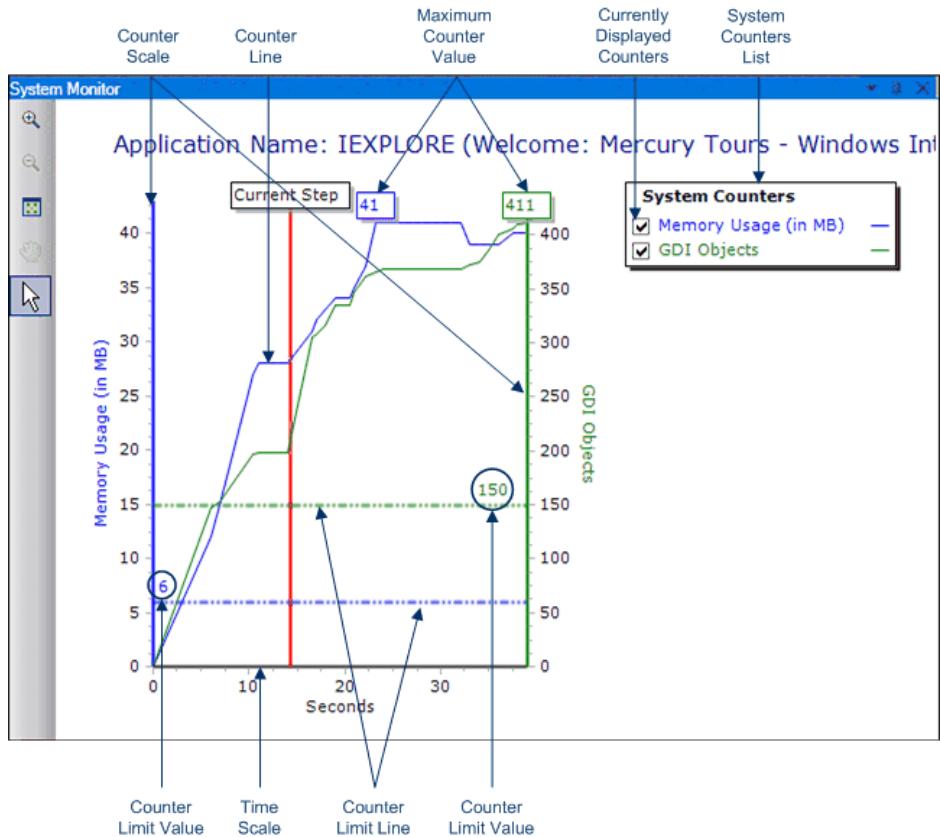
You can also attach **.fbr** files to defects in Quality Center. Quality Center users who have the QuickTest Add-in for ALM/QC installed can view the movies from Quality Center.

Viewing Screen Recorder Movie Files in the HP Micro Player

When you capture a movie of your run session using the Screen Recorder, the movie is saved as an **.fbr** file in your run results folder. You can also view these **.fbr** files without opening the Run Results Viewer, using the HP Micro Player. For details, see "How to Play a Screen Recorder Movie in the HP Micro Player" on page 1104.

System Monitor Pane (Run Results Viewer)

This pane enables you to view the system counters that you monitored for a run session in a line graph.



To access	Open the Run Results Viewer, as described in "How to Open Run Results" on page 1093. Select View > System Monitor or click the System Monitor tab.
Important information	You can export the data from the System Monitor tab to the following file types: text (.csv or .txt) , Excel , XML , or HTML . (Graphs are not exported.) For details, see "Export Run Results Dialog Box (Run Results Viewer)" on page 1141.
See also	<ul style="list-style-type: none"> ➤ For information on enabling local system monitoring, see "Local System Monitor" on page 1460. ➤ "Troubleshooting and Limitations - Viewing Run Results" on page 1168

User interface elements are described below:

UI Elements	Description
	Zoom In. Click this button and click anywhere on the graph to zoom in. You can also click and drag over an area of the graph to zoom in on that area.
	Zoom Out. Click this button and click anywhere on the graph to zoom out.
	View Full Graph. Click this button to zoom out and view the entire graph. This button is disabled when the graph is not zoomed in.
	Move. Click this button and then click and drag on the graph to scroll right and left. This button is disabled when the graph is not zoomed in.
	Arrow. Click this button and double-click anywhere on the graph to select that point as the current step. The Current Step indicator moves to the new location and the step is highlighted in the Run Results tree. You can also hover over any point on a Counter Line in the graph to see the value for the Counter Line at that point.

UI Elements	Description
Application Name	The name of the application for which system counters were monitored.
System Counters List	The list of system counters monitored for the application.
Currently Displayed Counters	The list of counters currently displayed in the line graph. The System Monitor tab displays a maximum of two counters at one time. To change the counters being displayed, clear the check box for one or both of the currently selected counters, and select the check box for the desired counters.
Counter Scale	The scale of measurement for the performance of that counter.
Maximum Counter Value	The maximum value the counter achieved during the run session.
Current Step	The point in the graph representing the step that is currently highlighted in the Run Results tree.
Counter Limit Line	A visual representation of the limit, if set, for that counter, as defined in the Local System Monitor pane of the Test Settings dialog box. If set, a counter that exceeds this limit causes the step to fail. Only the first step that exceeds the counter limit fails. Subsequent steps that exceed the counter limit are not affected.
Counter Limit Value	The numeric value of the limit, if set, for that counter, as defined in the Local System Monitor pane of the Test Settings dialog box. If set, a counter that exceeds this limit causes the step to fail. Only the first step that exceeds the counter limit fails. Subsequent steps that exceed the counter limit are not affected.
Time Scale	The scale of time in seconds, for the run session.

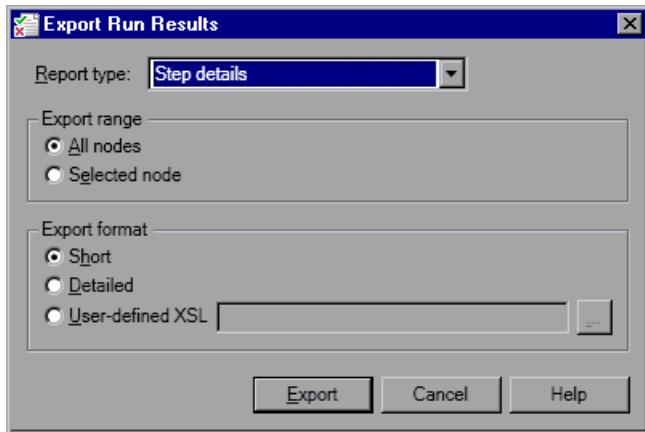
Run Results Viewer Dialog Boxes

This section includes (in alphabetical order):

- ▶ Export Run Results Dialog Box (Run Results Viewer) on page 1141
- ▶ Find Dialog Box (Log Tracking Pane - Run Results Viewer) on page 1144
- ▶ Filter Dialog Box (Run Results Viewer) on page 1146
- ▶ Open Run Results Dialog Box on page 1149
- ▶ Print Dialog Box (Run Results Viewer) on page 1151
- ▶ Print Preview Dialog Box (Run Results Viewer) on page 1153
- ▶ HP ALM Connection Dialog Box (Run Results Viewer) on page 1155

Export Run Results Dialog Box (Run Results Viewer)

This dialog box enables you to export run results to a file so that you can view them even if the Run Results Viewer is unavailable. For example, you can send the file containing the run results in an e-mail to a third-party who does not have the Run Results Viewer installed.



To access	Open the Run Results Viewer, as described in "How to Open Run Results" on page 1093. Select File > Export to File .
------------------	--

Important information	<ul style="list-style-type: none"> ➤ The export time varies according to the size of the results file and the file type you select. When selecting the file type, consider the length of time it will take to generate different document types, especially for a report with many images. HTML files generate the fastest, followed by PDF and DOC. When exporting a report with 100 or more images to a DOC file, a dialog box is displayed reminding you that it may take a long time to generate the file. The dialog box gives you the option to continue exporting with images, continue exporting without images, or to export to PDF. ➤ Screen capture images are not exported for steps on Web-based applications. When you export run results containing steps on a Web application, any screen capture images for those steps are not exported to the file. This is because for Web-based applications, the Run Results Viewer displays the HTML corresponding to the relevant Web page (with downloaded images) rather than a captured image and thus no image is saved with the report. ➤ Exporting to a DOC file. Requires that a supported version of Microsoft Word be installed on the Run Results Viewer computer. For details, see the <i>HP QuickTest Professional Product Availability Matrix</i>, available from the Documentation Library Home page or the root folder of the QuickTest Professional DVD.
Relevant tasks	"How to Export Run Results" on page 1102
See also	"Troubleshooting and Limitations - Viewing Run Results" on page 1168

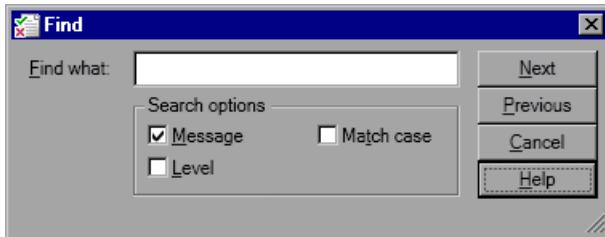
User interface elements are described below:

UI Elements	Description
Report type	The type of report you want to export, for example, Step Details or System Monitor.

UI Elements	Description
Export range	<p>Relevant only for Step Details report type.</p> <ul style="list-style-type: none"> ➤ All nodes. Exports the results for the entire test. ➤ Selected node. Exports run result information for the selected branch in the run results tree.
Export format	<p>Relevant only for Step Details report type.</p> <ul style="list-style-type: none"> ➤ Short. Exports a summary line (when available) for each item in the run results tree. The short report does not include still images associated with the steps in your run results. This option is only available if you selected All nodes in Export range. ➤ Detailed. Exports all available information for each item in the run results tree, or for the selected branch, according to your Export range selection. The detailed report includes still images associated with the steps in your run results. (In the Run Results Viewer, these images are displayed in the Captured Data pane.) If a bitmap checkpoint step displays expected, actual, and difference bitmaps, these are also included in the printed report. ➤ User-defined XSL. Enables you to browse to and select a customized .xsl file. You can create a customized .xsl file that specifies the information to be included in the exported report, and the way it should appear. For more information, see "Run Results XML File" on page 1088. <p>Note: The Export format options are available only for run results created with QuickTest 8.0 and later.</p>

Find Dialog Box (Log Tracking Pane - Run Results Viewer)

This dialog box enables you to search for a log message by message text, level, and case (upper-case/lower-case characters).



To access	<p>Open the Run Results Viewer, as described in "How to Open Run Results" on page 1093.</p> <p>Display the Log Tracking pane, as described in "Log Tracking Pane (Run Results Viewer)" on page 1132.</p> <p>In the Log Tracking pane title bar, click Find.</p>
Important information	<p>This dialog box is relevant only for the Log Tracking pane. To perform a search in the run results tree, use the Search box described on page 1114.</p> <p>To stop a search, press the Esc key on your keyboard.</p>

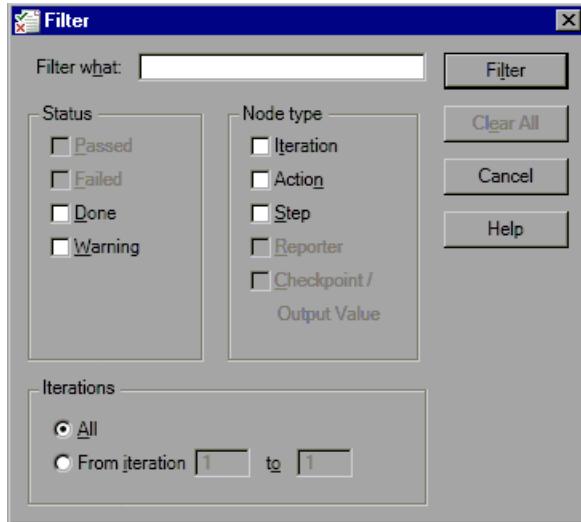
User interface elements are described below:

UI Elements	Description
Find	Enter the text to find. This can be message text or a severity level.
Message	Searches the Message column for the text specified in the Find box.

UI Elements	Description
Level	Searches the Level column for the text specified in the Find box. Possible level values: <ul style="list-style-type: none">➤ TRACE➤ DEBUG➤ INFO➤ WARN➤ ERROR➤ FATAL
Match case	Displays only those occurrences that match the capitalization (upper-case and lower-case characters) you specified for the text in the Find box.
Next	Jumps to the next instance that matches the selected criteria.
Previous	Jumps to the previous instance that matches the selected criteria.

Filter Dialog Box (Run Results Viewer)

This dialog box enables you to filter the results tree of the Run Results Viewer to display only those nodes that match the conditions that you specify.



To access Open the Run Results Viewer, as described in "How to Open Run Results" on page 1093.

Select **View > Filters** or click the **Filter** button .

Important information	<ul style="list-style-type: none"> ➤ When you apply a filter, a filter icon is displayed on the status bar, and (Filtered) is appended to the Search box title bar. These visual changes indicate that the run results tree is currently displaying only those nodes that match your filter criteria. ➤ You can use <code>Reporter.Filter</code> statements in the Expert View to disable or enable the saving of selected steps, or to save only steps with Failed or Warning status. For more information on saving run session information, see "Choosing Which Steps to Report During the Run Session" on page 964 or the <i>HP QuickTest Professional Object Model Reference</i>. The <code>Reporter.Filter</code> statement differs from the Filters dialog box described above. The <code>Reporter.Filter</code> statement determines which steps are saved in the Run Results, while the Filter dialog box determines which steps are displayed at any time.
------------------------------	--

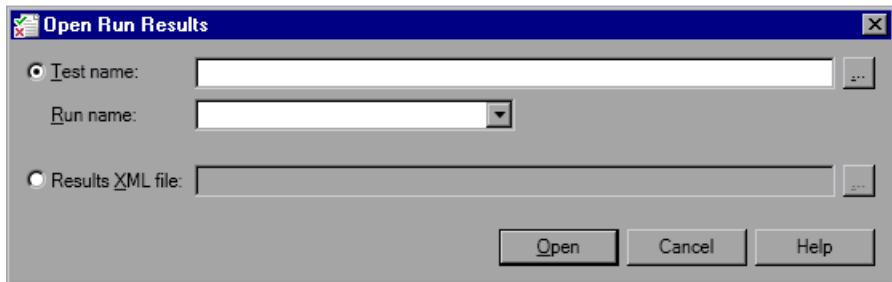
User interface elements are described below:

UI Elements	Description
Filter what	A text box in which you enter the text by which you want to filter. (Optional)
Status	<p>The status of the node by which to filter. (Optional)</p> <p>Possible values:</p> <ul style="list-style-type: none"> ➤ Passed. Displays the run results for the steps that passed and match your filter criteria. ➤ Failed. Displays the run results for the steps that failed and match your filter criteria. ➤ Done. Displays the run results for the steps with the status Done (steps that were performed successfully but did not receive a pass, fail, or warning status) that match your filter criteria. ➤ Warning. Displays the run results for the steps with the status Warning (steps that did not pass, but did not cause the test to fail) that match your filter criteria.

UI Elements	Description
Node type	<p>Displays all results that match your selection criteria based on:</p> <p>(Optional)</p> <ul style="list-style-type: none"> ➤ Iteration. Displays the run results in the run results tree for the iterations specified in the Iterations area. ➤ Action. Displays the run results for all actions in the run results tree that match your other selection criteria. ➤ Step. Displays the run results for all steps in the run results tree that match your other selection criteria. ➤ Reporter. Displays the run results for all Reporter steps in the run results tree that match your other selection criteria. <p>Note: This is not relevant for Reporter.ReportNote steps, which are displayed in the Notes area of the Executive Summary page and not in the run results tree.</p> <ul style="list-style-type: none"> ➤ Checkpoint/Output Value. Displays the run results for all checkpoint and output value steps in the run results tree that match your other selection criteria.
Iterations	<p>(This option is available only for tests.)</p> <ul style="list-style-type: none"> ➤ All. Displays run results from all iterations. ➤ From iteration X to Y. Displays the run results from the specified range of test iterations.

Open Run Results Dialog Box

This dialog box enables you to open run results in the Run Results Viewer.



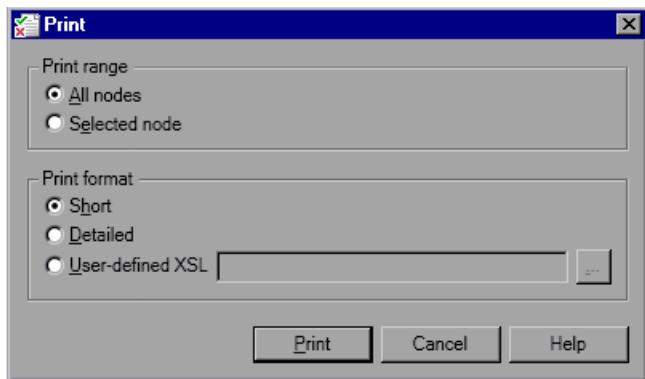
To access	Open the Run Results Viewer, as described in "How to Open Run Results" on page 1093. Select File > Open or click the Open button  .
Important information	<ul style="list-style-type: none">▶ To view results stored in Quality Center, you must first connect to your Quality Center project. For details, see "HP ALM Connection Dialog Box (Run Results Viewer)" on page 1155.▶ By default, results files for QuickTest tests are stored in: <QuickTest installation folder>\Tests\<Test name>\<ResultName>\Report.▶ To view result files for QuickTest Professional version 6.5 and earlier, use the Test/Run Results viewer in an earlier version of QuickTest.
Relevant tasks	"How to Open Run Results" on page 1093

User interface elements are described below:

UI Elements	Description
Test name	The name of the test for which you want to view results. The test can be located in the file system or in a Quality Center project. If you select this option, you must also specify the Run name .
Run name	The results for a particular run. The results are listed in the order of execution, with the most recent results at the top of the drop-down list.
Results XML file	The actual XML results file. The file must be located in the file system.

Print Dialog Box (Run Results Viewer)

This dialog box enables you to print run results from the Run Results Viewer. You can select the type of report you want to print, and you can also create and print a customized report.



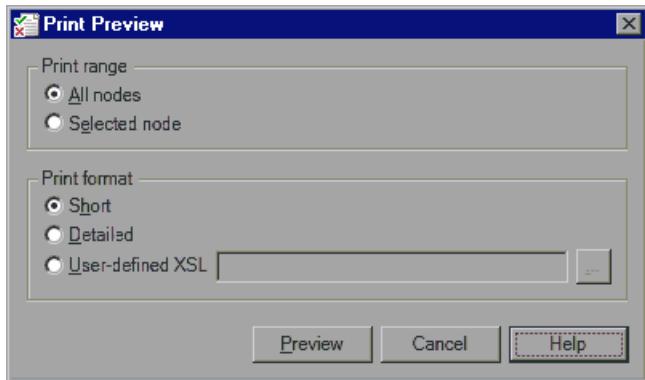
To access	<p>Open the Run Results Viewer, as described in "How to Open Run Results" on page 1093.</p> <p>Use one of the following:</p> <ul style="list-style-type: none">▶ Select File > Print.▶ Click the Print button .
------------------	---

User interface elements are described below:

UI Elements	Description
Print range	<ul style="list-style-type: none"> ➤ All. Prints the run results for the entire test. ➤ Selection. Prints the run results information for the selected branch in the run results tree.
Print format	<ul style="list-style-type: none"> ➤ Short. Prints a summary line (when available) for each item in the run results tree. This option is only available if you selected All in Print range. ➤ Detailed. Prints all available information for each item in the run results tree, or for the selected branch, according to your selection in Print range. The printed report includes still images associated with the steps in your run results. If a bitmap checkpoint step displays expected, actual, and difference bitmaps, these are also included. ➤ User-defined XSL. Enables you to browse to and select a customized .xsl file. You can create a customized .xsl file that specifies the information to be included in the printed report, and the way it should appear. For more information, see "Run Results XML File" on page 1088. <p>Note: The Print format options are available only for run results created with QuickTest version 8.0 and later.</p>
Print	Opens the standard Windows Print dialog box, enabling you to send the selected run results any installed printer.

Print Preview Dialog Box (Run Results Viewer)

This dialog box enables you to preview run results on screen before you print them. You can select the type and quantity of information you want to view, and you can also display the information in a customized format.



To access	Open the Run Results Viewer, as described in "How to Open Run Results" on page 1093. Select File > Print Preview .
Important information	The Print Preview option is available only for run results created with QuickTest version 8.0 and later.

User interface elements are described below:

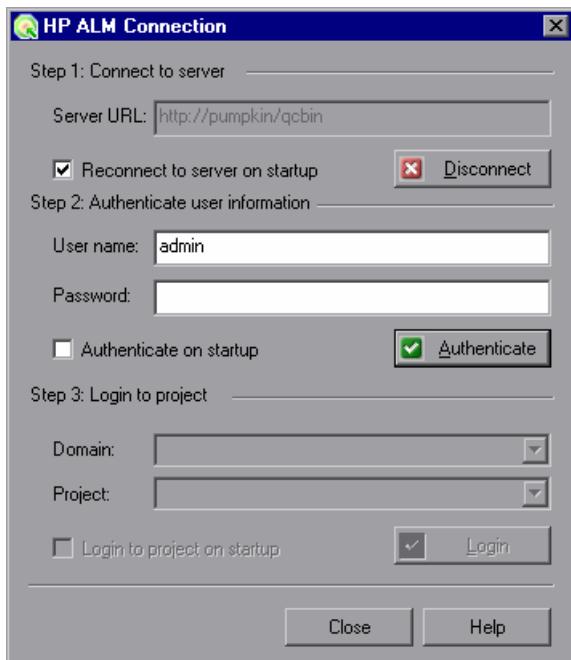
UI Elements	Description
Print range	<ul style="list-style-type: none"> ➤ All. Previews the run results for the entire test. ➤ Selection. Previews run results information for the selected branch in the run results tree.
Print format	<ul style="list-style-type: none"> ➤ Short. Previews a summary line (when available) for each item in the run results tree. This option is only available if you selected All in Print range. ➤ Detailed. Previews all available information for each item in the run results tree, or for the selected branch, according to your selection in Print range. The preview includes still images associated with the steps in your run results. If a bitmap checkpoint step displays expected, actual, and difference bitmaps, these are also included. ➤ User-defined XSL. Enables you to browse to and select a customized .xsl file. You can create a customized .xsl file that specifies the information to be included in the preview, and the way it should appear. For more information, see "Run Results XML File" on page 1088.
Preview	<p>Displays the run results on screen as they will appear when printed.</p> <p>Tip: If some of the information is cut off in the preview, for example, if checkpoint names are too long to fit in the display, click the Page Setup button  in the Print Preview window and change the page orientation from Portrait to Landscape.</p>

HP ALM Connection Dialog Box (Run Results Viewer)

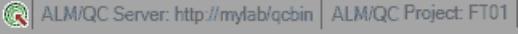
This dialog box enables you to connect or disconnect the Run Results Viewer to or from a project in any supported version of HP ALM or Quality Center.



After you perform step 1, the dialog box expands to include the remaining connection fields.



To access	<p>Use one of the following:</p> <ul style="list-style-type: none"> ➤ Select Tools > ALM/QC Connection. ➤ Click the ALM/QC Connection toolbar button .
Important information	<ul style="list-style-type: none"> ➤ 1st time connection. The first time you connect your computer to an HP ALM or Quality Center server, you must connect as a user with administrator privileges. ➤ Connecting to different HP ALM or Quality Center servers. You can simultaneously connect your Web browser to multiple HP ALM clients and one Quality Center 9.2 or 10.00 client simultaneously. While these clients are open, you can connect the Run Results Viewer to the Quality Center 9.2 or 10.00 client that is currently open on your computer. However, if you want to connect the Run Results Viewer to an HP ALM client, you must first close the Quality Center 9.2 or 10.00 client. ➤ Windows Vista, Windows 7, Windows Server 2008, and Windows Server 2008 R2 Users. See "Guidelines for Windows Vista, Windows 7, Server 2008, and Server 2008 R2 Users" on page 1159. ➤ Connect. The connection process has two stages. First, you connect the Run Results Viewer to a local or remote HP ALM or Quality Center server. This server handles the connections between the Run Results Viewer and the HP ALM or Quality Center project. Next, you log in and choose the project you want QuickTest to access. The project stores tests and run session information for the application you are testing. Projects are password protected, so you must provide a user name and a password. ➤ Disconnect. You can disconnect the Run Results Viewer from an HP ALM or Quality Center project or from an HP ALM or Quality Center server at any time. If you disconnect the Run Results Viewer from an HP ALM or Quality Center server without first disconnecting from a project, the Run Results Viewer connection to that project database is automatically disconnected.

Relevant tasks	<p>To view the current connection, look at the ALM/QC icon in the status bar.</p> 
-----------------------	---

User interface elements are described below:

UI Elements (A-Z)	Description
Authenticate / Change User	<p>Authenticates your user information against the HP ALM or Quality Center server.</p> <p>Note: After your user information has been authenticated, the edit boxes in the Authenticate user information area are displayed in read-only format. The Authenticate button changes to a Change User button.</p> <p>Tip: You can log in to the same HP ALM or Quality Center server using a different user name by clicking Change User, and then entering a new user name and password and clicking Authenticate again.</p>
Authenticate on Startup	<p>Instructs the Run Results Viewer to automatically authenticate your user information against the HP ALM or Quality Center server every time you open the Run Results Viewer.</p>
Close	<p>Closes the HP ALM Connection dialog box.</p> <p>Note: The dialog box does not close automatically. You must click this button to close the dialog box.</p>
Connect / Disconnect	<p>Connects or disconnects the Run Results Viewer to or from the selected HP ALM or Quality Center server. (After you successfully connect to a server, the button changes to Disconnect.)</p>
Domain	<p>The domain that contains the HP ALM or Quality Center project.</p> <p>Note: Only those domains to which you have permission to connect are displayed.</p>

UI Elements (A-Z)	Description
Login / Logout	<ul style="list-style-type: none"> ➤ Login. Logs into the selected domain and project using the current user information. (After you successfully log into a project, the button changes to Logout.) ➤ Logout. Logs out of the selected domain and project. <p>Note: You must click Close to close the dialog box after logging into or out of a project.</p>
Login to project on startup	Instructs the Run Results Viewer to automatically log into the selected project every time you open QuickTest.
Password	<p>Your HP ALM or Quality Center password.</p> <p>Note: To enter a password in any CJK (Chinese, Japanese, Korean) language, copy/paste the password into the edit box. (Windows does not support typed CJK characters in password fields.)</p>
Project	<p>The HP ALM or Quality Center project with which you want to work.</p> <p>Note: Only those projects for which you are a defined user are displayed.</p>
Reconnect to Server on startup	Instructs the Run Results Viewer to automatically reconnect to the HP ALM or Quality Center server every time you open the Run Results Viewer.
Server URL	<p>The URL address of the Web server where HP ALM or Quality Center is installed.</p> <p>You can choose a server that is accessible via a Local Area Network (LAN) or a Wide Area Network (WAN).</p> <p>Note: You can connect to any currently supported version of HP ALM or Quality Center. For a list of supported versions, see the <i>HP QuickTest Professional Product Availability Matrix</i>, available from the Documentation Library Home page or the root folder of the QuickTest Professional DVD.</p>
User name	Your HP ALM or Quality Center user name.

Guidelines for Windows Vista, Windows 7, Server 2008, and Server 2008 R2 Users

The security settings in the following operating systems may prevent you from connecting to an HP ALM or Quality Center project:

- Windows Vista
- Windows 7
- Windows Server 2008
- Windows Server 2008 R2

This can occur when the UAC (User Account Control) option is set to ON, and you have never connected to an HP ALM or Quality Center project.

To connect to HP ALM or Quality Center for the first time, you must disable the UAC option. After you successfully connect to HP ALM or Quality Center, you can turn the UAC option on again. Thereafter, you should be able to connect to HP ALM or Quality Center, as needed.

To enable QuickTest to connect to an HP ALM or Quality Center project:

For Microsoft Windows Vista and Windows Server 2008:

- 1** Log in as an administrator.
- 2** From the Control Panel, select **User Accounts > Change Security Settings**, and clear the **Use User Account Control (UAC) to help protect your computer** check box.
- 3** Connect to HP ALM or Quality Center, as described above.
- 4** Select the **Use User Account Control (UAC) to help protect your computer** check box and click **OK** to turn the UAC option on again.

For Microsoft Windows 7 and Windows Server 2008 R2:

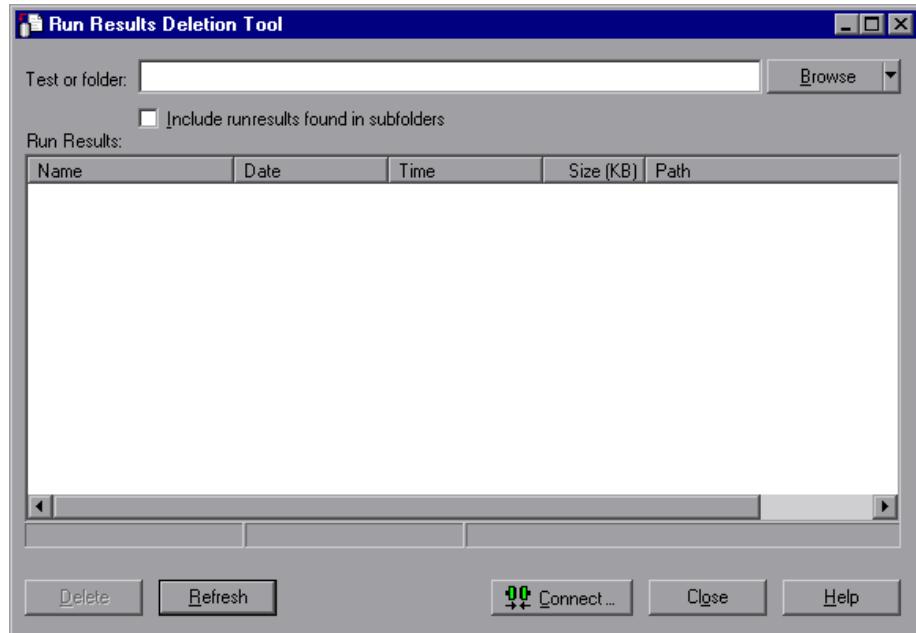
- 1** Log in as an administrator.
- 2** From the Control Panel, select **User Accounts > User Accounts > Change User Account Settings**.
- 3** In the User Account Control Settings window, move the slider to **Never notify**.

- 4 Connect to HP ALM or Quality Center, as described above.
- 5 Return to the User Account Control Settings window, and restore the slider to its previous position to turn the UAC option on again.

Run Results Deletion Tool

This window enables you to view a list of all the run results in a specific location in your file system or in a Quality Center project. You can then delete any run results that you no longer require.

You can sort the run results by name, date, size, and so on, so that you can more easily identify the results you want to delete.



To access	Select Start > Programs > HP QuickTest Professional > Tools > Run Results Deletion Tool.
Important information for Quality Center users	<p>To delete run results from a Quality Center project, click Connect to connect to Quality Center before browsing or entering the path. Specify the Quality Center test path in the standard Quality Center format.</p> <p>Example: [Quality Center] Subject\<folder name>\<test name>.</p> <p>You can delete results from only one test at a time. Make sure that you have Delete Run permission for this Quality Center project.</p> <p>For information on connecting to Quality Center, see "HP ALM Connection Dialog Box (Run Results Viewer)" on page 1155.</p> <p>For information on Quality Center project permissions, contact your Quality Center administrator or see the section on permission settings in the Quality Center or HP ALM administrator guide.</p>
See also	"Command Line Options for the Run Results Deletion Tool" on page 1162 to learn how to use this tool from a command line interface.

User interface elements are described below (unlabeled elements are shown in angle brackets):

UI Elements	Description
Test or folder	The path from which you want to delete run results. When working with the file system, you can specify a test or a folder. When working with Quality Center, you cannot specify folders.
Browse	Enables you to browse to and select the folder or specific run results that you want to delete. By default, clicking the Browse button opens the Open Test dialog box. If you want to browse to a folder, click the down arrow and select Folders to open the Open Folder dialog box.

UI Elements	Description
Include run results found in subfolders	Adds all run results contained in subfolders of the specified folder to the Run Results area of this dialog box. (Relevant only for folders in the file system. This option is not supported when working with tests in Quality Center.)
Run Results	Lists the run results that are stored in the specified test or folder, together with descriptive information for each. You can click a column's title to sort run results based on the entries in that column.
Delete	Deletes the selected run results from the file system and/or the Quality Center project. You can select multiple run results for deletion using standard Windows selection techniques.
<status bar>	Displays information regarding the displayed run results, including the number of results selected, the total number of results in the specified location and the size of the files.
Refresh	Updates the list of tests in the Run Results area.
Connect	Opens the HP ALM Connection dialog box, enabling you to connect to your Quality Center project. For more details, see "HP ALM Connection Dialog Box (Run Results Viewer)" on page 1155.



Command Line Options for the Run Results Deletion Tool

You can use command line options to specify the criteria for the run results that you want to delete using the Run Results Deletion Tool. (For details on this tool, see "Run Results Deletion Tool" on page 1160.)

If you add command line options that contain spaces, you must specify the option within quotes, for example:

```
TestResultsDeletionTool.exe -Test "F:\Tests\Keep\web objects"
```

Following is a description of each command line option:

Caution: If you use the -Silent command line option to run the Run Results Deletion Tool, all run results that meet the specified criteria are deleted. Otherwise, the Delete Run Results Viewer opens.

-Domain *Quality_Center_domain_name*

Specifies the name of the Quality Center domain to which you want to connect. This option should be used in conjunction with the -Server, -Project, -User, and -Password options.

-FromDate *results_creation_date*

Deletes run results created after the specified date. Results created on or before this date are not deleted. The format of the date is MM/DD/YYYY.

The following example deletes all results created after November 1, 2005:

```
TestResultsDeletionTool.exe -Silent -Test "C:\tests\test1" -FromDate "11/1/2005"
```

-Log *log_file_path*

Creates a log file containing an entry for each run results file in the folder or test you specified. The log file indicates which results were deleted and the reasons why other results were not. For example, results may not be deleted if they are smaller than the minimum file size you specified.

You can specify a file path and name or use the default path and name. If you do not specify a file name, the default log file name is

TestResultsDeletionTool.log in the folder where the Run Results Deletion Tool is located.

The following example creates the log file **C:\temp\Log.txt**:

```
TestResultsDeletionTool.exe -Silent -Log "C:\temp\Log.txt" -Test "C:\tests\test1"
```

The following example creates a log file named **TestResultsDeletionTool.log** in the folder where the Run Results Deletion Tool is located:

```
TestResultsDeletionTool.exe -Silent -Log -Test "C:\tests\test1"
```

-MinSize *minimum_file_size*

Deletes run results larger than or equal to the specified minimum file size. Specify the size in bytes.

Note: The -MinSize option is available only for run results in the file system. It is not supported when working with tests in Quality Center.

The following example deletes all results larger than or equal to 10000 bytes. Results that are smaller than 10000 bytes are not deleted:

```
TestResultsDeletionTool.exe -Silent -Test "C:\tests\test1" -MinSize "10000"
```

-Name *result_file_name*

Specifies the names of the result files to be deleted. Only results with the specified names are deleted.

You can use regular expressions to specify criteria for the result files you want to delete. For more information on regular expressions and regular expression syntax, see "Regular Expressions Overview" on page 863.

The following example deletes results with the name **Res1**:

```
TestResultsDeletionTool.exe -Silent -Test "C:\tests\test1" -Name "Res1"
```

The following example deletes all results whose name starts with **Res** plus one additional character: (For example, **Res1** and **ResD** would be deleted. **ResDD** would not be deleted.)

```
TestResultsDeletionTool.exe -Silent -Test "C:\tests\test1" -Name "Res."
```

-Password *Quality_Center_password*

Specifies the password for the Quality Center user name. This option should be used in conjunction with the -Domain, -Server, -Project, and -User options.

The following example connects to the **Default** Quality Center domain, using the server located at **http://QCServer/qcbin**, with the project named **Quality Center_Demo**, using the user name **Admin** and the password **PassAdmin**:

```
TestResultsDeletionTool.exe -Domain "Default" -Server "http://QCServer/qcbin"  
-Project "Quality Center_Demo" -User "Admin" -Password "PassAdmin"
```

-Project *Quality_Center_project_name*

Specifies the name of the Quality Center project to which you want to connect. This option should be used in conjunction with the -Domain, -Server, -User, and -Password options.

-Recursive

Deletes run results from all tests in a specified file system folder and its subfolders. When using the -Recursive option, the -Test option should contain the path of the folder that contains the tests results you want to delete (and not the path of a specific test).

The following example deletes all results in the **F:\Tests** folder and all of its subfolders:

```
TestResultsDeletionTool.exe -Test "F:\Tests" -Recursive
```

Note: The -Recursive option is available only for folders in the file system. It is not supported when working with tests stored in Quality Center.

-Server *Quality_Center_server_path*

Specifies the full path of the Quality Center server to which you want to connect. This option should be used in conjunction with the -Domain, -Project, -User, and -Password options.

-Silent

Instructs the Run Results Deletion Tool to run in the background (silently), without the user interface.

The following example instructs the Run Results Deletion Tool to run silently and delete all results located in C:\tests\test1:

```
TestResultsDeletionTool.exe -Silent -Test "C:\tests\test1"
```

-Test *test_or_folder_path*

Sets the test or test path from which the Run Results Deletion Tool deletes run results. You can specify a test name and path, file system path, or full Quality Center path.

This option is available only when used in conjunction with the -Silent option.

Note: The -Domain, -Server, -Project, -User, and -Password options must be used to connect to Quality Center.

The following example opens the Run Results Deletion Tool with a list of the results in the F:\Tests\Keep\webobjects folder:

```
TestResultsDeletionTool.exe -Test "F:\Tests\Keep\webobjects"
```

The following example deletes all results in the Quality Center Tests\webojects test:

```
TestResultsDeletionTool.exe -Domain "Default" -Server "http://QCServer/qcbin"  
-Project "Quality Center_Demo592" -User "Admin" -Password "PassAdmin"  
-Test "Subject\Tests\webojects"
```

Tip: The -Test option can be combined with the -Recursive option to delete all run results in the specified file system folder and all its subfolders.

-UntilDate *results_creation_date*

Deletes run results created before the specified date. Results created on or after this date are not deleted. The format of the date is MM/DD/YYYY.

This option is available only when used in conjunction with the -Silent option.

The following example deletes all results created before November 1, 2005:

```
TestResultsDeletionTool.exe -Silent -Test "C:\tests\test1" -UntilDate "11/1/2005"
```

-User *Quality_Center_user_name*

Specifies the user name for the Quality Center project to which you want to connect. This option should be used in conjunction with the -Domain, -Server, -Project, and -Password options.

This option is available only when used in conjunction with the -Silent option.

Troubleshooting and Limitations - Viewing Run Results

This section describes troubleshooting and limitations for viewing run results.

► **Local system monitor.** After you run a test with the local system monitoring option activated when the test is either very short, or the number of seconds entered for the **Enable local system monitoring every: __ seconds** option is high (a high percentage relative to the length of your entire test run), then when you select one of the last steps in the Run Results tree, the **Current Step** indicator in the System Monitor pane may jump to a position outside (to the right) of the graph.

Workaround: Add a **Wait** statement to the end of the test or reduce the number of seconds entered in the **Enable local system monitoring every: __ seconds** option.

► **Run session errors.** Errors during the run session produce more than one error node in the run results.

► **Exporting run results.** When UAC is set to ON and you select to export the Run Results to a system folder, the exported file is stored under Virtual Storage rather than under the specified folder. (Relevant for Microsoft Windows Vista, Windows 7, Windows Server 2008, and Windows Server 2008 R2)

32

Run Results - Understanding Step Results

This chapter includes:

Concepts

- ▶ Smart Identification in the Run Results on page 1170
- ▶ Checkpoint and Output Value Results on page 1174
- ▶ Parameterized Values in the Run Results on page 1188
- ▶ QuickTest Tests Containing Calls to Service Test Tests on page 1190

Reference

- ▶ XML Checkpoint Results Window (Run Results Viewer) on page 1191
- ▶ XML Output Value Results Window (Run Results Viewer) on page 1199

Concepts

Smart Identification in the Run Results

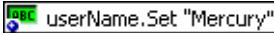
If the learned description does not enable QuickTest to identify the specified object in a step, and a Smart Identification definition is defined (and enabled) for the object, then QuickTest tries to identify the object using the Smart Identification mechanism. The following examples illustrate two possible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

- "Smart Identification - No Object Matches the Learned Description" on page 1170
- "Smart Identification - Multiple Objects Match the Learned Description" on page 1172

Smart Identification - No Object Matches the Learned Description

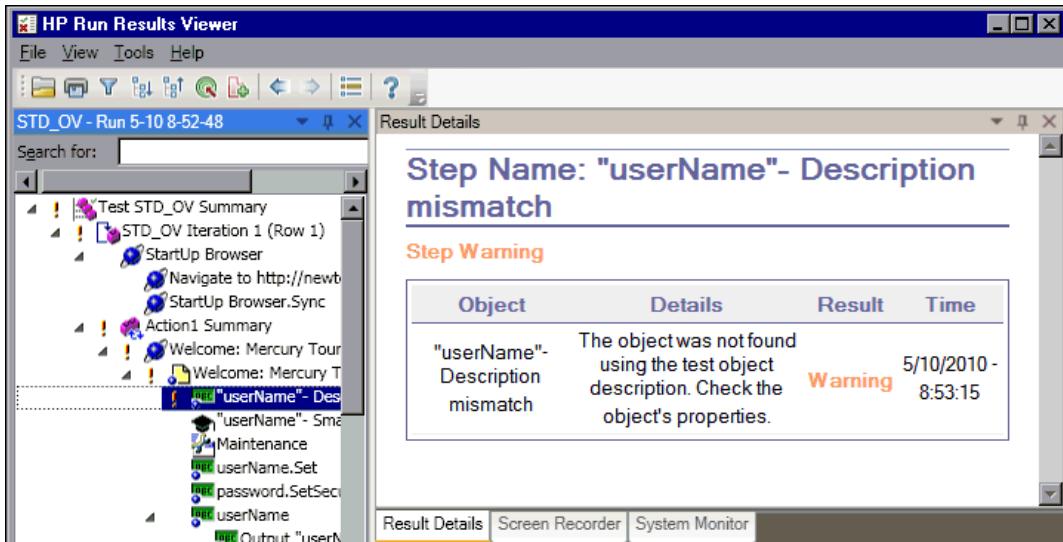
If QuickTest successfully uses Smart Identification to find an object after no object matches the learned description, the Run Results display a warning status and include the following information:

In the results tree	In the Result Details pane
A description mismatch icon for the missing object. For example:  "userName"- Description mismatch	An indication that the object (for example, the <code>userName</code> WebEdit object) was not found.
A Smart Identification icon for the missing object. For example:  "userName"- Smart Identification	An indication that the Smart Identification mechanism successfully found the object, and information on the properties used to find the object. You can use this information to modify the learned test object description, so that QuickTest can find the object using the description in future run sessions.

In the results tree	In the Result Details pane
The actual step performed. For example: 	Normal result details for the performed step.

For more information on the Smart Identification mechanism, see Chapter 7, "Configuring Object Identification."

The image below shows the results for a test in which Smart Identification was used to identify the `userName` WebEdit object after one of the learned description property values changed.



Smart Identification - Multiple Objects Match the Learned Description

If QuickTest successfully uses Smart Identification to find an object after multiple objects are found that match the learned description, QuickTest shows the Smart Identification information in the Run Results Viewer. The step still receives a passed status, because in most cases, if Smart Identification was not used, the test object description plus the ordinal identifier could have potentially identified the object.

In such a situation, the Run Results show the following information:

In the results tree	In the Result Details pane
A Smart Identification icon for the missing object. For example:  "home"- Smart Identification	An indication that the Smart Identification mechanism successfully found the object, and information on the properties used to find the object. You can use this information to create a unique object description for the object, so that QuickTest can find the object using the description in future run sessions.
The actual step performed. For example:  home.Click	Normal result details for the performed step.

The image below shows the results for a test in which Smart Identification was used to uniquely identify the Home object after the learned description resulted in multiple matches.

The screenshot shows the 'HP Run Results Viewer' window. On the left, there's a tree view of test steps under 'Web_To...'. In the center, the 'Result Details' pane is open for a step named 'Step Name: "home"- Smart Identification'. The 'Step Done' section contains a table:

Object	Details	Result	Time
"home"- Smart Identification	<p>The smart identification mechanism was invoked.</p> <p>Reason: object not unique (3 objects found)</p> <p>Original description: micclass=Image image type=Image Link html tag=IMG alt=</p> <p>Smart Identification Alternative Description:</p> <p><u>Base filter properties (11 objects found)</u> micclass=Image html tag=IMG</p> <p><u>Optional filter properties</u> alt= (Used, 10 matches) image type=Image Link (Used, 3 matches) html id= (Used, 3 matches) name=Image (Used, 3 matches) file name=home.gif (Used, 1 matches) class= (Ignored) visible=1 (Ignored) width=118 (Ignored)</p>	Done	5/4/2010 - 18:27:18

If the Smart Identification mechanism cannot successfully identify the object, the test fails and a normal failed step is displayed in the Run Results.

Checkpoint and Output Value Results

The information displayed in the Run Results Viewer and the available options are determined by the type of checkpoint or output value step you select.

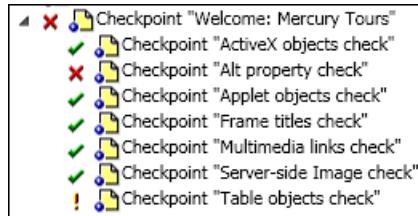
This section includes:

- "Accessibility Checkpoint Results" on page 1174
- "Bitmap Checkpoint Results" on page 1180
- "Standard Checkpoint Results" on page 1182
- "Table and Database Checkpoint Results" on page 1183
- "Text and Text Area Checkpoint Results" on page 1184
- "XML Checkpoint Results" on page 1185
- "Output Value Results" on page 1186
- "XML Output Value Results" on page 1187

Accessibility Checkpoint Results

When you include accessibility checkpoints in your test, the Run Results Viewer displays the results of each accessibility option that you checked.

The run results tree displays a separate step for each accessibility option that was checked in each checkpoint. For example, if you selected all accessibility options, the run results tree for an accessibility checkpoint may look something like this:



The run result details provide information that can help you pinpoint parts of your Web site that may not conform to the W3C Web Content Accessibility Guidelines. The information provided for each check is based on the W3C requirements.

Note: Some of the W3C Web Content Accessibility Guidelines that are relevant to accessibility checkpoints are cited or summarized in the following sections. This information is not comprehensive. When checking whether your Web site satisfies the W3C Web Content Accessibility Guidelines, you should refer to the complete document at: <http://www.w3.org/TR/WAI-WEBCONTENT/>.

For more information on accessibility checkpoints, see the section on testing Web objects in the *HP QuickTest Professional Add-ins Guide*.

This section includes:

- "ActiveX Check" on page 1176
- "Alt Property Check" on page 1176
- "Applet Check" on page 1177
- "Frame Titles Check" on page 1177
- "Multimedia Links Check" on page 1178
- "Server-Side Image Check" on page 1178
- "Tables Check" on page 1179

ActiveX Check

Guideline 6 of the W3C Web Content Accessibility Guidelines requires you to ensure that pages are accessible even when newer technologies are not supported or are turned off. When you select the ActiveX check, QuickTest checks whether the selected page or frame contains any ActiveX objects. If it does not contain any ActiveX objects, the checkpoint passes. If the page or frame does contain ActiveX objects then the results display a warning and a list of the ActiveX objects so that you can check the accessibility of these pages on browsers without ActiveX support. For example:

ActiveX objects check	
Object Tag	Object Name
OBJECT	ControlX

Alt Property Check

Guideline 1.1 of the W3C Web Content Accessibility Guidelines requires you to provide a text equivalent for every non-text element. The Alt property check checks whether objects that require the Alt property under this guideline, do in fact have this attribute. If the selected frame or page does not contain any such objects, or if all such objects have the required attribute, the checkpoint passes. If one or more objects that require the property do not have it, the test fails and the run result details display a list that shows which objects are lacking the attribute. For example:

Alt property check		
Object Tag	Object Name	Alt Value
IMG	logo	[NONE]
IMG	Dogbert	Dogbert

The Captured Data pane displays the captured page or frame, so that you can see the objects listed in the Alt property check list.

Applet Check

The Applet Check also helps you ensure that pages are accessible, even when newer technologies are not supported or are turned off (Guideline 6 of the W3C Web Content Accessibility Guidelines), by finding any Java applets or applications in the checked page or frame. The checkpoint passes if the page or frame does not contain any Java applets or applications. Otherwise, the results display a warning and a list of the Java applets and applications. For example:

Applet objects check	
Object Tag	Object Name
APPLET	JavaClock.class

Frame Titles Check

Guideline 12.1 of the W3C Web Content Accessibility Guidelines requires you to title each frame to facilitate frame identification and navigation. When you select the Frame Titles check, QuickTest checks whether Frame and Page objects have the TITLE tag. If the selected page or frame and all frames within it have titles, the checkpoint passes. If the page, or one or more frames, do not have the tag, the test fails and the run result details display a list that shows which objects are lacking the tag. For example:

Frame titles check			
Object Class	Object Tag	Object Name	Title Value
Frame	IFRAME	takeOver	Takeover Ad
Frame	IFRAME	adSpotFrame5	Click here to find out more!
Frame	IFRAME	theFrame	[NONE]
Page		NBA.com	NBA.com

The Captured Data pane displays the captured page or frame, so that you can see the frames listed in the Frame Titles check list.

Multimedia Links Check

Guidelines 1.3 and 1.4 of the W3C Web Content Accessibility Guidelines require you to provide an auditory, synchronized description of the visual track of a multimedia presentation. Guideline 6 requires you to ensure that pages are accessible, even when newer technologies are not supported or are turned off. The Multimedia Links Check identifies links to multimedia objects so that you can confirm that alternate links are available where necessary. The checkpoint passes if the page or frame does not contain any multimedia links. Otherwise, the results display a warning and a list of the multimedia links.

Server-Side Image Check

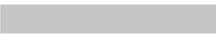
Guideline 1.2 of the W3C Web Content Accessibility Guidelines requires you to provide redundant text links for each active region of a server-side image map. Guideline 9.1 recommends that you provide client-side image maps instead of server-side image maps except where the regions cannot be defined with an available geometric shape. When you select the Server-side Image check, QuickTest checks whether the selected page or frame contains any server-side images. If it does not, the checkpoint passes. If the page or frame does contain server-side images, then the results display a warning and a list of the server-side images so that you can confirm that each one answers the guideline requirements. For example:

Server-side Image check	
Object Class	Object Name
Image	[Historical Congressional Documents]

Tables Check

Guideline 5 of the W3C Web Content Accessibility Guidelines requires you to ensure that tables have the necessary markup to be transformed by accessible browsers and other user agents. It emphasizes that you should use tables primarily to display truly tabular data and to avoid using tables for layout purposes unless the table still makes sense when linearized. The TH, TD, THEAD, TFOOT, TBODY, COL, and COLGROUP tags are recommended so that user agents can help users to navigate among table cells and access header and other table cell information through auditory means, speech output, or a Braille display.

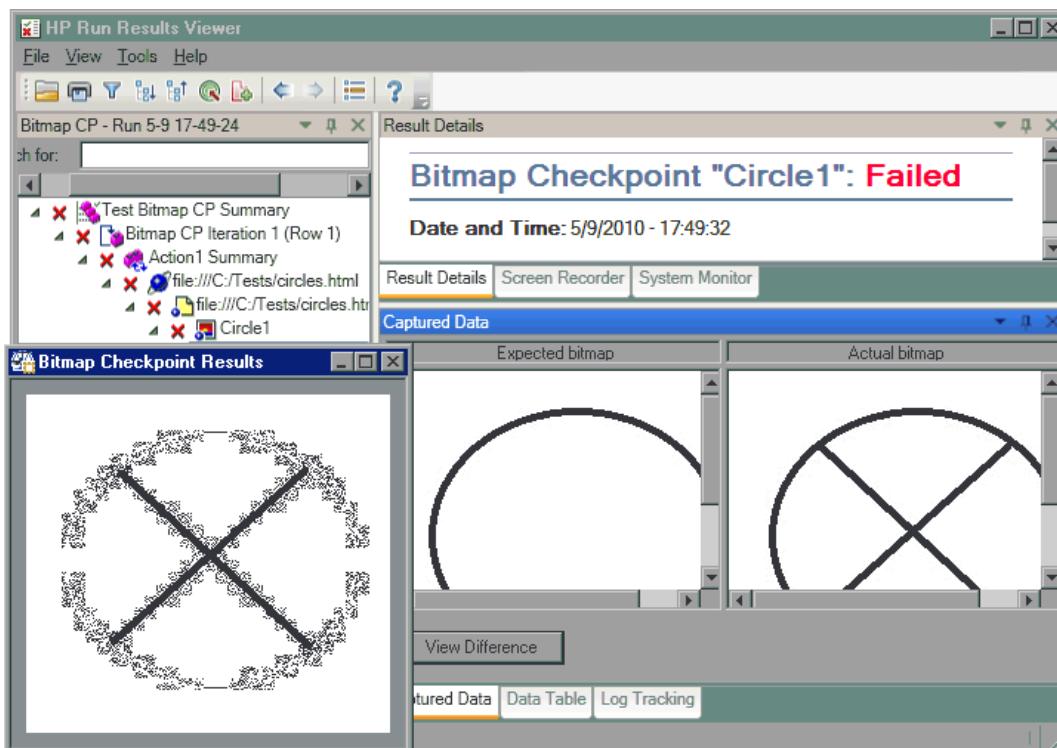
The Tables Check checks whether the selected page or frame contains any tables. If it does not, the checkpoint passes. If the page or frame does contain tables, the results display a warning and a visual representation of the tag structure of the table. For example:

Table objects check		
Object Class	Object Name	Table Structure
WebTable	Table 1	

Bitmap Checkpoint Results

The Result Details pane displays the checkpoint step results, including its status (**Passed** or **Failed**), the date and time the checkpoint was run and the portion of the checkpoint timeout interval that was used (if any).

The Captured Data pane shows the expected and actual bitmaps that were compared during the run session, and a **View Difference** button. When you click the **View Difference** button, QuickTest opens the Bitmap Checkpoint Results window, displaying an image that represents the difference between the expected and actual bitmaps. This image is a black-and-white bitmap that contains a black pixel for every pixel that is different in the two images.



Note: By default, the information in the Captured Data pane is available only if the bitmap checkpoint fails. You can change the conditions for when bitmaps are saved in the run results, using the **Save still image captures to results** option in the Run > Screen Capture pane of the Options dialog box. For more information, see "Screen Capture Pane (Options Dialog Box)" on page 1450.

Considerations for Reviewing Bitmap Checkpoint Results

- If the checkpoint is defined to compare only a specific area of the bitmap, the run results display the actual and expected bitmaps with the selected area highlighted.
- When the dimensions of the actual and expected bitmaps are different, QuickTest fails the checkpoint without comparing the bitmaps. In this case the **View Difference** functionality is not available in the results.
- The **View Difference** functionality is not available when viewing results generated in a version of QuickTest earlier than 10.00.
- If the bitmap checkpoint is performed by a custom comparer:
 - QuickTest passes the bitmaps to the custom comparer for comparison even if their dimensions are different.
 - The Result Details pane also displays the name of the custom comparer (as it appears in the **Comparer** box in the Bitmap Checkpoint Properties dialog box), and any additional information provided by the custom comparer.
 - The difference bitmap is provided by the custom comparer.

For more information on using custom comparers for bitmap checkpoints, see "Fine-Tuning the Bitmap Comparison" on page 621.

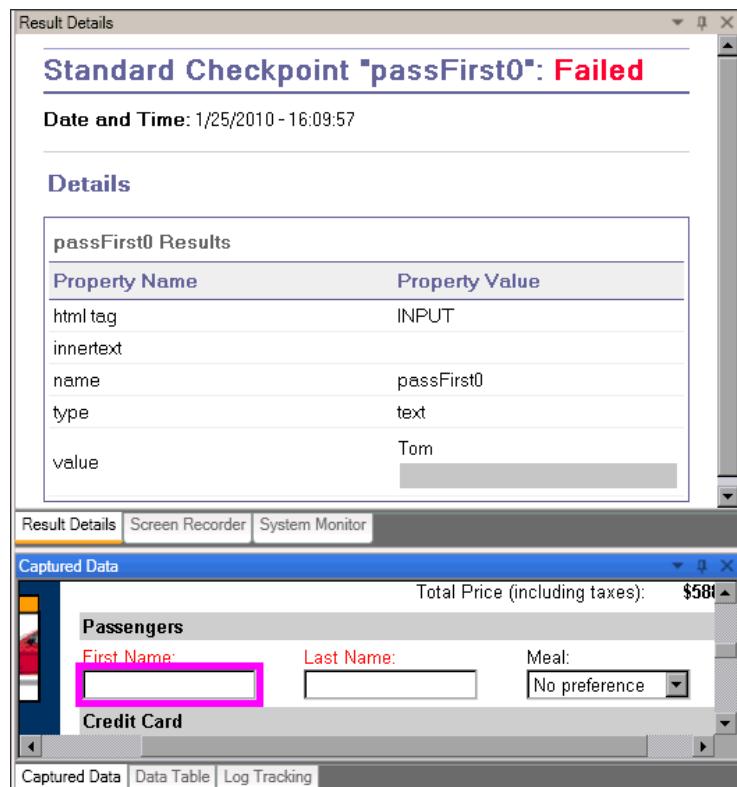
For more information on bitmap checkpoints, see Chapter 17, "Bitmap Checkpoints."

Standard Checkpoint Results

The Result Details pane displays detailed results of the selected checkpoint, including its status (**Passed** or **Failed**), the date and time the checkpoint was run, and the portion of the checkpoint timeout interval that was used (if any). It also displays the values of the object properties that are checked, and any differences between the expected and actual property values.

The Captured Data pane displays the image capture for the checkpoint step (if available).

In the following example, the details of the failed checkpoint indicate that the expected results and the current results do not match. The expected value of the flight departure is **Paris**, but the actual value is **Frankfurt**.



Result Details

Standard Checkpoint "passFirst0": Failed

Date and Time: 1/25/2010 - 16:09:57

Details

passFirst0 Results	
Property Name	Property Value
html tag	INPUT
innertext	
name	passFirst0
type	text
value	Tom

Captured Data

Total Price (including taxes): \$581

Passengers

First Name:

Last Name:

Meal:

Credit Card

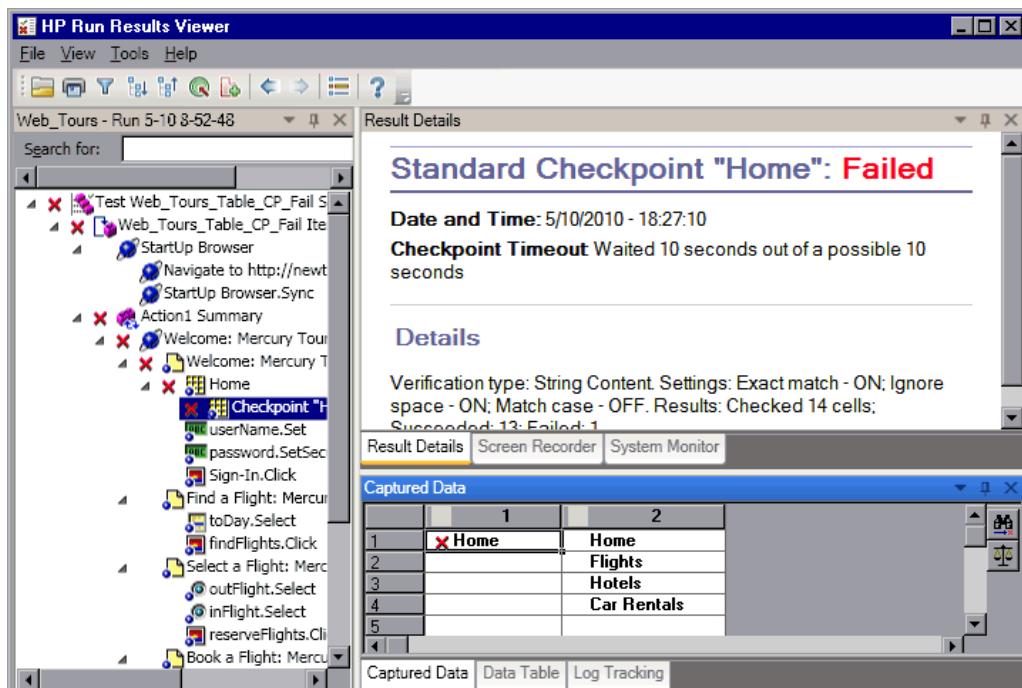
For more information on standard checkpoints, see "Standard Checkpoints" on page 605.

Table and Database Checkpoint Results

The results displayed for table and database checkpoints are similar. The Result Details pane displays the checkpoint step results, including its status (**Passed** or **Failed**), the date and time the checkpoint was run, the verification settings you specified for the checkpoint, and the number of individual table cells or database records that passed and failed the checkpoint.

If the checkpoint failed, the Captured Data pane shows the table cells or database records that were checked by the checkpoint. Cell values or records that were checked are displayed in black; cell values or records that were not checked are displayed in gray. Cells or records that failed the checkpoint are marked with a failed  icon.

The following is an example of the results for a table checkpoint:



The screenshot shows the HP Run Results Viewer interface. The left pane displays a hierarchical tree of test steps, with several steps failing, indicated by red X icons. The middle pane, titled "Result Details", shows the details for a "Standard Checkpoint 'Home': Failed". It includes the date and time (5/10/2010 - 18:27:10), a checkpoint timeout message, and a "Details" section with verification settings and results. The right pane, titled "Captured Data", shows a table with 5 rows and 2 columns. The first row contains a red X icon next to the value "Home" in the first column, indicating a mismatch. The other rows show "Flights", "Hotels", and "Car Rentals".

1	 Home
2	Home
3	Flights
4	Hotels
5	Car Rentals



You can click the **Next Mismatch** button in the Captured Data pane to highlight the next table cell or database record that failed the checkpoint.



You can click the **Compare Values** button in the Captured Data pane to display the expected and actual values of the selected table cell or database record.

For more information on table and database checkpoints, see Chapter 18, "Table Checkpoints" and Chapter 20, "Database Checkpoints."

Text and Text Area Checkpoint Results

The Result Details pane displays the checkpoint step results, including its status (**Passed** or **Failed**), the date and time the checkpoint was run and the portion of the checkpoint timeout interval that was used (if any). It also shows the expected text and actual text that was checked, and the verification settings you specified for the checkpoint.

The following is an example of the results for a text checkpoint:

The screenshot shows the HP Run Results Viewer interface. The left pane displays a tree view of recorded steps, including 'Welcome: Mercury Tours', 'Find a Flight: Mercury', and 'Book a Flight: Mercury'. The right pane, titled 'Result Details', shows a failed text checkpoint for 'Book a Flight: Mercury Tours'. The details pane indicates the capture was taken on 5/9/2010 at 17:41:53. The 'Details' section shows the captured text: 'Text Checkpoint captured "Last Name:" between First Name: and Meal: No preference Bland Diabetic Hindu Kosher Low Calorie Low Cholesterol Low Sodium Muslim Vegetarian, expected "Pinimog"'. The 'Captured Data' pane shows a screenshot of a web form with a pink border around the 'Passenger' and 'Credit Card' fields. The 'First Name' field contains 'John', the 'Last Name' field contains 'Doe', and the 'Meal' dropdown is set to 'No preference'. The 'Card Type' field is empty, and the 'Number' and 'Expiration' fields are also empty.

For more information on text and text area checkpoints, see Chapter 19, "Text Checkpoints."

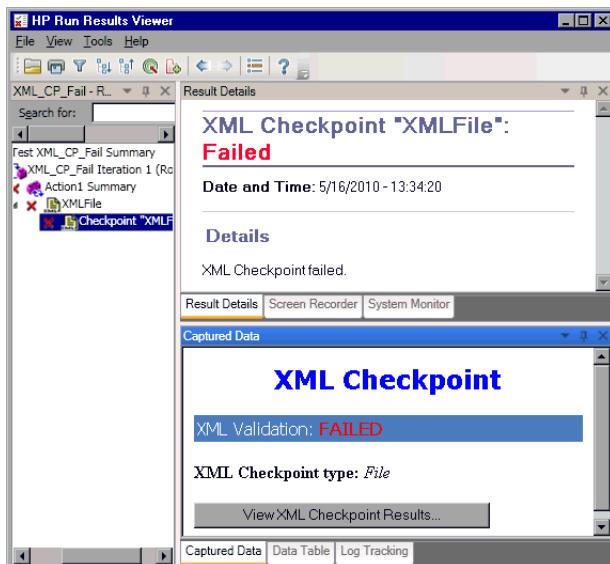
XML Checkpoint Results

The Result Details pane displays the checkpoint step results.

The Captured Data pane shows the details of the schema validation (if applicable) and a summary of the checkpoint results. If the schema validation failed, the reasons for the failure are also shown.

If the checkpoint failed, you can view details of each check performed in the checkpoint by clicking **View XML Checkpoint Results** in the Captured Data pane. The XML Checkpoint Results window opens, displaying details of the checkpoint's failure. For details, see "XML Checkpoint Results Window (Run Results Viewer)" on page 1191.

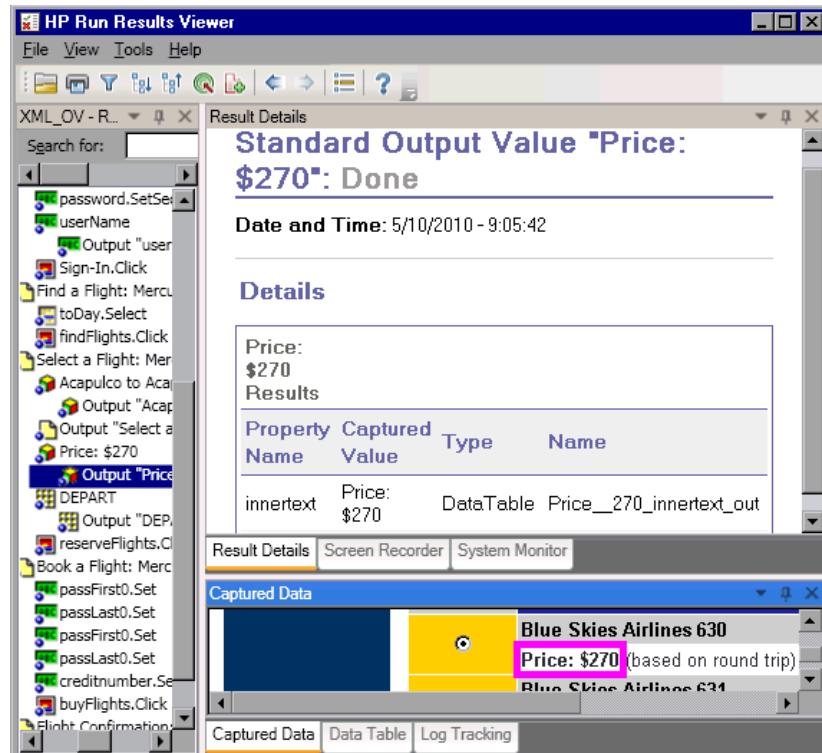
The following is an example of the results for an XML checkpoint:



Note: By default, if the checkpoint passes, the **View XML Checkpoint Results** button is not available. The availability of these detailed results is dependent on the **Save still image captures to results** setting in the **Run > Screen Capture** pane of the Options dialog box. For more information, see "Screen Capture Pane (Options Dialog Box)" on page 1450.

Output Value Results

The Result Details pane displays detailed results of the selected output value step, including its status, and the date and time the output value step was run. It also displays the details of the output value, including the value that was captured during the run session, its type, and its name, as shown in the following example.



For details on output values, see Chapter 23, "Output Values."

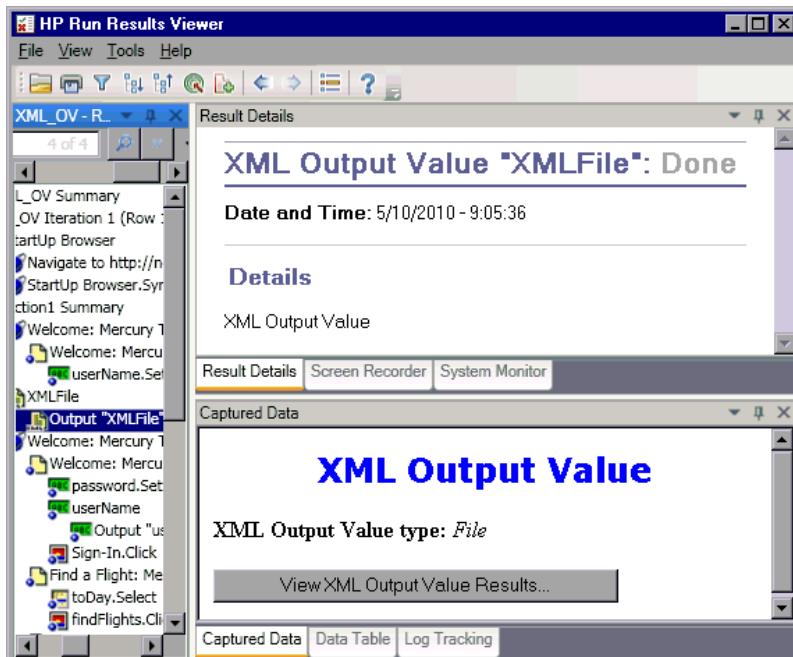
For details on XML output value steps, see "XML Output Value Results" on page 1187.

XML Output Value Results

The Result Details pane displays a summary of the output value results.

From the Captured Data pane, you can view detailed results by clicking **View XML Output Value Results** to open the XML Output Value Results window. For details, see "XML Output Value Results Window (Run Results Viewer)" on page 1199.

The following is an example of the results for an XML output value



Note: By default, the **View XML Output Value Results** button is available only when an error occurs. The availability of these detailed results is dependent on the **Save still image captures to results** setting in the **Run > Screen Capture** pane of the Options dialog box. For more information, see "Screen Capture Pane (Options Dialog Box)" on page 1450.

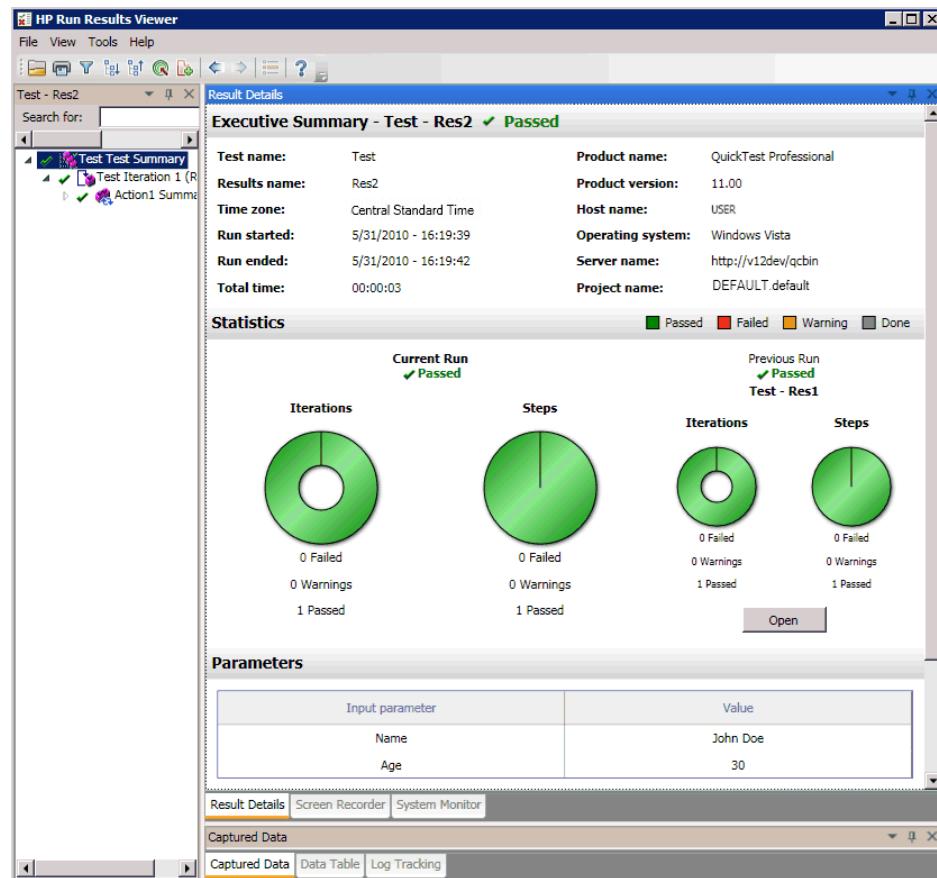
For more information on XML output values, see "How to Create or Modify an XML Output Value Step" on page 798.

Parameterized Values in the Run Results

A **parameter** is a variable that is assigned a value from an external data source or generator. You can view the values for the parameters defined in your test in the Run Results Viewer.

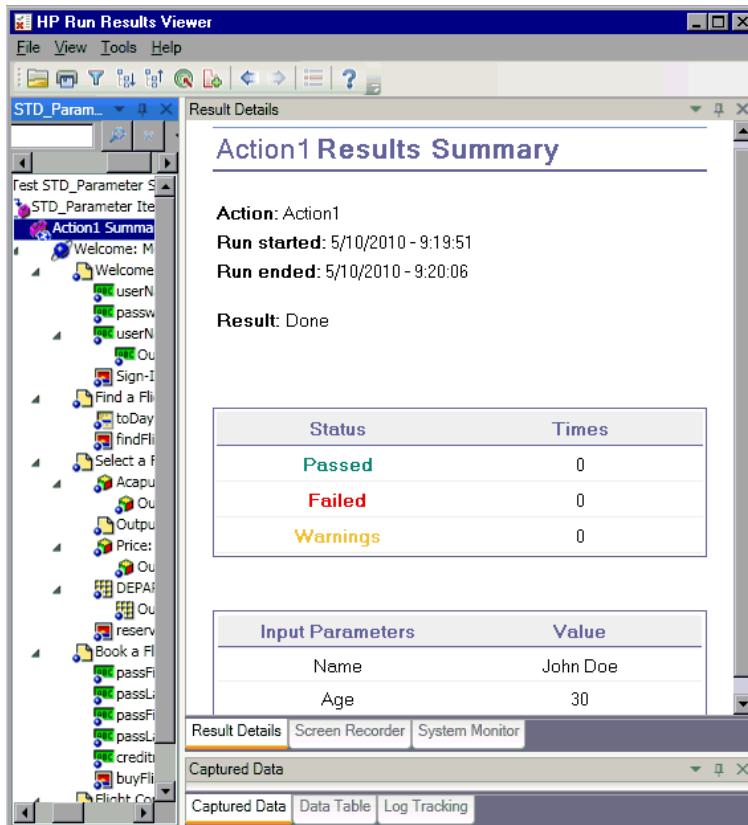
To view parameterized values, expand the nodes in the run results tree and click the root node to view test input and output parameters, or click an action node that contains parameterized values.

Test parameters are displayed in the **Parameters** section of the Executive Summary area of the Results Details pane, which you display by clicking the root node of the run results tree. The example below shows input test parameters.



If output test parameters were defined, they would be displayed in this pane beneath the input parameters.

For action parameters, the name and value of the input and output parameters are displayed in the Result Details pane.



The example above shows input parameters that were defined at the action level. If output parameters were defined at this level, they would also be displayed in this pane.

For more information on defining and using parameters in your tests, see Chapter 22, "Parameterizing Values."

QuickTest Tests Containing Calls to Service Test Tests

If your test contains a call to a Service Test test, you can view the results of that test in your QuickTest run results. The run results tree displays all of the QuickTest-specific nodes that preceded the call to the Service Test test, all of the Service Test-specific nodes from that test call, and all of the QuickTest-specific nodes that followed that call.

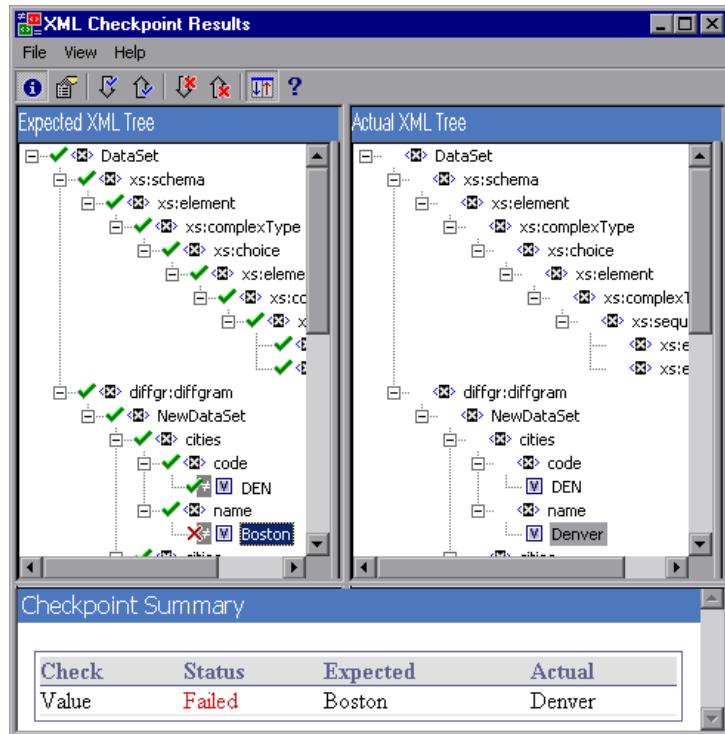
For details on what is displayed for Service Test steps, see the Run Results Viewer Help.

Reference

XML Checkpoint Results Window (Run Results Viewer)

This window displays the XML file hierarchy.

- **Expected XML Tree pane.** Displays the expected results—the elements, attributes, and values, as stored in your XML checkpoint.
- **Actual XML Tree pane.** Displays the actual results—what the XML document actually looked like during the run session.
- **Checkpoint Summary pane.** Displays results information for the check performed on the selected item in the expected results pane.



To access	In the Captured Data pane of the Run Results Viewer, click the View XML Checkpoint Results button.
Important information	When you open the XML Checkpoint Results window, the Checkpoint Summary pane displays the summary results for the first checked item in the expected results pane. Tip: You can double-click any element value in this window to open the Element Value dialog box, which displays the value in a multi-line edit control. For details, see "Element Value Dialog Box (Run Results Viewer)" on page 1198.
See also	"XML Checkpoint Results" on page 1185

User interface elements are described below:

UI Elements	Description
	View Checkpoint Summary. Displays the Checkpoint Summary pane, which provides a detailed description of which parts of an element passed or failed. Menu option: View > Checkpoint Summary.
	View Attribute Details. Displays the Expected Attributes and Actual Attributes panes, for an element whose attributes were checked. Menu option: View > Attribute Details.
	Find Next Check. Jumps directly to the next checked item in the XML Tree. Menu option: View > Find Next Check
	Find Previous Check. Jumps directly to the previous checked item in the XML Tree. Menu option: View > Find Previous Check

UI Elements	Description
	Find Next Error. Jumps directly to the next error in the XML Tree. Menu option: View > Find Next Error
	Find Previous Error. Jumps directly to the previous error in the XML Tree. Menu option: View > Find Previous Error
	Scroll Trees Simultaneously. Synchronizes the scrolling of the Expected and Actual XML Trees. If this option is selected, the Expected and Actual XML Trees scroll simultaneously as you navigate through either of the tree structures. If this option is not selected, you can scroll only one tree at a time. Menu option: View > Scroll Trees Simultaneously
	Help Topics. Opens the help for the XML Checkpoint Results window. Menu option: View > Help Topics

Sample XML Checkpoint Result Scenarios

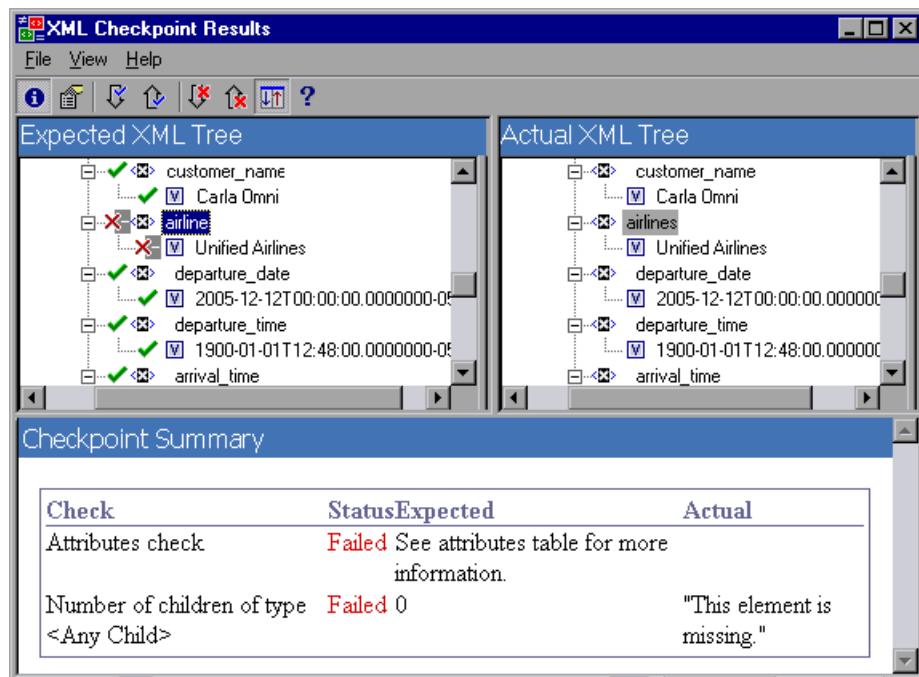
Below are four sample XML checkpoint scenarios. Each example describes the changes that occurred in the actual XML document, explains how you locate the cause of the problem in the XML checkpoint results, and displays the corresponding XML Checkpoint Results window.

Scenario 1

In the following example, the `airline` element tag was changed to `airlines` and the XML checkpoint identified the change in the tag structure. The `airline` element's child element check also failed because of the mismatch at the parent element level.

To view details of the failed element, select the `airline` tag from the Expected XML Tree and select **View > Checkpoint Summary** to view the Checkpoint Summary in the bottom pane in the XML Checkpoint Results window.

The text "This element is missing" indicates that the `airline` element tag changed in your XML document.

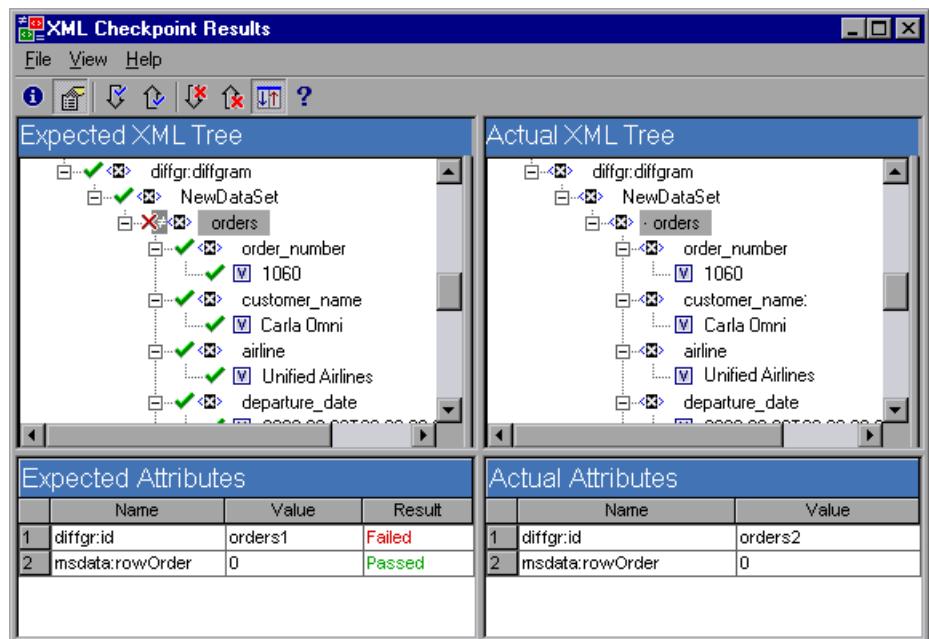


Scenario 2

In the following example, an attribute that is associated with the orders element tag was changed from the original, expected value of `orders1`, to a new value of `orders2`.

To view details of the failed attribute, select the failed element from the Expected XML Tree and select **View > Attribute Details**. The Expected Attributes and Actual Attributes panes are displayed at the bottom of the XML Checkpoint Results window.

Using the Expected Attributes and Actual Attributes panes, you can identify which attribute caused the error and which values were mismatched.

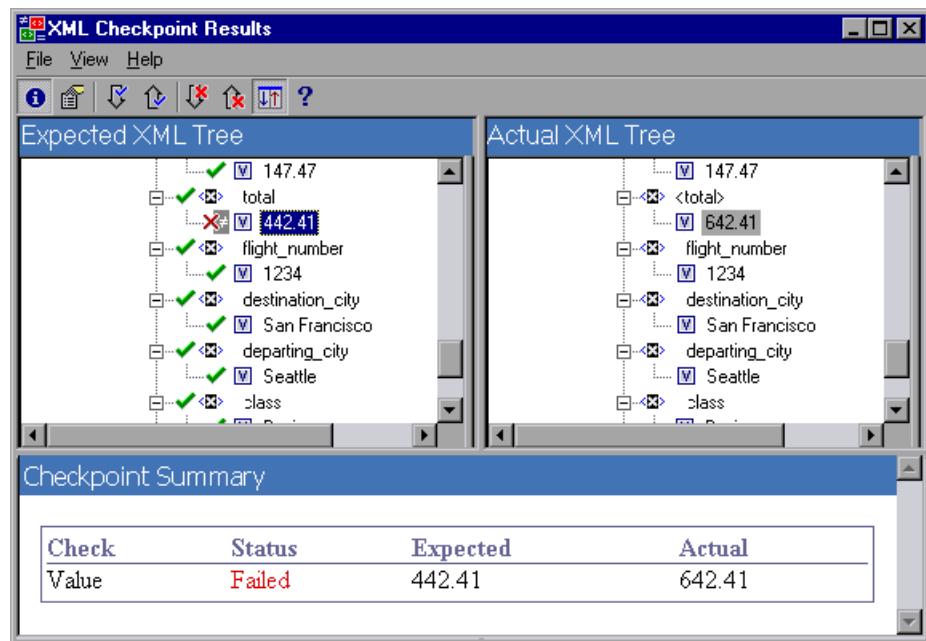


Scenario 3

In the following example, the actual value of the total element was changed between execution runs, causing the checkpoint to fail.

To view details of the failed value, select the failed element from the Expected XML Tree and select **View > Checkpoint Summary** to view the Checkpoint Summary in the bottom pane in the XML Checkpoint Results window.

Using the Checkpoint Summary pane, you can compare the expected and actual values of the total element.



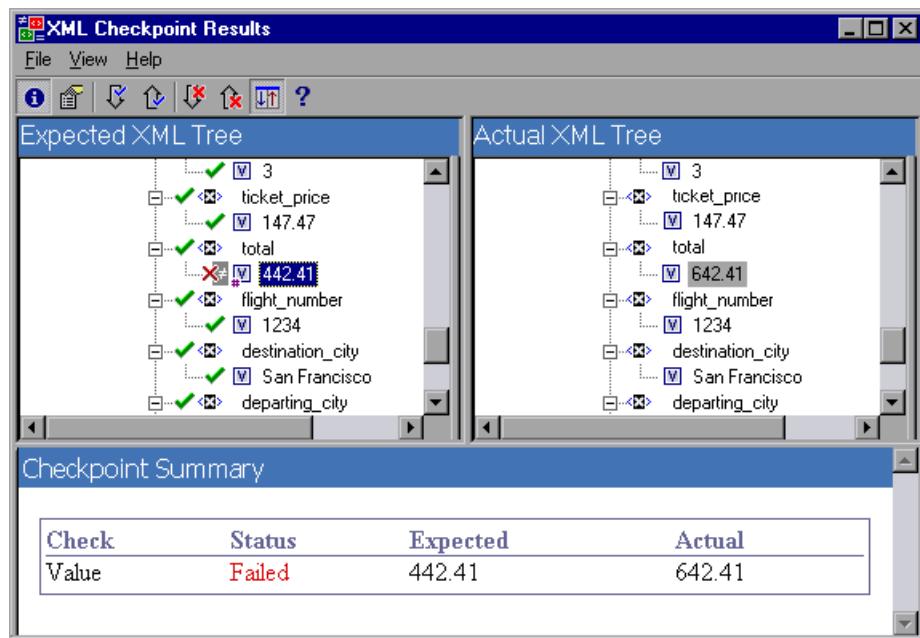
Scenario 4

In the following example, the value of the total element was parameterized and the value's content caused the checkpoint to fail in this iteration.

Note that the value icon is displayed with a pound symbol to indicate that the value was parameterized.

To view details of the failed value, select the failed element from the Expected XML Tree and select **View > Checkpoint Summary** to view the Checkpoint Summary in the bottom pane in the XML Checkpoint Results window. Note that the procedure for analyzing the checkpoint results does not change even though the value was parameterized.

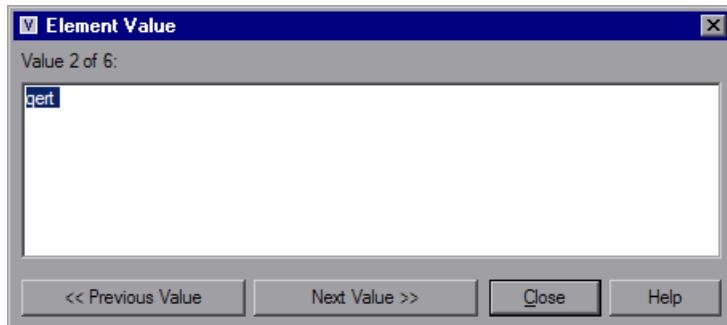
Using the Checkpoint Summary pane, you can compare the expected and actual values of the total element.





Element Value Dialog Box (Run Results Viewer)

This dialog box enables you to view element values from the XML Checkpoint Results window in a multi-line edit window. It also enables you to navigate between the values in the **Expected XML Tree** or **Actual XML Tree**.



To access	Double-click a value in the XML Checkpoint Results window.
See also	"XML Checkpoint Results" on page 1185

User interface elements are described below (unlabeled elements are shown in angle brackets):

UI Elements	Description
Value x of y	Indicates the ordinal position of the selected value within the Expected XML Tree or Actual XML Tree .
<edit window>	Displays the full value of the element or attribute in a multi-line window.
<< Previous Value	Enables you to navigate backward through the element values in the XML Checkpoint Results window. Clicking this button displays the next value in the Expected XML Tree or Actual XML Tree .
Next Value >>	Enables you to navigate forward through the element values in the XML Checkpoint Results window. Clicking this button displays the next value in the Expected XML Tree or Actual XML Tree .

XML Output Value Results Window (Run Results Viewer)

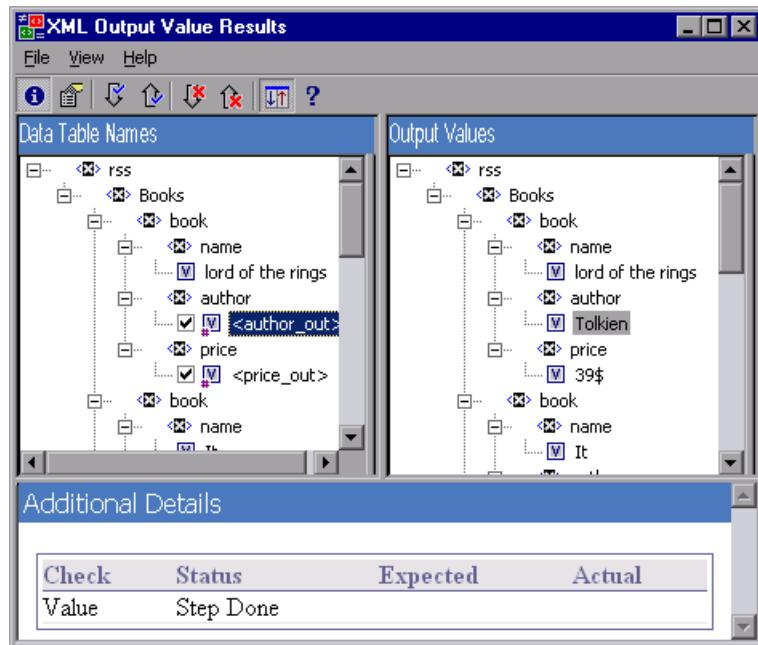
This window displays the XML file hierarchy in the following panes.

- **Data Table Names pane.** Displays the XML output value settings—the structure of the XML and the Data Table parameters (column names) you selected to output for Data Table output values.
- **Output Values pane.** Displays the actual XML tree—what the XML document or file actually looked like and the actual values that were output during the run.

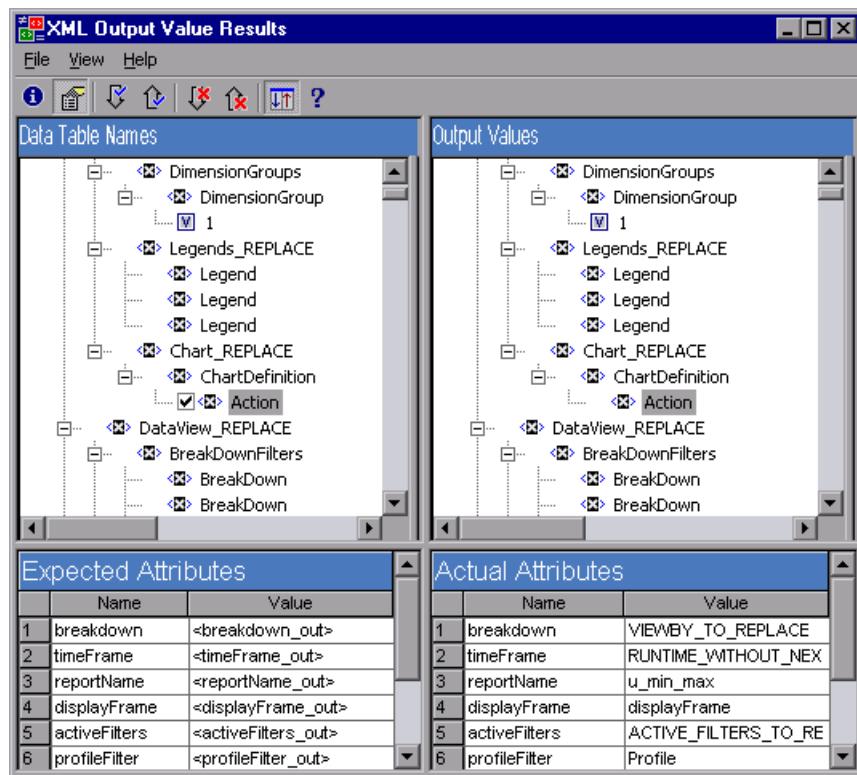
This window also displays:

- **Additional Details pane.** Displays results information for the selected item. (Available only if the **Output Value Summary** option is selected.)
- **Expected Attributes pane.** Displays each attribute name and its expected value or output value name. (Available only if the **Attribute Details** option is selected.)
- **Actual Attributes pane.** Displays the attribute name and the actual value of each attribute during the run session. (Available only if the **Attribute Details** option is selected.)

Example of XML Output Value Results window with Additional Details pane:



Example of XML Output Value Results window with Expected Attributes and Actual Attributes panes:



To access	In the Captured Data pane of the Run Results Viewer, click the View XML Output Value Results button.
See also	"XML Output Value Results" on page 1187

User interface elements are described below:

UI Elements	Description
	View Output Value Summary. Displays the Output Value Summary pane, which provides information regarding the output value for the element, attribute, or value currently selected in the XML tree. Menu option: View > Output Value Summary.
	View Attribute Details. Displays the Expected Attributes and Actual Attributes panes containing the details of the attributes' output value for the selected element in the XML tree. Menu option: View > Attribute Details.
	Find Next Output Value. Jumps directly to the next output value in the XML Tree. Menu option: View > Find Next Output Value
	Find Previous Output Value. Jumps directly to the previous output value in the XML Tree. Menu option: View > Find Previous Output Value
	Find Next Error. Jumps directly to the next error in the XML Tree. Menu option: View > Find Next Error
	Find Previous Error. Jumps directly to the previous error in the XML Tree. Menu option: View > Find Previous Error
	Scroll Trees Simultaneously. Synchronizes the scrolling of the Data Table Names and Output Values trees. If this option is selected, the Data Table Names and Output Values trees scroll simultaneously as you navigate through either of the tree structures. If this option is not selected, you can scroll only one tree at a time. Menu option: View > Scroll Trees Simultaneously
	Help Topics. Opens the help for the XML Output Value Results window. Menu option: View > Help Topics

Part VII

Maintaining and Debugging Tests

33

Debugging Tests and Function Libraries

This chapter includes:

Concepts

- Debugging Overview on page 1206
- Considerations for Debugging on page 1207
- Debug Session Speed on page 1208
- Single Step Commands on page 1208
- Running to a Step and Debugging from a Step on page 1209
- Modifying and Watching the Values of Variables and Properties of Objects During a Run Session on page 1212
- Breakpoints on page 1213
- Run Errors on page 1214

Tasks

- How to Debug Your Test or Function Library on page 1215
- How to Use Breakpoints on page 1219
- How to Debug an Action or a Function - Exercise on page 1220
- How to Step Into, Out of, or Over a Specific Step - Exercise on page 1224

Reference

- Debug Viewer Pane on page 1227
- Watch Tab (Debug Viewer Pane) on page 1229
- Variables Tab (Debug Viewer Pane) on page 1232
- Command Tab (Debug Viewer Pane) on page 1234

- Run Error Message Box on page 1236

[Troubleshooting and Limitations - Debugging](#) on page 1238

Concepts

Debugging Overview

After you create a test or function library (including registered user functions), you should check that they run smoothly, without errors in syntax or logic. To debug a function library, you must first associate it with a test and then debug it from that test.

By controlling and debugging your run sessions, you can identify and handle problems in your tests, function libraries, and registered user functions.

QuickTest provides different options that you can use to detect and isolate defects in a test or function library. For example:

- You can control the run session using the **Pause** command, breakpoints, and various step commands that enable you to step into, over, and out of a specific step.
- If QuickTest displays a run error message during a run session, you can click the **Debug** button on the error message to suspend the run and debug the test or function library.
- When a run session is paused (suspended), you can use the Debug Viewer to check and modify the values of VBScript objects and variables and to manually run VBScript commands.
- You can use the **Debug from Step** or **Debug from Action** command to begin (and pause) your debug session at a specific point in your test. You can also use the **Run to Step** or **Run to Action** command to pause the run at a specific point in your test. You can set breakpoints, and then enable and disable them as you debug different parts of your test or function library.

- You can also use the **Run from Step** or **Run from Action** command to run your test from a selected step or action. This enables you to check a specific section of your application or to confirm that a certain part of your test or function library runs smoothly. For more information, see "Run Sessions - Overview" on page 1064.

Considerations for Debugging

- While the test and function libraries are running in debug mode, they are read-only. You can modify the content after you stop the debug session (not when you pause it). If needed, you can enable the function library for editing (**File > Enable Editing**) after you stop the session. For more information, see "How to Manage Function Libraries" on page 1028. After you implement your changes, you can continue debugging your test and function libraries.
- If you perform a file operation (for example, you open a different test or create a new test), the debug session stops.
- If a file is called using an `ExecuteFile` statement, you cannot debug the file or any of the functions contained in the file. In addition, when debugging a test that contains an `ExecuteFile` statement, the execution marker may not be displayed correctly.
- In QuickTest, when you open a test, QuickTest creates a local copy of the external resources that are saved to your Quality Center project. Therefore, any changes you apply to any external resource that is saved in your Quality Center project, such as a function library, will not be recognized in the test until the test is closed and reopened. (An external resource is any resource that can be saved separately from the test, such as a function library, a shared object repository, or a recovery scenario.)
In contrast with this, any changes you apply to external resources saved in the file system, such as function libraries, are implemented immediately, as these files are accessed directly and are not saved as local copies when you open your test.

Debug Session Speed

During a run session, QuickTest normally runs steps quickly. While you are debugging a test or function library, you may want QuickTest to run the steps more slowly so you can pause the run when needed or perform another task.

For task details, see "Slow your debugging session" on page 1216.

Single Step Commands

You can run a single step of a test or function library using the **Step Into**, **Step Out**, and **Step Over** commands.

Tip: To display the Debug toolbar, select **View > Toolbars > Debug**.

Step Into

Step Into runs only the current step in the active test or function library. If the current step calls another action or a function, the called action or function is displayed in the QuickTest window, and the test or function library pauses at the first line of the called action or function.

Step Out

After using **Step Into** to enter a function in a function library, you can use the **Step Out** command. **Step Out** continues the run to the end of the function, returns to the calling test or function library, and then pauses the run session at the next line (if one exists).

Step Over

Step Over runs only the current step in the active test or function library.

If the current step calls a user-defined function, the called function is executed in its entirety, but the called function script is not displayed in the QuickTest window. The run session then returns to the calling test or function library and pauses at the next line (if one exists).

If the current step calls another action, the called action is displayed in the QuickTest window, and the run session pauses at the first line of the called action (like **Step Into**).

For task details, see "Step into, out of, or over a specific step during a debug session" on page 1216.



Running to a Step and Debugging from a Step

You can use the **Run to Step**, **Run to Action**, **Debug from Step**, and **Debug from Action** commands to instruct QuickTest to run a test or action (including any associated function library) until it reaches a particular step or action, or to begin debugging from a specific step or action. You can also use the **Run from Step** or **Run from Action** command to start or continue a run from a particular step or action.

The **Run to Step** and **Debug from Step** commands are available in the **Debug** menu, and when you right-click a step in your action.

The **Run from Step** command is available from the **Automation** menu, and when you right-click a step in your action.

The **Run to Action**, **Debug from Action**, and **Run from Action** commands are available when you right-click an action in the Test Flow pane.

Run to Step

You can instruct QuickTest to run from the beginning of the test or action (Expert View only)—or from the current location in the test or action—and to stop at a particular step. This is similar to adding a temporary breakpoint to a step. For example, if you want to begin debugging your test or action from a particular step, you may want to run your test or action to that step, as this opens your application to the relevant location.

Debug from Step

You can instruct QuickTest to begin your debug session from a particular step instead of beginning the run at the start of the test or action. Before you start debugging from a specific step, make sure that the application is open to the location where you want to start debugging. You can begin debugging from a specific step in your test or action when editing a test or action.

For task details, see "Start or pause your debugging session at a specific point in your test" on page 1216.

Run from Step

You can instruct QuickTest to run your action from a particular step instead of from the beginning of the test or action. Before you start running from a specific step, make sure that the application is open to the location where you want the run to begin.

- In the Expert View, the **Run from Step** option runs your test from the selected step until the end of the action (or until it reaches a breakpoint). Using **Run from Step** in this mode ignores any iterations. However, if the action contains nested actions, QuickTest runs the nested actions for the defined number of iterations of the nested action.
- In the Keyword View, if one action is displayed, the **Run from Step** option runs your action from the selected step until the end of the action (or until it reaches a breakpoint). If the test flow is displayed, the **Run from Step** option runs the test from the selected step until the end of the test (or until it reaches a breakpoint), as long as all of the actions are internal actions that are local to your test.

Iterations are handled as follows:

- If you select an Action node, the **Run from Step** option runs only one iteration of the test, from the selected step until the end of the test. Within the part of the test that runs, QuickTest runs each action for the number of iterations defined for that action.
- If you select a step inside an action, the **Run from Step** option runs only one iteration of the action, from the selected step until the end of the action. Within the part of the action that runs, QuickTest runs any nested actions for the number of iterations defined.

Note: If your test contains a call to an external action, the run session stops when that action is reached. Similarly, if you use the **Run from Step** option from within an external action, the run stops at the end of that action (or when a breakpoint is reached).

Example:

After you debug a part of your action, you may want to skip over a set of steps that you know work correctly, and then continue the debug session from a later step. You can do this by inserting a breakpoint in the step where you want to continue debugging, and then using the **Run from Step** option to run the action from the step at which you stopped debugging until it reaches the breakpoint.

Alternatively, you can use the **Run from Step** option to continue the run session from a particular step to the end of the test or action instead of stopping at a breakpoint.

Run to Action

You can instruct QuickTest to run from the beginning of the test until the beginning of the selected action and then pause the run session. For example, if you want to begin debugging your test from a particular action, you may want to run your test until that action, as this opens your application to the relevant location.

Debug from Action

You can instruct QuickTest to begin a debug session, and pause it, at the beginning of the selected action.

Run from Action

You can instruct QuickTest to start a run session from the beginning of the selected action.

Modifying and Watching the Values of Variables and Properties of Objects During a Run Session

You can use the Watch tab and Variables tab to view the current value of different VBScript expressions, variables, and object properties in a suspended run session of your test or function library. A run session is suspended, for example, if you use the **Debug > Pause** command, or when the test or function library stops at breakpoint.

The Variables tab displays the current values and types of all variables in the main script of the current action, or in a selected function in your test or function library, and enables you to modify their values.

The Watch tab displays the current values and the types of VBScript expressions that you add to the tab.

As you continue stepping through the subsequent steps in your test or function library, QuickTest automatically updates the Watch tab and Variables tab with the current value for any variable or expression whose value changes.

You can also change the value of a variable or property manually in these tabs. For example, for objects that support the **Object** property, you can edit the value of a run-time object property displayed in the Watch tab, thereby changing the value of the property in the application you are testing before you resume the run session.

You can add any of the following types of expressions to the Watch tab:

- The name of a test object
- The name of a variable
- The name of a property
- Any other type of VBScript expression

Caution: QuickTest runs the expressions in the Watch tab to evaluate them. Therefore, do not add a test object method or any expression whose evaluation could affect the state of the test object, as this can lead to unexpected behavior of your test or function library.

For task details, see "Check and modify the values of variables and VBScript expressions during a debug session" on page 1217.



Breakpoints

You can use breakpoints to instruct QuickTest to pause a run session at a predetermined place in a test or function library. QuickTest pauses the run when it reaches the breakpoint, before executing the step. You can then examine the effects of the run up to the breakpoint, make any necessary changes, and continue running the test or function library from the breakpoint. Breakpoints are applicable only to the current QuickTest session and are not saved with your test or function library.

You can use breakpoints to:

- Suspend a run session and inspect the state of your application
- Mark a point from which to begin stepping through a test or function library using the step commands

By setting a breakpoint, you can pause a run session at a predetermined place in a test or function library. A breakpoint is indicated by a filled red circle icon in the left margin adjacent to the selected step.

You can instruct QuickTest to ignore an existing breakpoint during a debug session by temporarily disabling the breakpoint. Then, when you run your test or function library, QuickTest runs the step containing the breakpoint, instead of stopping at it. When you enable the breakpoint again, QuickTest pauses there during the next run. This is particularly useful if your test or function library contains many steps, and you want to debug a specific part of it.

You can enable or disable breakpoints individually or all at once. For example, suppose you add breakpoints to various steps throughout your test or function library, but for now, you want to debug only a specific part of your testing document. You could disable all breakpoints in your test or function library, and then enable breakpoints only for specific steps. After you finish debugging that section of your document, you could disable the enabled breakpoints, and then enable the next set of breakpoints (in the section you want to debug). Because the breakpoints are disabled and not removed, you can find and enable any breakpoint, as needed.

Enabled breakpoint. An enabled breakpoint is indicated by a filled red circle icon in the left margin  adjacent to the selected step.

Disabled breakpoint. A disabled breakpoint is indicated by an empty circle icon in the left margin  adjacent to the selected step.

For task details, see "Use breakpoints in your test" on page 1217.

Run Errors

There are two types of Run Error message boxes that can be displayed during a run session:

- ▶ A pure VBScript syntax error. When a syntax run error message box is displayed, click **OK** in the message box and address the error in your step.
- ▶ The other message box can be displayed in a number of situations. It offers information about the error and a number of buttons for dealing with errors encountered.

For user interface details, see "Run Error Message Box" on page 1236.

Tasks

How to Debug Your Test or Function Library

This task describes different ways you can control and debug your run sessions so you can identify and handle problems in your tests, function libraries, and registered user functions.

To practice this task, see "How to Debug an Action or a Function - Exercise" on page 1220.

Tip: You can use the Screen Recorder to capture a movie of your application as it is being tested. For more information, see "Screen Recorder Pane (Run Results Viewer)" on page 1134.

This task contains the following sections:

- "Prerequisites" on page 1216
- "Slow your debugging session" on page 1216
- "Step into, out of, or over a specific step during a debug session" on page 1216
- "Start or pause your debugging session at a specific point in your test" on page 1216
- "Use breakpoints in your test" on page 1217
- "Handle run errors" on page 1217
- "Check and modify the values of variables and VBScript expressions during a debug session" on page 1217
- "Manually run VBScript commands during a debug session" on page 1218

Prerequisites

You must have the Microsoft Script Debugger installed to run tests in debug mode. If it is not installed, you can use the QuickTest Additional Installation Requirements Utility to install it. (Select **Start > Programs > QuickTest Professional > Tools > Additional Installation Requirements.**)

Slow your debugging session

In the Run pane of the Options dialog box (**Tools > Options > Run** node), specify the time (in milliseconds) QuickTest pauses between each step by modifying the **Delay each step execution by** option. For more information on the Run pane options, see "Run Pane (Options Dialog Box)" on page 1447.

Step into, out of, or over a specific step during a debug session



➤ To use the **Step Into** command select **Debug > Step Into**, click the **Step Into** button, or press F11.



➤ To use the **Step Out** command select **Debug > Step Out**, click the **Step Out** button, or press Shift+F11.



➤ To use the **Step Over** command select **Debug > Step Over**, click the **Step Over** button, or press F10.

For details, see "Single Step Commands" on page 1208.

For an example of using single step commands, see "How to Step Into, Out of, or Over a Specific Step - Exercise" on page 1224.

Start or pause your debugging session at a specific point in your test

To instruct QuickTest to run to a particular step, do one of the following:

- In the test, right-click in the step in which you want QuickTest to stop the run and select **Run to Step** from the context menu.
- In the Test Flow pane, right-click the action at which you want QuickTest to stop the run and select **Run to Action** from the context menu. This instructs QuickTest to stop the run at the first step in that action.

To instruct QuickTest to run from a particular step, do one of the following:

- Right-click in the step where you want QuickTest to start the run and select **Debug from Step** from the context menu.
- In the Test Flow pane, right-click the action where you want QuickTest to start the run and select **Debug from Action** from the context menu. This instructs QuickTest to begin the run at the first step in that action.

To temporarily pause or resume a run session:

- To temporarily suspend a run session, click the **Pause** button. A paused test or function library stops running when all previously interpreted steps have been run.
- To resume running a paused run, click the **Run** button. The run continues from the point it was suspended.

Use breakpoints in your test

For details, see "How to Use Breakpoints" on page 1219.

Handle run errors

For details, see "Run Error Message Box" on page 1236.

Check and modify the values of variables and VBScript expressions during a debug session

- To add an expression to the Watch tab, right-click the expression and select **Add to Watch** from the context menu.
- To remove an expression from the Watch tab, select the row in the Watch tab that you want to remove and press the Delete key on your keyboard.
- To view the current values for all variables up to the current step in the test or function library, use the Variables tab.

You can also change the value of a variable or property manually in these tabs.

For more details, see:

- "Debug Viewer Pane" on page 1227
 - "Watch Tab (Debug Viewer Pane)" on page 1229
 - "Variables Tab (Debug Viewer Pane)" on page 1232
-

Note:

- To add an **identification property** to the Watch tab, you must use an expression that calls **GetROProperty**. This enables you to watch the run-time value of the object's identification property. For example, to watch the value currently displayed in the Calculator application, you can add the expression:
`Window("Calculator").WinEdit("Edit").GetROProperty("text")`
 - You cannot modify the run-time value of an object's identification property from the Watch tab.
 - You can add an expression to the Watch tab from the Expert View or from a function library.
-

Manually run VBScript commands during a debug session

For details, see "Command Tab (Debug Viewer Pane)" on page 1234.

How to Use Breakpoints

The following steps describe how to set breakpoints, and temporarily enable or disable them. After you finish using them, you can remove them from your test or function library.

This task includes the following sections:

- "Set a breakpoint" on page 1219
- "Enable or disable a breakpoint" on page 1219
- "Enable or disable all breakpoints" on page 1219
- "Remove a single breakpoint or all breakpoints" on page 1220

Set a breakpoint

To set a breakpoint click in the left margin of a step in the test or function library where you want the run to stop.

The breakpoint symbol  is displayed in the left margin adjacent to the selected step.

Enable or disable a breakpoint

To enable/disable a specific breakpoint, click in the step containing the breakpoint you want to disable/enable and select **Debug > Enable/Disable Breakpoint**. The breakpoint is either disabled or enabled (depending on its previous state).

Enable or disable all breakpoints



To enable/disable all breakpoints select **Debug > Enable/Disable All Breakpoints** or click the **Enable/Disable All Breakpoints** button. If at least one breakpoint is enabled, QuickTest disables all breakpoints in the test or function library. Alternatively, if all breakpoints are disabled, QuickTest enables them.

Remove a single breakpoint or all breakpoints

To remove a single breakpoint click the breakpoint icon in the left margin of the step. The breakpoint symbol is removed from the left margin of the testing document.



To remove all breakpoints click the **Clear All Breakpoints** button, or select **Debug > Clear All Breakpoints**. All breakpoint symbols are removed from the left margin of the testing document.

How to Debug an Action or a Function - Exercise

In this exercise, you create and debug an action or a function, to practice using some of QuickTest's debugging capabilities.

Suppose you create an action or a function that defines variables that will be used in other parts of your test or function library. You can add breakpoints to the action or function to see how the value of the variables changes as the test or function library runs. To see how the test or function library handles the new value, you can also change the value of one of the variables during a breakpoint.

Note: For a task related to this exercise, see "How to Debug Your Test or Function Library" on page 1215.

This exercise includes the following steps:

- "Create a New Action or Function" on page 1221
- "(For Function Libraries Only) Associate the Function Library with a Test" on page 1222
- "(For Function Libraries Only) Add a Call to the Function in Your Test" on page 1222
- "Add Breakpoints" on page 1222
- "Begin Running the Test" on page 1222

- "Check the Value of the Variables in the Debug Viewer Pane" on page 1222
- "Check the Value of the Variables at the Next Breakpoint" on page 1223
- "Modify the Value of a Variable Using the Variables Tab" on page 1223
- "Modify the Value of a Variable Using the Command Tab" on page 1223
- "Repeat a Command from the Command History" on page 1224

1 Create a New Action or Function

Open a test and insert a new action, or open a new function library and create a new function called **SetVariables**. For more information on inserting actions, see Chapter 14, "Actions." For more information on working with functions, see Chapter 29, "User-Defined Functions and Function Libraries."

In the Expert View or function library, enter the VBScript code, as follows:

Expert View <pre>Dim a a="hello" b="me" MsgBox a</pre>	Function Library <pre>Function SetVariables Dim a a="hello" b="me" MsgBox a EndFunction</pre>
--	---

For more information on the Expert View, see Chapter 27, "Working in the Expert View and Function Library Windows."

Note: If you are working in the Expert View, skip to Step 4. If you are working in a function library, continue with Step 2 and Step 3.

2 (For Function Libraries Only) Associate the Function Library with a Test

- a Make sure the function library is in focus.
- b Select **File > Associate Library '<Function Library Name>' with '<Test Name>'**. QuickTest associates the function library with your test.

3 (For Function Libraries Only) Add a Call to the Function in Your Test

Add a call to the function by inserting a new step and typing the following in the Expert View:

SetVariables

4 Add Breakpoints

Add breakpoints at the lines containing the text `b="me"` and `MsgBox a`. For more information on adding breakpoints, see "Breakpoints" on page 1213.

5 Begin Running the Test

Run the test. The test or function library stops at the first breakpoint, before executing that step (line of script).

6 Check the Value of the Variables in the Debug Viewer Pane

- a Select **View > Debug Viewer** to open the Debug Viewer pane, if it is not already open. Then click the **Watch** tab on the Debug Viewer pane.
- b In the document pane, select the variable **a** and select **Debug > Add to Watch**. QuickTest adds the variable **a** to the Watch tab. The **Value** column indicates that the value of **a** is currently "`hello`", because the breakpoint stopped after the value of variable **a** was initiated. The **Type** column indicates that **a** is a **String** variable.
- c In the document pane, select the variable **b** and select **Debug > Add to Watch**. QuickTest adds the variable **b** to the Watch tab. The **Value** column indicates `<Variable is undefined: 'b'>` (and the **Type** column displays **Error**), because the test stopped before variable **b** was declared.

- d Click the **Variables** tab in the Debug Viewer pane. If you are working with a test, only variable **a** is displayed (with the value "hello"), because **a** is the only variable that was initiated up to this point. If you are working with a function library, both **SetVariables** (with the value **Empty**) and variable **a** (with the value "hello") are displayed. Variable **b** is not displayed because the test stopped before variable **b** was declared.

7 Check the Value of the Variables at the Next Breakpoint

Click the **Run** button to continue running the test. The test stops at the next breakpoint. Note that the values of variables **a** and **b** were both updated in the Watch and Variables tabs.

8 Modify the Value of a Variable Using the Variables Tab

- a Click the **Variables** tab in the Debug Viewer pane.
- b In the **Value** column, select the string "me", replace it with the string "you", and press ENTER on the keyboard.
- c Click the **Watch** tab. You can see that the value of variable **b** was also updated in the Watch tab.

9 Modify the Value of a Variable Using the Command Tab

- a Click the **Command** tab in the Debug Viewer pane.
- b At the command prompt, type:
if b="me" then a="b is me" else a="b is you" end if
Then press ENTER on the keyboard.
- c Click the **Variables** tab to verify that the value of variable **a** was updated according to the command you entered and now displays the value: "b is you"
- d Click the **Run** button to continue running the test. The message box that opens displays "b is you" (which is the modified value of **a**). This indicates that you successfully modified the values of both **a** and **b** using the Debug Viewer pane.
- e Click **OK** to close the message box.

10 Repeat a Command from the Command History

- a** Remove the first breakpoint and run the test again. When the test stops at the breakpoint (before displaying the message box), modify the value of variable **b** in the Variables tab to "not me".
- b** Select the **Command** tab and press the UP arrow key on your keyboard. QuickTest copies the command that you typed in the previous test run (if **b="me"** then **a="b** is me" else **a="b** is you" end if) to the active command line. Press ENTER to run the command, and then click the **Run** button to complete the test run.

How to Step Into, Out of, or Over a Specific Step - Exercise

In this exercise, you create a sample function library and run it (from a test) using the **Step Into**, **Step Out**, and **Step Over** commands.

Note: For a task related to this exercise, see "How to Debug Your Test or Function Library" on page 1215.

This exercise includes the following steps:

- "Create the sample function library and test" on page 1224
- "Run the function library from your test and use the Step Into, Step Out, and Step Over commands" on page 1225

1 Create the sample function library and test

- a** Select **File > New > Function Library** to open a new function library.
- b** In the function library, enter the following lines exactly:

```
public Function myfunc()
msgbox "one"
msgbox "two"
msgbox "three"
End Function
```

- c Save the function library to the file system or your Quality Center project with the name **SampleFL.qfl**. (For more information, see "How to Manage Function Libraries" on page 1028.)
- d Select **File > New > Test** to open a new test.
- e Click the tab for the **SampleFL.qfl** function library to bring it into focus.
- f Select **File > Associate Library 'SampleFL.qfl' with 'Test'** to associate the function library with your test.
- g Click the tab for the test you created to bring it into focus. Click the **Expert View** tab to display the Expert View and enter the following lines exactly:
myfunc
myfunc
myfunc
endOfTest="true"

2 Run the function library from your test and use the Step Into, Step Out, and Step Over commands

- a Add a breakpoint on the first step of the test (the first call to the myfunc function) by pressing F9 (**Insert/Remove Breakpoint**). The breakpoint symbol is displayed in the left margin . For more information, see "Breakpoints" on page 1213.
- b Run the test. The test pauses at the breakpoint.
- c Press F11 (**Step Into**). The execution arrow points to the first line (msgbox "one") of the function in the function library.
- d Press F11 (**Step Into**) again. A message box displays the text one.
- e Click **OK** to close the message box. The execution arrow moves to the next line in the function.
- f Continue pressing F11 (**Step Into**) (and pressing **OK** on the message boxes that open) until the execution arrow leaves the function and is pointing to the second step in the test (the second call to the myfunc function).
- g Press F11 (**Step Into**) to enter the function again. The execution arrow points to the first msgbox line within the function.

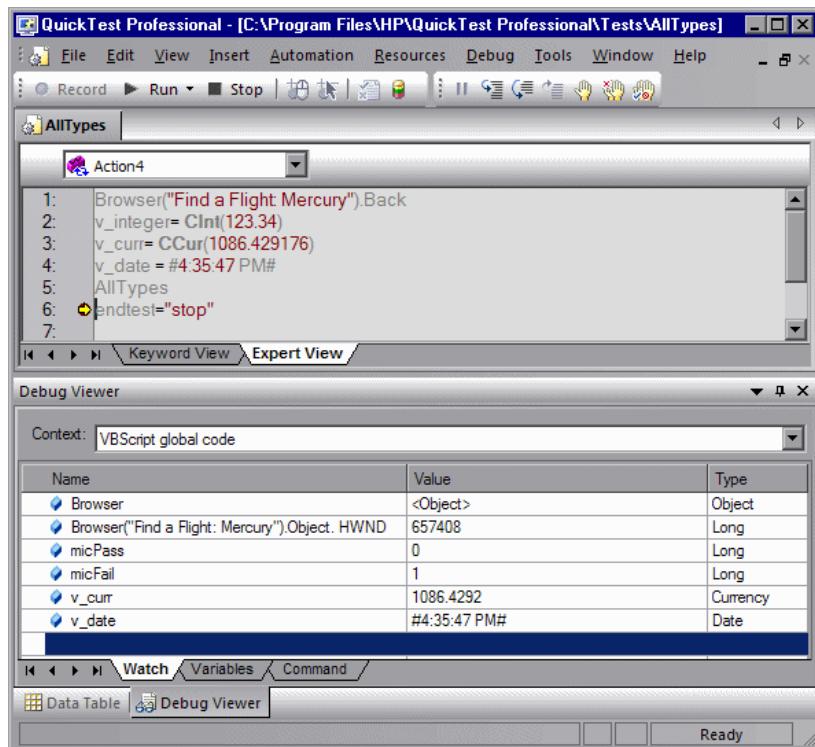
- h Press SHIFT+F11 (**Step Out**). Close each of the message boxes that opens. Notice that the execution arrow continues to point to the first line in the function until you close the last of the three message boxes. After you close the third message box, the execution arrow points to the next line in the test (the third call to the myfunc function).
- i Press F10 (**Step Over**). The three message boxes open again—this time, in the Keyword View. The execution arrow remains on the same step in the test until you close the last of the three message boxes. After you close the third message box, the execution arrow points to the next step in the test.

Reference

Debug Viewer Pane

This pane enables you to perform one of the following activities when a run session is suspended:

- View, set, or modify the current value of objects or variables in your test or function library.
- Run VBScript commands in your paused run session.



To access	View > Debug Viewer
Important information	A run session can be suspended in the following situations: <ul style="list-style-type: none"> ➤ The run session stops at a breakpoint. ➤ You use Debug menu commands or toolbar buttons (such as Pause or Run to Step) to suspend the run session. ➤ A step fails and you select the Debug option.
Relevant tasks	"How to Debug Your Test or Function Library" on page 1215
See also	<ul style="list-style-type: none"> ➤ "Debugging Overview" on page 1206 ➤ "Considerations for Debugging" on page 1207 ➤ "Modifying and Watching the Values of Variables and Properties of Objects During a Run Session" on page 1212 ➤ "How to Debug an Action or a Function - Exercise" on page 1220

User interface elements are described below:

UI Elements	Description
Watch tab	Displays the current values and types of variables and VBScript expressions that you add to the Watch tab, and enables you to modify the values of displayed variables and properties. For more information, see "Watch Tab (Debug Viewer Pane)" on page 1229.
Variables tab	Displays the current values and types of all variables in the main script of the current action, or in a selected subroutine, and enables you to modify their values. For more information, see "Variables Tab (Debug Viewer Pane)" on page 1232.
Command tab	Enables you to run VBScript commands in your paused run session. For more information, see "Command Tab (Debug Viewer Pane)" on page 1234.

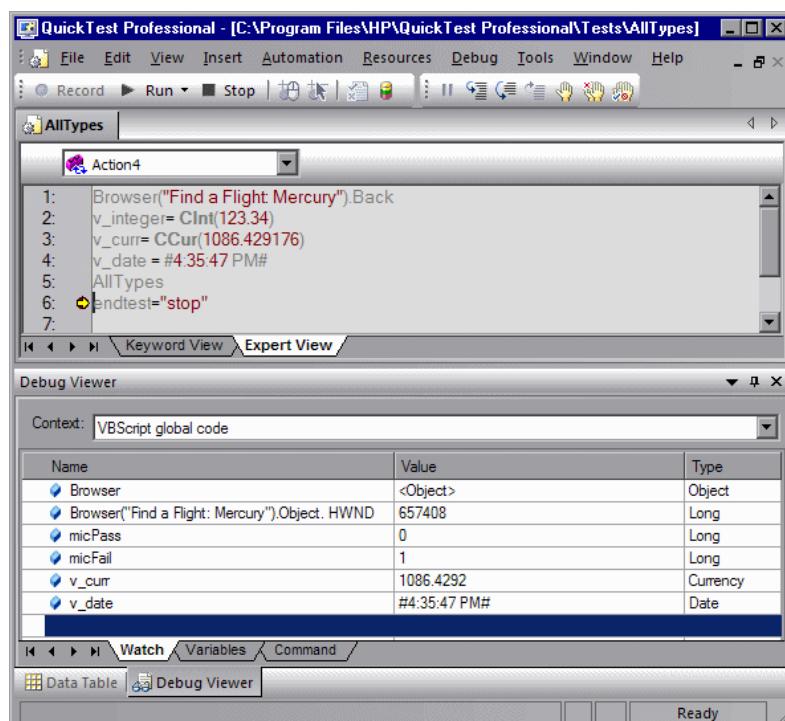
Watch Tab (Debug Viewer Pane)

This tab enables you to view the current values and types of selected variables, properties, and VBScript expressions in your test or function library.

You can also use this tab to manually change the value of a variable or property.

This image shows a run session that was suspended before running a test step. The **Context** box therefore contains the string VBScript global code and the values displayed in the Watch tab were evaluated within the context of the suspended action.

You can see some of the types of expressions that can be displayed in the Watch tab (for example, the **HWND** native property of the **Find a Flight: Mercury** Browser object). For additional types and contexts, see the image shown in "Variables Tab (Debug Viewer Pane)" on page 1232.



To access	View menu > Debug Viewer item > Watch tab
Relevant tasks	"How to Debug Your Test or Function Library" on page 1215
See also	<ul style="list-style-type: none"> ➤ "Debugging Overview" on page 1206 ➤ "Considerations for Debugging" on page 1207 ➤ "Modifying and Watching the Values of Variables and Properties of Objects During a Run Session" on page 1212 ➤ "How to Debug an Action or a Function - Exercise" on page 1220

User interface elements are described below:

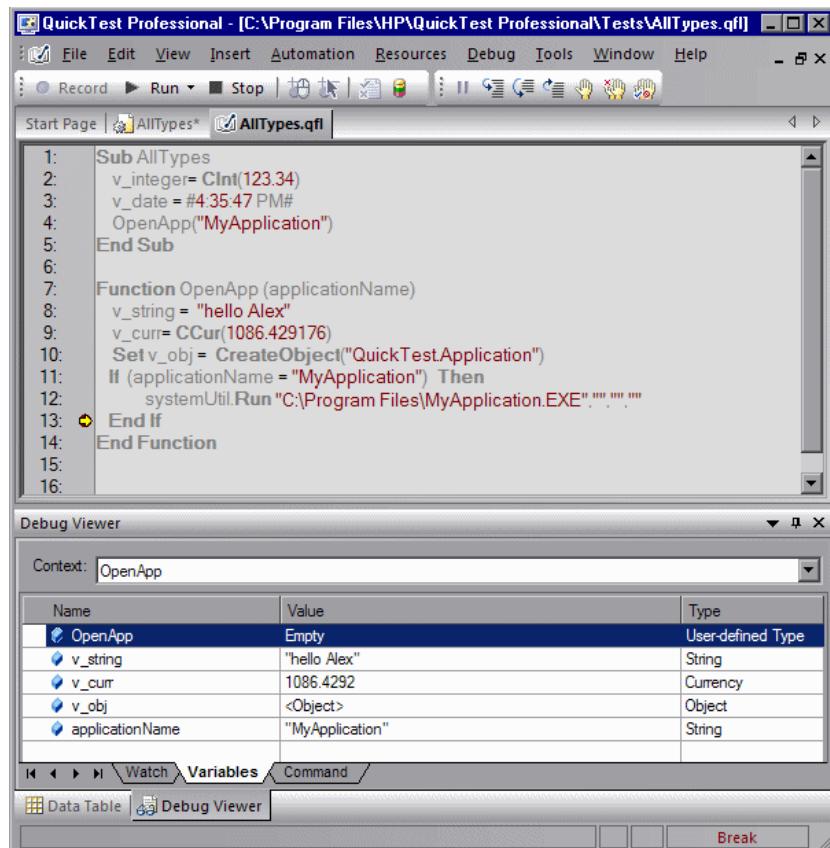
UI Elements	Description
Context	<p>The context in which the expressions displayed in the Watch tab are evaluated.</p> <ul style="list-style-type: none"> ➤ If the run session was suspended before running a test step, the Context box contains the string VBScript global code and the expressions displayed in the Watch tab are evaluated within the context of the suspended action. ➤ If the run session was suspended within a function library, the Context box initially displays the name of the function in which the run paused and enables you to switch to the context of other functions and subroutines within the same function library. <p>The expressions displayed in the Watch tab are evaluated within the context of the selected function or subroutine.</p>

UI Elements	Description
Name	<p>The VBScript expression whose value you want to watch. For information on adding and removing expressions from the Watch tab, see "Modifying and Watching the Values of Variables and Properties of Objects During a Run Session" on page 1212.</p> <p>Caution: QuickTest runs the expressions in the Watch tab to evaluate them. Therefore, do not enter a test object method or any expression whose evaluation could affect the state of the test object, as this can lead to unexpected behavior of your test or function library.</p>
Value	<p>The current value of the expression. The evaluated value is displayed only when a run session is suspended.</p> <p>In this column, you can also set or modify the value of a variable or property that is being watched.</p> <p>For example, you can edit the value of a run-time object property displayed in the Watch tab, thereby changing the value of the property in the application you are testing before you resume the run session. (Relevant only for objects that support the Object property.)</p> <p>You cannot modify the run-time value of an object's identification property from the Watch tab.</p>
Type	<p>The type of the expression's value after it is evaluated (for example, Integer or String).</p> <p>If an expression cannot be evaluated in the current context, the type displayed is Error (indicated also by an icon in the Name column).</p>

Variables Tab (Debug Viewer Pane)

This pane displays the current values and types of all variables in the main script of the current action, or in a selected function in your test or function library, and enables you to modify their values.

This image shows a run session that was suspended within a function in a function library. The Variables tab therefore displays only the variables that are defined within the context of the suspended function.



The screenshot shows the QuickTest Professional interface with the title bar "QuickTest Professional - [C:\Program Files\HP\QuickTest Professional\Tests\AllTypes.qfl]". The menu bar includes File, Edit, View, Insert, Automation, Resources, Debug, Tools, Window, Help. The toolbar has icons for Record, Run, Stop, and various debug functions. The main script editor window shows the following VBA code:

```

1: Sub AllTypes
2:   v_integer= CInt(123.34)
3:   v_date = #4.35.47 PM#
4:   OpenApp("MyApplication")
5: End Sub
6:
7: Function OpenApp(applicationName)
8:   v_string = "hello Alex"
9:   v_curr= CCur(1086.429176)
10:  Set v_obj= CreateObject("QuickTest.Application")
11:  If (applicationName = "MyApplication") Then
12:    systemUtil.Run "C:\Program Files\MyApplication.EXE", "", ""
13:  End If
14: End Function
15:
16:

```

The "AllTypes" function is currently executing, indicated by the yellow dot on line 13. The "Context" dropdown in the Debug Viewer pane is set to "OpenApp". The Variables table shows the following data:

Name	Value	Type
OpenApp	Empty	User-defined Type
v_string	"hello Alex"	String
v_curr	1086.4292	Currency
v_obj	<Object>	Object
applicationName	"MyApplication"	String

The "Variables" tab is selected in the Debug Viewer pane. Other tabs include Watch, Command, Data Table, and Break.

To access	View menu > Debug Viewer item > Variables tab
Important information	Only variables that were recognized up to the last step that was performed are displayed in the Variables tab. As you continue stepping through the subsequent steps in your test or function library, QuickTest adds any additional variables that it recognizes and updates the values displayed in the Variables tab.
Relevant tasks	"How to Debug Your Test or Function Library" on page 1215
See also	<ul style="list-style-type: none"> ➤ "Debugging Overview" on page 1206 ➤ "Considerations for Debugging" on page 1207 ➤ "Modifying and Watching the Values of Variables and Properties of Objects During a Run Session" on page 1212 ➤ "How to Debug an Action or a Function - Exercise" on page 1220

User interface elements are described below:

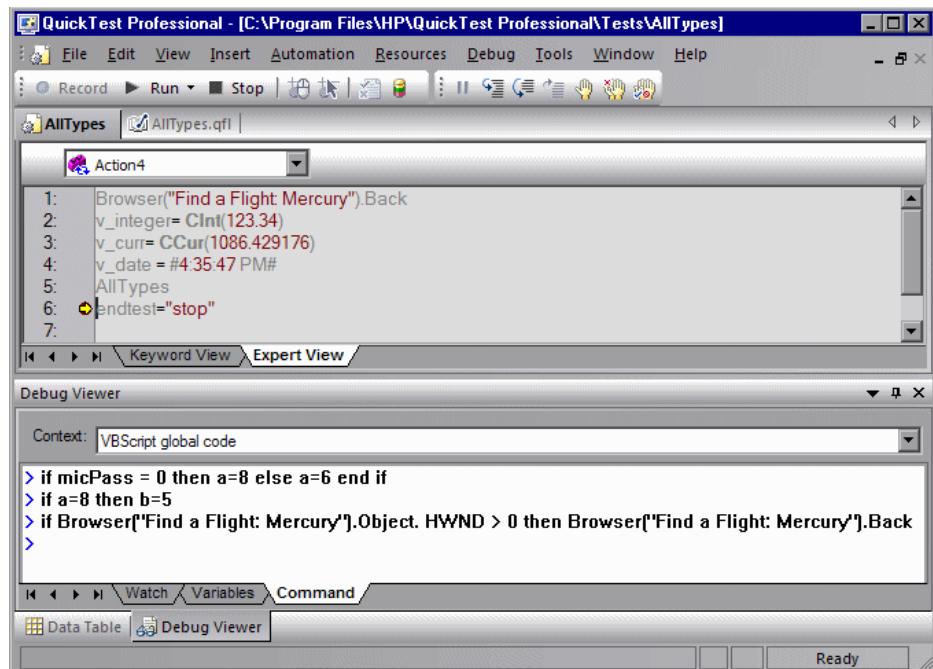
UI Elements	Description
Context	<p>The context of the variables displayed in this tab.</p> <ul style="list-style-type: none"> ➤ If the run session was suspended before running a test step, the Context box contains the string VBScript global code and this tab displays only variables that are defined within the context of the suspended action. ➤ If the run session was suspended within a function library, the Context box initially displays the name of the function in which the run paused and enables you to switch to the context of other functions and subroutines within the same function library. <p>Only variables that are defined within the context of the selected function or subroutine are displayed in the Variables tab.</p>
Name	The name of the variable.
Value	The current value of the variable. You can edit this value to set or modify the value of the variable before you continue the run session.
Type	The type of the variable's value (for example, Integer or String).

Command Tab (Debug Viewer Pane)

This pane to run lines of VBScript code in your test or function library.

For example, you can run VBScript code that performs any of the following activities before you resume the run session:

- ▶ Retrieves information from the application you are testing
- ▶ Runs a test object method and displays the return value, enabling you to learn more about how the method works
- ▶ Modifies the value of a native (run-time object) property in the application
- ▶ Calls a native (run-time object) method in the application



To access	View menu > Debug Viewer item > Command tab
Important information	<ul style="list-style-type: none"> ➤ You can enter lines of code in the Command tab only when a run session is suspended. When no run session is suspended, you can view the command history, select and copy text from it, or use the Clear All context menu command. ➤ Running RunAction or LoadAndRunAction statements from this tab is not supported.
Relevant tasks	"How to Debug Your Test or Function Library" on page 1215
See also	<ul style="list-style-type: none"> ➤ "Debugging Overview" on page 1206 ➤ "Considerations for Debugging" on page 1207 ➤ "Modifying and Watching the Values of Variables and Properties of Objects During a Run Session" on page 1212 ➤ "How to Debug an Action or a Function - Exercise" on page 1220

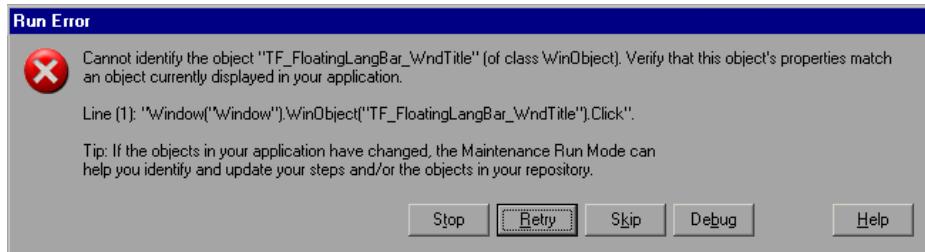
User interface elements are described below (unlabeled elements are shown in angle brackets):

UI Elements	Description
Context	The context of the expressions and variables displayed in the Watch and Variable tabs.
<command line prompt>	Enables you to run a line of VBScript code in the context of your suspended run session. Type or paste the line of code at the prompt and press enter to run the code.
<command line history>	<p>Displays the lines of VBScript code that you ran.</p> <ul style="list-style-type: none"> ➤ You cannot make any changes to these lines, but you can select and copy text from them. ➤ You can use the UP and DOWN arrow keys to browse through the command history. QuickTest copies the commands to the active command line, enabling you to repeat or reuse commands that you entered earlier.

UI Elements	Description
<right-click context menu>	<p>Provides commands that you can use to edit the content of the Command tab.</p> <ul style="list-style-type: none"> ➤ The Cut, Copy, and Paste commands enable you to use the clipboard to copy text from the command history and to edit the active command line. ➤ The Clear All command enables you to erase all of the command history.

Run Error Message Box

This message box enables you to handle run errors during a run session.



To access	This message box is displayed when there is a run error during a run session.
Relevant tasks	"How to Debug Your Test or Function Library" on page 1215

User interface elements are described below:

UI Elements	Description
Stop	Stops the run session. The run results are displayed if QuickTest is configured to show run results after the run.
Retry	QuickTest attempts to perform the step again. If the step succeeds, the run continues.
Skip	QuickTest skips the step that caused the error, and continues the run from the next step.
Debug	QuickTest suspends the run, enabling you to debug the test and any associated function library that contains a function called by the test. You can perform any of the debugging operations described in this chapter. After debugging, you can continue the run session from the step where the test or function library stopped, or you can use the step commands to control the remainder of the run session.
Help	Opens the QuickTest troubleshooting Help for the displayed error message. After you review the Help topic, you can select another button in the error message box.

Troubleshooting and Limitations - Debugging

This section describes troubleshooting and limitations for debugging.

- If a run-time error occurs during a run session, you can click **Skip** in the error message box that opens, to skip the problematic step and continue the run session. However, when debugging your test or function library, if you set a breakpoint on a line immediately after a line that causes a run-time error, QuickTest does not stop at this breakpoint when the run session continues after the error.

Workaround: Set the breakpoint at least two lines after the line that causes the run-time error.

- You cannot run the Microsoft Visual Studio debugger if QuickTest is configured to record and run your test on any open Windows-based application (**Automation > Record and Run Settings > Windows Applications**).

Workaround: Do one of the following:

- In the **MicIPC** section of the **mercury.ini** file (located in **%SYSTEMROOT%**) add these entries:
devenv.exe=0
msdev.exe=0
msscrdbg.exe=0
- Open the debugging program from a user account other than the one running QuickTest Professional.

34

Maintaining and Updating Tests

This chapter includes:

Concepts

- ▶ Why Tests Fail on page 1240
- ▶ Maintenance Run Mode on page 1242
- ▶ Updating Your Test Object Descriptions, Checkpoints, Output Values, or Active Screen Captures on page 1244

Tasks

- ▶ How to Use Maintenance Run Mode to Update Your Test When Your Application Changes on page 1248
- ▶ How to Update Test Object Descriptions, Checkpoints, Output Values, or Active Screens on page 1252

Reference

- ▶ Maintenance Run Wizard on page 1255
- ▶ Update Options Tab (Update Run Dialog Box) on page 1271

Troubleshooting and Limitations - Maintenance Mode on page 1274

Concepts

Why Tests Fail

Tests fail when QuickTest encounters a step it cannot perform or the results of a step indicate failure. In many cases this is due to the application being tested not functioning properly. QuickTest then provides you with run results that assist you in understanding how to fix your application.

Sometimes a test fails because the application being tested has changed from when the test was created and the QuickTest test needs to be updated to reflect those changes. Your object repository may also be missing some of the objects it needs to run the test. QuickTest provides tools that help identify and resolve some of these issues.

Object Changes

When QuickTest runs a step in a test, it looks for the object referred to by that step, in the object repositories associated with that test. Using the description of the object in the repository, QuickTest attempts to identify that object in the application.

QuickTest may not be able to identify the object in the application for a number of reasons:

The Object Does Not Exist in the Application

QuickTest cannot find an object in the application that matches the description of the object in the object repository. The Maintenance Run Wizard enables you to identify the object that you want your test to use.

The Parent Object Changed

QuickTest cannot find an object in the application that matches and has the same hierarchy as the object in the object repository. The Maintenance Run Wizard enables you to identify the object that you want your test to use.

The Object Description Property Values Changed

QuickTest cannot find an object in the application that is similar to, and has the same description property values as the object in the object repository. The Maintenance Run Wizard enables you to identify the object that you want your test to use.

The Object Does Not Exist in the Object Repository

QuickTest looks for the object to which the test refers, in the associated object repositories before attempting to identify that object in the application. If the object in your test cannot be found in any associated object repository, The Maintenance Run Wizard enables you to identify the object in your application that you want to add to your repository and use in your test.

Checkpoint Changes

Checkpoints fail when they encounter conditions in the application being tested that are unexpected. In many cases this is due to the application not functioning properly. QuickTest provides you with run results that assist you in understanding how to fix your application.

Sometimes checkpoints fail because the application has changed since the test was created and the QuickTest checkpoints need to be updated to reflect those changes. Update Run Mode enables you to update the checkpoints in your test to reflect changes in the application.

For example, suppose your application has an edit box whose default value used to be <Enter value> and you have checkpoint that checks this value before a new value is entered in the edit box. If the default value in the application changes to be <Enter name> then your checkpoint will fail. Update Run Mode enables you to update the expected values of your checkpoint to reflect the change in the application.

 **Maintenance Run Mode**

The Maintenance Run Wizard helps you to maintain your test when it encounters the following problems and provides the following solutions:

Problem	Solution
<p>The step failed because the object in your test cannot be identified in the application.</p>	<p>The Maintenance Run Wizard helps you identify the object in the application that you want your test to use.</p> <p>If you point to an object in the application being tested, the Maintenance Run Wizard will compare that object to the objects in the associated object repositories.</p> <p>Depending on how the property values of the object to which you point compare to the property values of the objects in the associated repositories, the Maintenance Run Wizard will suggest one of a several options for updating your test to reflect the changes in the application.</p> <p>You can also choose to add a comment to your test before the failed step.</p>
<p>The step failed because the object in your test is missing from your associated object repositories.</p>	<p>The Maintenance Run Wizard helps you add the missing object to the repository.</p> <p>You can also choose to add a comment to your test before the failed step.</p>
<p>The object in your step exists in the application, but can be identified only through Smart Identification.</p>	<p>Identifying objects using Smart Identification may cause tests to run slower. (For more information see, "Smart Identification" on page 292.) The Maintenance Run Wizard helps you modify the description of the object, so that Smart Identification is not needed.</p>

When you run a test in Maintenance Run Mode, QuickTest runs your test, and then guides you through the process of updating your steps and object repository. The Maintenance Run Wizard opens for each of the situations described above. Depending on the problem and user selections, the Maintenance Run Wizard will display several pages. For details, see "Maintenance Run Wizard Workflow" on page 1251.

When the Maintenance Run Mode ends, the Maintenance Run Wizard provides a summary of the changes it made to your test. The main Run Results Viewer also contains a Maintenance Summary which displays details of the changes made to your test, including updated and added objects, updated and commented steps, and a summary of changes to the object repository.

Updating Your Test Object Descriptions, Checkpoints, Output Values, or Active Screen Captures

When you run a test in **Update Run Mode**, QuickTest runs the test to update the set of identification properties used for test object descriptions, the Active Screen images and values, and/or the expected checkpoint values. QuickTest updates the set of identification properties for each object class in your associated object repositories according to the properties currently defined in the Object Identification dialog box. If the case-sensitivity designation of a property value has changed since you learned the object, update run mode also updates that setting. After you save the test, the updated data is used for subsequent runs.

How Test Object Descriptions are Updated when QuickTest uses Smart Identification

If you have a test that runs successfully, but in which certain objects are identified using Smart Identification, you can change the set of properties used for object identification and then use the **Update test object descriptions** option to update the test object description to use the set of properties that Smart Identification used to identify the object.

When you run the test with **Update test object descriptions** selected, QuickTest finds the test object specified in each step based on its current test object description. If QuickTest cannot find the test object based on its description, it uses the Smart Identification properties to identify the test object (if Smart Identification is enabled). After QuickTest finds the test object, it then updates its description based on the mandatory and assistive properties that you define in the Object Identification dialog box.

How Test Object Descriptions are Updated When they Contain Parameters or Regular Expressions

Any properties that were used in the previous test object description and are no longer part of the description for that test object class, as defined in the Object Identification dialog box, are removed from the new description, even if the values were parameterized or defined as regular expressions.

If the same property appears both in the test object's new and previous descriptions, one of the following occurs:

- If the property value in the previous description is parameterized or specified as a regular expression and matches the new property value, QuickTest keeps the property's previous parameterized or regular expression value. For example, if the previous property value was defined as the regular expression button.*, and the new value is button1, the property value remains button.*.
- If the property value in the previous description does not match the new property value, but the object is found using Smart Identification, QuickTest updates the property value to the new, constant property value. For example, if the previous property value was button.*, and the new value is My button, if a Smart Identification definition enabled QuickTest to find the object, My button becomes the new property value. In this case, any parameterization or use of regular expressions is removed from the test object description.

How the Case-Sensitivity of Properties is Updated

Most identification property values are not case-sensitive, meaning that QuickTest will identify an object in an application if it matches the test object description, even if the capitalization of the property values is different. However, QuickTest does treat some identification property values for specific test objects as case-sensitive. In some cases, the case-sensitivity of some identification properties may change from one version of QuickTest to the next, or as a result of a patch or hotfix installation.

In those cases, when you use Update Run to update test object descriptions, QuickTest also updates any case-sensitivity settings that may have changed.

Below is an example of the run results for a step in which the case-sensitivity of some identification properties were updated from **case-insensitive** to **case-sensitive**.

Object	Details	Result	Time
1-Update Description	Test object's previous description: Text = a (case-insensitive) Native Class = Button (case-insensitive)		
	Test object's new description: Text = a (case-sensitive) Native Class = Button (case-sensitive)	Done	05.05.2010 - 11:45:18

Result Details | Screen Recorder | System Monitor

An Example of When to Update Test Object Descriptions

Updating test object descriptions can be especially useful when you want to create or debug your test steps using object property values that are easy to recognize in your application (such as object labels), but may be language- or operating system-dependent. After you debug your test, you can use the **Update Run Mode** option to change the object descriptions to use more universal property values.

Suppose you design a test for the English version of your application. The test objects are recognized according to the identification property values in the English version, some of which may be language-dependent. You now want to use the same test for the French version of your application.

To do this, you define properties that are non-language-dependent, so that QuickTest can use these properties for object identification. For example, you can identify a link object by its **target** property value instead of its **text** property value. You can then perform an update run on the English version of your application using these new properties. This modifies the test object descriptions so that you can later run the test successfully on the French version of your application.

When to Update Checkpoint Properties and Output Property Values

Suppose you defined a text checkpoint as part of your test, and the text in your application has changed since you created your test. You can update the test to update the checkpoint properties to reflect the new text.

QuickTest updates the expected values of your checkpoints to reflect any changes that may have occurred in your application since you created the test and updates the list of items that can be retrieved in your output value steps.

When to Update Active Screen Images and Values

Suppose a dialog box in your application has changed since you recorded your test. You can update the test to update the dialog box appearance and its properties in the Active Screen.

QuickTest updates images and property values in the Active Screen to reflect any changes that may have occurred in your application since you recorded the test or if the Active Screen does not appear as it should.

You can also use this option to increase or decrease the amount of information saved and displayed in your Active Screen. Change the Capture Level slider (**Tools > Options > Active Screen** node), and run the test in Update Run Mode with the **Update Active Screen images and values** check box selected. QuickTest updates the amount of information it saves and displays in the Active Screen, based on the new setting. For more information, see "Active Screen Pane (Options Dialog Box)" on page 1434.

Tasks

How to Use Maintenance Run Mode to Update Your Test When Your Application Changes

This task describes how to use the Maintenance Run Wizard to help you maintain your test.

This task includes the following steps:

- "Prerequisites" on page 1248
- "Run the test in Maintenance Run Mode" on page 1249
- "Export and merge changes to your shared object repository (Optional)" on page 1250
- "Analyze the results of the Maintenance Run Mode session in the Run Results Viewer" on page 1250

1 Prerequisites

- You must have the Microsoft Script Debugger installed to run the tests in Maintenance Run Mode. If it is not installed, you can use the QuickTest Additional Installation Requirements Utility to install it. (Select **Start > Programs > QuickTest Professional > Tools > Additional Installation Requirements.**)
- You can run in Maintenance Run Mode only when QuickTest is set to use the **Normal** (displays execution marker) run mode. It cannot run in **Fast** mode. For more information, see "Run Pane (Options Dialog Box)" on page 1447.
- You cannot run tests in Maintenance Run Mode on applications that do not have a user interface, such as Web services.

2 Decide how long QuickTest should wait before determining that an object cannot be found (Optional)

The Maintenance Run Wizard opens often when your application has changed and QuickTest will be unable to identify objects. You may want to decrease the amount of time QuickTest waits for an object to be displayed before determining that the object cannot be found. You can change the object synchronization timeout in the Run pane of the Test Settings dialog box. Ensure that the timeout specified is sufficient for the objects in your application to load. For more information, see "Run Pane (Test Settings Dialog Box)" on page 1471.

After Maintenance Run Mode finishes you may want to return this setting to its previous value for regular test runs. (The default QuickTest setting is 20 seconds.)

3 Run the test in Maintenance Run Mode

- a Select **Automation > Maintenance Run Mode** or click the down arrow next to the **Run** button in the toolbar and select **Maintenance Run Mode**.
- b Specify the results location and the input parameter values (if applicable) for the Maintenance Run Mode session. For more information, see "Run Dialog Box: Results Location Tab (For Tests Stored in the File System)" on page 1073, and "Run Dialog Box: Input Parameters Tab" on page 1078.
- c Follow the steps in the Maintenance Run Wizard. For details, see "Maintenance Run Wizard" on page 1255 and "Maintenance Run Wizard Workflow" on page 1251.

Tip: As an alternative to using the Maintenance Run Wizard, you can update individual test object descriptions from the object in your application using the **Update from Application** option in the Object Repository window or Object Repository Manager. For more information, see "Updating Identification Properties from an Object in Your Application" on page 168.

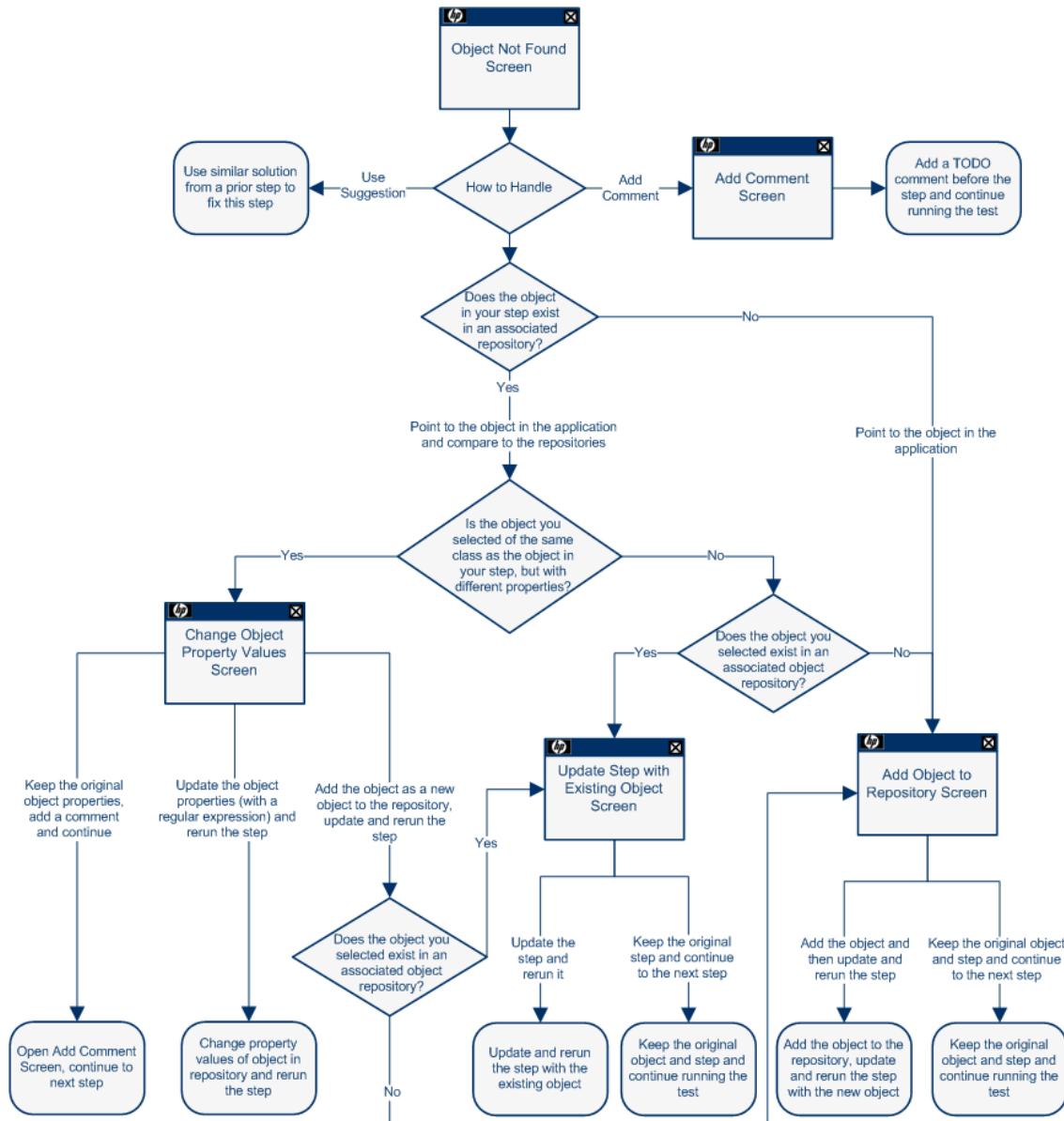
4 Export and merge changes to your shared object repository (Optional)

After using the Maintenance Run Wizard to update the test, you may want to use the **Update from Local Repository** option in the Object Repository Manager to merge the objects from the local repository back to a shared object repository. For more information, see Chapter 6, "Shared Object Repositories."

5 Analyze the results of the Maintenance Run Mode session in the Run Results Viewer

By default, when the run session ends, the Run Results Viewer opens. For more information on viewing the run session results, see Chapter 31, "Run Results Viewer." If you cleared the **View results when run session ends** check box in the Run pane of the Options dialog box, the Run Results Viewer does not open at the end of the run session. For more information on the Options dialog box, see Chapter 45, "Global Testing Options."

Maintenance Run Wizard Workflow



Note: The Object Not Found Page will not open when QuickTest uses Smart Identification to identify an object in your test. In that case, the Maintenance Run Wizard will suggest updating the object properties according to the properties currently defined in the Object Identification dialog box.

How to Update Test Object Descriptions, Checkpoints, Output Values, or Active Screens

This task describes how to update your test data so that it is accurate for subsequent runs.

This task includes the following steps:

- "Determine which data types you want to update" on page 1252
- "Run in Update Run Mode" on page 1252
- "Export and merge changes to your shared object repository (Optional)" on page 1253
- "Analyze the results of the Update Run Mode in the Run Results Viewer" on page 1253

1 Determine which data types you want to update

For details, see "Updating Your Test Object Descriptions, Checkpoints, Output Values, or Active Screen Captures" on page 1244.

2 Run in Update Run Mode

- a** Specify the settings for the update run process. For more information, see "Update Options Tab (Update Run Dialog Box)" on page 1271.
- b** Enter any required input parameter values in the Input Parameters tab. For more information, see "Run Dialog Box: Input Parameters Tab" on page 1078.

When QuickTest updates tests, it runs through only one iteration of the test and one iteration of each action in the test, according to the run option selected. For information on actions, see Chapter 14, "Actions."

When a test runs in Update Run Mode, it does not update parameterized values, such as Data Table data and environment variables. For information on parameterized values and environment variables, see Chapter 22, "Parameterizing Values." Update Run Mode does not modify the property values of existing object descriptions in the object repository. To fix the object property values to match your application, use **Maintenance Run Mode**. For more information, see "How to Use Maintenance Run Mode to Update Your Test When Your Application Changes" on page 1248.

3 Export and merge changes to your shared object repository (Optional)

When QuickTest updates tests, it always saves the updated objects in the local object repository, even if the objects being updated were originally from a shared object repository. The next time you run the tests, QuickTest uses the objects from the local object repository, as the local object repository has a higher priority than any shared object repositories.

After using Update Run Mode to update the test, you may want to use the Update from Local Repository option in the Object Repository Manager to merge the objects from the local repository back to a shared object repository. For more information, see Chapter 6, "Shared Object Repositories."

4 Analyze the results of the Update Run Mode in the Run Results Viewer

The run results for an update run session are always saved in a temporary location.

Test objects that cannot be identified during the update process are not updated. As in any run session, if an object cannot be found during the update run, the run session fails, and information on the failure is included in the Run Results. In these situations, you may want to use **Maintenance Run Mode** to resolve these problems.

By default, when the run session ends, the Run Results Viewer opens. For more information on viewing the run session results, see Chapter 31, "Run Results Viewer." If you cleared the **View results when run session ends** check box in the Run pane of the Options dialog box, the Run Results Viewer does not open at the end of the run session. For more information on the Options dialog box, see Chapter 45, "Global Testing Options."

When the update run ends, the Run Results Viewer can show:

- ▶ Updated values for checkpoints.
- ▶ Updated test object descriptions.

For example:

Step Name: Notifications:-Update Description				
Step Done				
Object	Details	Result	Time	
Notifications:- Update Description	<p>Test object's previous description:</p> <p>Text =</p> <p>Selection =</p> <p>Native Class =</p> <p>ListBox</p>	Done	5/20/2005 - 10:43:11	
	<p>Test object's new description:</p> <p>Attached Text =</p> <p>Notifications:</p>			

Reference

Maintenance Run Wizard

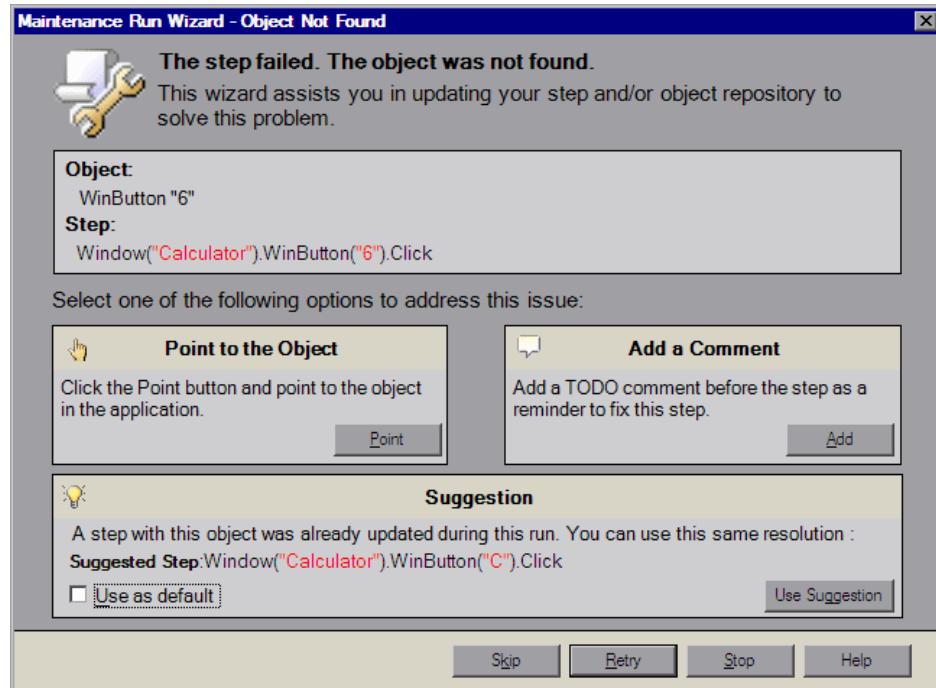
This wizard enables you to update your test when your application changes.

To access	Use one of the following: ► Select Automation > Maintenance Run Mode ► Click the down arrow next to the Run button in the toolbar and select Maintenance Run Mode .
Important information	See the prerequisite information in "How to Use Maintenance Run Mode to Update Your Test When Your Application Changes" on page 1248.
Relevant tasks	"How to Use Maintenance Run Mode to Update Your Test When Your Application Changes" on page 1248
Wizard map	"Maintenance Run Wizard Workflow" on page 1251
See also	► "Why Tests Fail" on page 1240 ► "Maintenance Run Mode" on page 1242

Object Not Found Page (Maintenance Run Wizard)

This wizard page enables you to identify the **Object** that could not be found and the **Step** QuickTest was trying to perform.

The Object Not Found page opens when an object in your test cannot be found in the application you are testing or in the associated object repositories.



Wizard map

"Maintenance Run Wizard Workflow" on page 1251

User interface elements are described below:

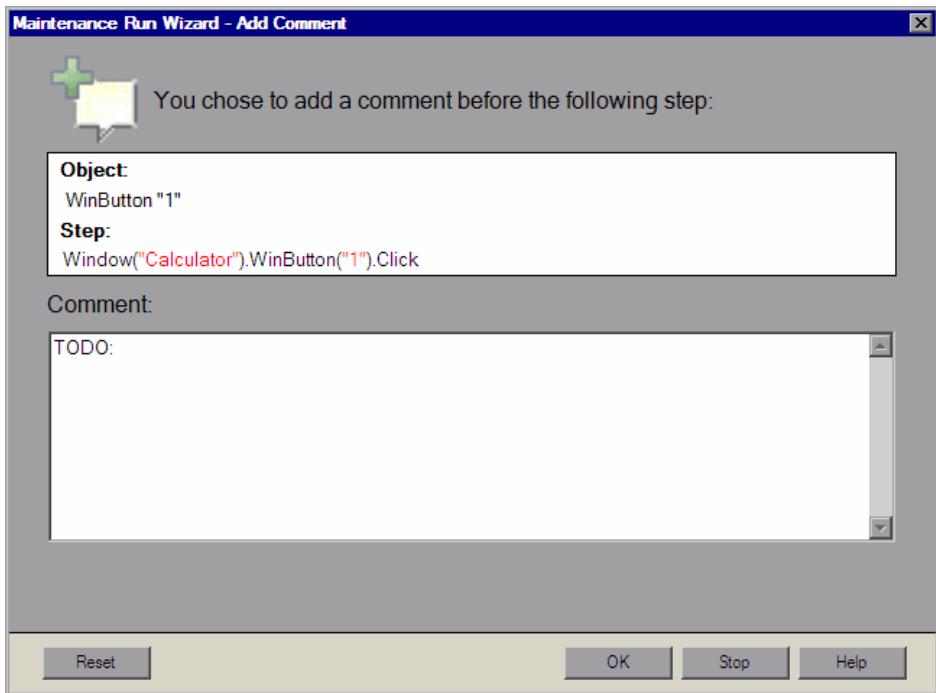
UI Elements	Description
Point to the Object	<p>Click the Point button and point to the object in the application that should be used in the step. Use this option if you know the application has changed and identifying a new object for use in the step will resolve the issue, or if the object does not exist in the associated object repositories.</p> <p>If the location to which you point is associated with several objects, the Object Selection dialog box opens. Select the correct object from the tree and click OK.</p> <p>The Point to the Object option is disabled when:</p> <ul style="list-style-type: none"> ➤ The test is open in read-only mode. ➤ The object is used within a function library function. ➤ The object's method is defined as a registered user function.
Add a Comment	<p>Use this option if you want to add a comment to your test as a reminder to fix the failed step.</p> <p>The Add a Comment option is disabled when:</p> <ul style="list-style-type: none"> ➤ The test is open in read-only mode. ➤ The object is used within a function library function. ➤ The object's method is defined as a registered user function.
Suggestion	<p>Displayed only if the Maintenance Run Wizard cannot find an object in the application that was not found earlier in the Maintenance Run Wizard run as well. If, when the object was first not found, you chose to replace it with a different object, the Maintenance Run Wizard will suggest replacing it with the same object now.</p>
Use as default	<p>If, in subsequent steps the same object cannot be found, the Maintenance Run Wizard will automatically replace the object not found with the object you added to the object repository. The Maintenance Run Wizard will not open on these subsequent steps.</p>

UI Elements	Description
Skip	<p>Skips the current step in the test and continues to run the Maintenance Run Wizard on the remainder of the test. This can be used when the problem is in the application being tested and not the QuickTest test.</p> <p>Before clicking Skip, ensure that the application is ready for the next step in the test.</p>
Retry	Retries the current step.
Stop	Stops the Maintenance Run and opens the Maintenance Mode Summary Page (Maintenance Run Wizard).

Add Comment Page (Maintenance Run Wizard)

This wizard page enables you to add a comment to your test before the current step. This can be used when you believe there is a problem in your test, but identifying the object in the application will not solve the problem, or you want to fix the test manually.

The Add Comment page creates a comment in your test beginning with the word TODO along with text you add, as a reminder to fix the step at a later time.



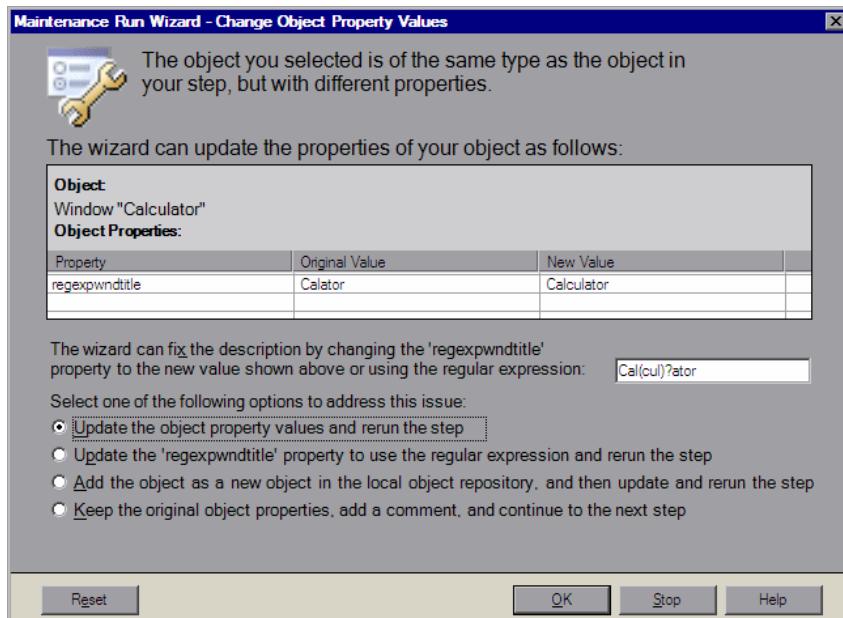
[Wizard map](#)

"Maintenance Run Wizard Workflow" on page 1251

Change Object Property Values Page (Maintenance Run Wizard)

This wizard page enables you to update the property values of the object in the associated object repository to match the property values of the object to which you pointed in the application.

The Change Object Property Values page opens when the object to which you pointed is of the same class as the object in your step, but it has different description property values.



Important information	If the object to which you point has a different parent object than the one in the object repository and has different property values, the Change Object Property Values page opens twice. The first time it enables you to update the parent object of the object in the object repository to match the parent object of the object to which you pointed. The second time it enables you to update the object in the object repository to match the object to which you pointed.
Wizard map	"Maintenance Run Wizard Workflow" on page 1251

Change Object Property Values Page - Central Area

User interface elements of the central area of the Change Object Property Values page are described below (unlabeled elements are shown in angle brackets):

UI Elements	Description
Object	The object in an associated object repository that is of the same class as the object to which you pointed in the application.
Object Properties	A table displaying the changes that will be made to the property values of the object in the object repository.
Property	The name of the property whose value will be changed.
Original Value	The original property value of the object in the object repository.
New Value	The new property value for the object in the object repository, based on the object to which you pointed in the application.
<Recommended regular expression>	<p>Depending on the object to which you pointed, the Change Object Property Value page may include a message that a regular expression can be used to update the property value of the object in the associated object repository. You can modify the suggested regular expression in the edit box. For more information on regular expressions, see "Regular Expressions Overview" on page 863.</p> <p>Notes:</p> <ul style="list-style-type: none"> ► If the Maintenance Run Wizard does not determine that a regular expression is relevant for the new property value, the Change Object Property Value page does not display the suggested regular expression below the properties table. The Update the <property name> property to use the regular expression and rerun the step radio button is also not displayed. ► In a situation where more than one property can use a regular expression, the Maintenance Run Wizard will only suggest a regular expression for the first property value.

Change Object Property Values Page - Options

User interface elements are described below:

UI Elements	Description
Update the object property and rerun the step	Updates the property values of the object in the object repository to match those of the object to which you pointed in the application, and reruns the step. The new property values are shown under New Value.
Update the <property name> property to use the regular expression and rerun the step	Displayed only if the property value can be updated to use a regular expression. Updates the property value of the object in the object repository with the regular expression as shown in the edit box, and reruns the step.
Add the object as a new object in the local object repository, and then update and rerun the step	<p>Adds the object to which you pointed, with its current properties, as a new object in the local object repository. This new object may already exist in an associated object repository. Depending on whether the object you want to add already exists in an associated object repository, one of the following pages opens:</p> <ul style="list-style-type: none"> ▶ "Update Step with Existing Object Page (Maintenance Run Wizard)" on page 1263 ▶ "Add Object to Repository Page (Maintenance Run Wizard)" on page 1265
Keep the original object properties, add a comment, and continue to the next step	Keeps the original object properties of the object in the object repository. Opens the Add Comment page, enabling you to add a comment before the step, and then continues to the next step.
Reset	Returns to the Object Not Found page, where you can point to a different object in the application or choose a different course of action for this step.

Update Step with Existing Object Page (Maintenance Run Wizard)

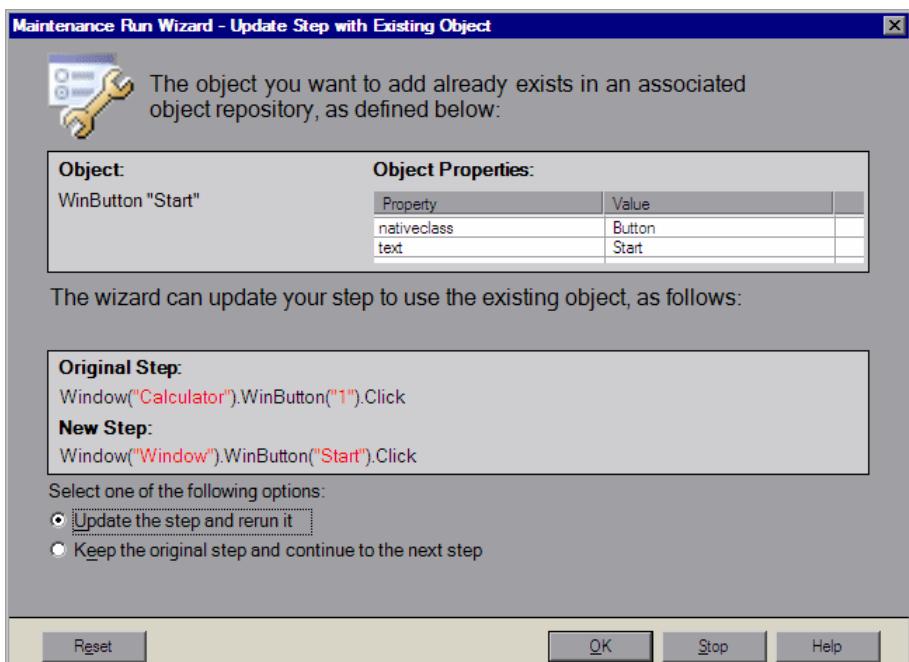
This wizard page enables you to update the step in your test to use an object that already exists in an associated object repository.

The Update Step with Existing Object page opens if the object to which you pointed in the Object Not Found page exists in an associated object repository and:

- The object to which you pointed is not of the same class as the object in your step, but with different description property values.

Or

- In the Change Object Property Values page you chose **Add the object as a new object in the local object repository, and then update and rerun the step.**



Wizard map

"Maintenance Run Wizard Workflow" on page 1251

Update Step with Existing Object Page - Central Area

User interface elements of the central area of the Update Step with Existing Object page are described below:

UI Elements	Description
Object	The object in an associated object repository that is the same as the object to which you pointed in the application.
Object Properties	The properties and property values of the object to which you pointed in the test application.
Original Step	The failed original step, with the object that could not be found.
New Step	The new step as it would appear updated to refer to the object which already exists in an associated object repository.

Update Step with Existing Object Page - Options

User interface elements are described below:

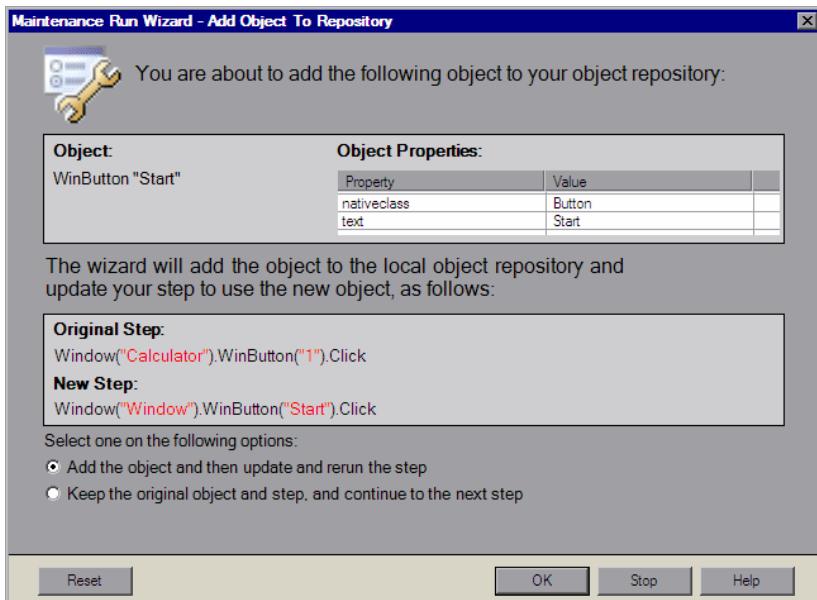
UI Elements	Description
Update the step and rerun it	Updates the failed step as shown under New Step and reruns the step. The Maintenance Run Wizard does not remove the original step from your test. The original step is converted into a comment and the updated step is added below it.
Keep the original step and continue to the next step	Keeps the original step and continues to run the Maintenance Run Wizard on the remainder of the test.
Reset	Returns to the Object Not Found page, where you can point to a different object in the application or choose a different course of action for this step.

Add Object to Repository Page (Maintenance Run Wizard)

This wizard page enables you to add the object to which you pointed to the object repository.

The Add Object to Repository page opens when:

- the object **in your step** does not exist in any associated repository.
 - the object **to which you pointed** does not exist in any associated object repository and:
 - the object to which you pointed is not of the same class as the object in your step, but with different description property values.
- Or
- in the Change Object Property Values page you chose **Add the object as a new object in the local object repository, and then update and rerun the step**.



Wizard map

"Maintenance Run Wizard Workflow" on page 1251

Add Object to Repository Page - Central Area

User interface elements are described below:

UI Elements	Description
Object	The object to which you pointed in the test application.
Object Properties	The properties and property values of the object to which you pointed in the test application.
Original Step	The failed original step, with the object that could not be found.
New Step	The new step as it would appear updated to refer to the object being added to the object repository.

Add Object to Repository Page - Options

User interface elements are described below:

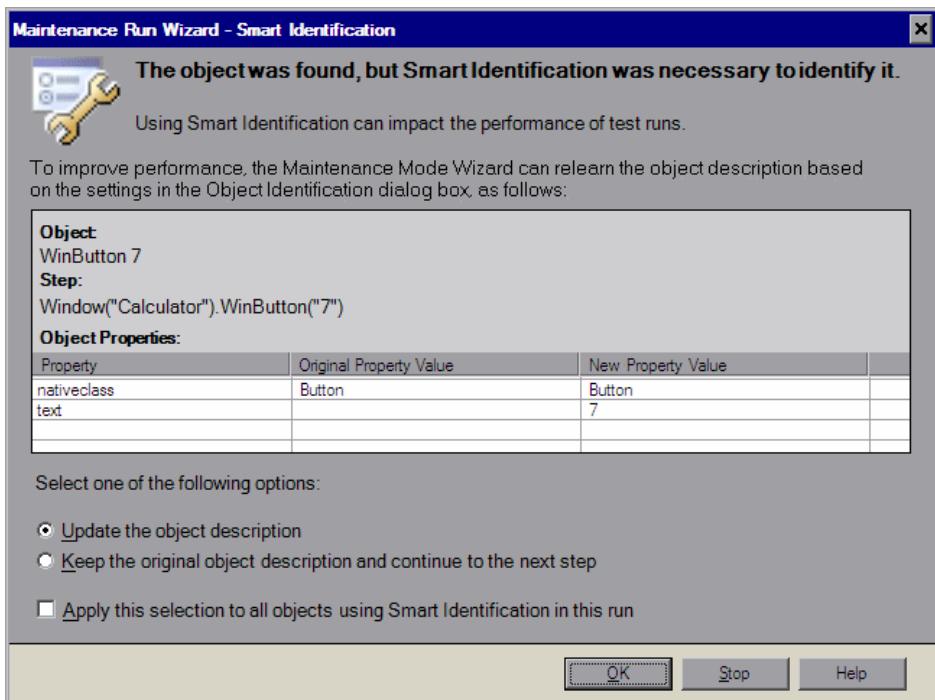
UI Elements	Description
Add the object and then update and rerun the step	Adds the new object to the object repository, updates the failed step as shown under New Step and reruns the step. The Maintenance Run Wizard does not remove the original step from your test. The original step is converted into a comment and the updated step is added below it.
Keep the original object and step, and continue to the next step	Keeps the original step containing the original object and continues to run the Maintenance Run Wizard on the remainder of the test.
Reset	Returns to the Object Not Found page, where you can point to a different object in the application or choose a different course of action for this step.

Smart Identification Page (Maintenance Run Wizard)

This wizard page enables you to update the object description according to the properties currently defined in the Object Identification dialog box.

The Smart Identification page opens if QuickTest used the Smart Identification mechanism to identify the object in your test. For information on the Smart Identification mechanism, see "Smart Identification" on page 292.

Smart Identification may slow down test execution, as it is only activated after the object synchronization timeout has been reached.



Important information	Your test may use Smart Identification because the set of properties used for object identification does not uniquely identify all the objects of a particular class. You can use the Update test object descriptions option of Update Run Mode to change the set of properties used for object identification for all objects in a specific class. This may be more useful than using the Maintenance Run Wizard for every object in that class. For more information, see "How Test Object Descriptions are Updated when QuickTest uses Smart Identification" on page 1244.
Wizard map	"Maintenance Run Wizard Workflow" on page 1251

Smart Identification Page - Central Area

User interface elements are described below:

UI Elements	Description
Object	The object in your application that required the use of the Smart Identification mechanism to be identified.
Step	The step in your test in which the object is referenced.
Object Properties	<p>Property. The list of properties in the old and new object description.</p> <p>Original Property Value. The original value of the property in the Property column. Properties that have no value were not part of the original object description.</p> <p>New Property Value. The new value of the property in the Property column.</p>

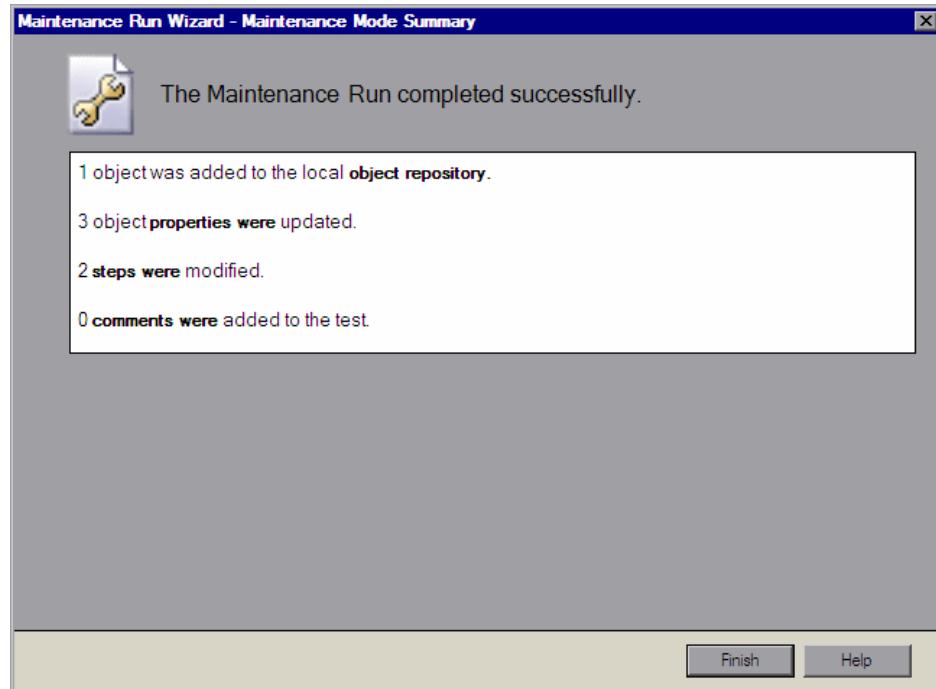
Smart Identification Page - Options

User interface elements are described below:

UI Elements	Description
Update the object description	Updates the object description to use the set of properties currently defined in the Object Identification dialog box for the object in your test. Make sure that the set of properties defined in the Object Identification dialog box for the object is sufficient to uniquely identify the object.
Keep the original description and continue to the next step	Keeps the original step containing the original object and continues to run the Maintenance Run Wizard on the remainder of the test. The Smart Identification page will not open for this object again during the run.
Apply this selection to all objects using Smart Identification in this run	Uses your radio button selection above for all objects in the test that need the Smart Identification mechanism to be identified.

Maintenance Mode Summary Page (Maintenance Run Wizard)

This wizard page enables you to view a summary of the Maintenance Mode run.



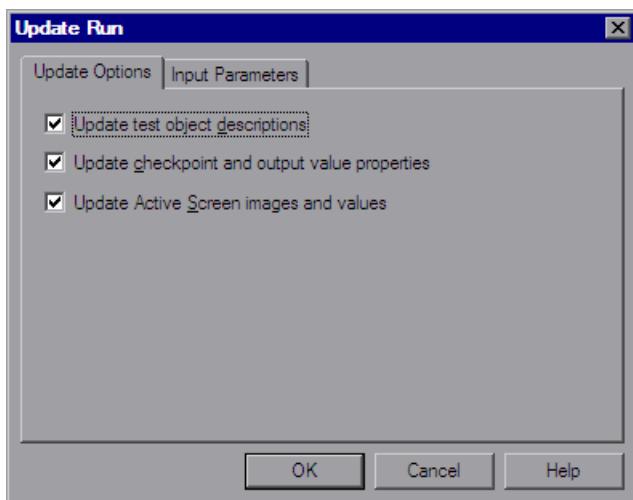
Wizard map

"Maintenance Run Wizard Workflow" on page 1251

The Maintenance Mode Summary page displays the number of objects that were added to the local **object repository**, the number of object **properties** that were updated, the number of **steps** that were modified, and the number of **comments** that were added to the test.

Update Options Tab (Update Run Dialog Box)

The Update Options tab enables you to specify which aspects of your test you want to update, such as test object descriptions, expected checkpoint values, and/or Active Screen images and values. After you save the test, the results of the updated test are used for subsequent runs.



To access	Use one of the following: <ul style="list-style-type: none"> ▶ Select Automation > Update Run Mode ▶ Click the down arrow next to the Run button in the toolbar and select Update Run Mode
Important information	See "Considerations for the Update Options Tab (Update Run Dialog Box)" on page 1273
Relevant tasks	"How to Update Test Object Descriptions, Checkpoints, Output Values, or Active Screens" on page 1252
See also	<ul style="list-style-type: none"> ▶ "Updating Your Test Object Descriptions, Checkpoints, Output Values, or Active Screen Captures" on page 1244 ▶ "Run Dialog Box: Input Parameters Tab" on page 1078

User interface elements are described below:

UI Elements	Description
Update test object descriptions.	<p>QuickTest updates the set of properties for each object class in your associated object repositories according to the properties currently defined in the Object Identification dialog box. You can use this option to modify the set of properties used to identify an object of a certain type.</p> <p>For details, see "Updating Your Test Object Descriptions, Checkpoints, Output Values, or Active Screen Captures" on page 1244.</p>
Update checkpoint properties and output property values.	<p>QuickTest updates the expected values of your checkpoints to reflect any changes that may have occurred in your application since you created the test and updates the list of items that can be retrieved in your output value steps.</p> <p>The output value option is mainly relevant for XML output value steps used with Web services tests. For more information, see the section describing Web services in the <i>HP QuickTest Professional Add-ins Guide</i>.</p> <p>For more information, see "When to Update Checkpoint Properties and Output Property Values" on page 1247.</p>
Update Active Screen images and values	<p>QuickTest updates images and property values in the Active Screen to reflect any changes that may have occurred in your application since you recorded the test or if the Active Screen does not appear as it should.</p> <p>For more information, see "When to Update Active Screen Images and Values" on page 1247.</p>

Considerations for the Update Options Tab (Update Run Dialog Box)

- When using the **Update test object descriptions**, if the property set you select in the Object Identification dialog box for an object class is not ideal for a particular object, the new object description may cause future runs to fail. Therefore, it is recommended that you save a copy of your object repositories (or check them into a Quality Center project with version control support, if applicable) before updating them, so that you can return to the previously saved version, if necessary.
- If checkpoint property values are parameterized or include regular expressions, they are not updated when using the **Update checkpoint properties and output property values** option.
- If you selected the **Save only selected area** check box when creating a bitmap checkpoint, the **Update Run Mode** option updates only the saved area of the bitmap; it does not update the original, full size object. To include more of the object in the checkpoint, create a new checkpoint. For more information, see "Bitmap Checkpoints" on page 619.

Troubleshooting and Limitations - Maintenance Mode

This section describes troubleshooting and limitations for Maintenance Mode.

- When running in Maintenance Mode, if a step contains a programmatic description for an object that is not found in an application, it may take a while for Maintenance Mode to indicate there is a problem. If you use the option to point to the object, it may take a while for Maintenance Mode to reopen afterwards.
- When the Maintenance Run Wizard cannot find an object, and the test object description for that object does not have any mandatory or assistive properties (it is identified only by its ordinal identifier, such as a Browser test object), then when you point to the object, the wizard is unable to fix the problem and displays a message that the object you pointed to has a test object description that is similar to the object that QuickTest could not identify.

Workaround: Use the **Update from Application** option in the Object Repository dialog box (for objects in the local respiratory) or in the Object Repository Manager (for objects in a shared object repository) to fix the test object description.

- When your Windows Display settings are set to use large fonts, the text in the Maintenance Run Wizard screens may be truncated.
- When the Maintenance Run Wizard cannot find an object in the application, and you point to a different object class to replace it, the Maintenance Run Wizard offers to add a step with that object and the object's default method. However, the wizard does not insert any method arguments for the step. If the step method has required arguments and you accept the step that the Maintenance Run Wizard proposes without modifying it, the step will fail when you run it.

Workaround: Enter valid method arguments for the step.

- When running in Maintenance Mode, QuickTest may replace test objects with XPath or CSS identifier property values with new objects from your application.

Workaround: Use the **Update from Application** option in the Object Repository Manager to update specific test objects with XPath or CSS identifier property values.

- Maintenance Mode cannot fix problems in an object's properties, if the object is used in function library that is called by a step. If Maintenance Mode detects a problem in an object in a function library, a message is displayed indicating that the test is read-only and that Maintenance Mode is disabled.

Part VIII

Working with the QuickTest Panes

35

QuickTest Window Layout

This chapter includes:

Concepts

- QuickTest Window Layout Customization - Overview on page 1280

Tasks

- How to Customize the QuickTest Window on page 1281

Reference

- Tips and Considerations for Customizing the QuickTest Window Layout on page 1291
- Button Appearance Dialog Box on page 1294
- Customization Mode - Context Menu Options on page 1296
- Customize Dialog Box on page 1297
- Windows Dialog Box on page 1304

Troubleshooting and Limitations - QuickTest Window Layout
on page 1305

Concepts

QuickTest Window Layout Customization - Overview

You can modify the layout of the QuickTest window. For example, you can move and resize panes, select to show or auto-hide panes, create tabbed panes, and select which toolbars to display. If needed, you can also restore the default layout.

You can also resize the QuickTest window to suit your needs for each type of QuickTest session (view/edit, record, and run sessions). For example, you can display QuickTest in full view when creating or editing a test, and minimize the QuickTest window during a run session.

When you customize or restore the QuickTest window layout, QuickTest applies the changes to all document types and session types.

QuickTest Window Layout Customization in Different Modes

QuickTest works in several different modes: view/edit, record, and run. You may want to modify the QuickTest layout to match the functionality of a mode. For example, when recording, it is often convenient to have QuickTest partially visible. This enables you to watch steps being added as you record your test without viewing the Active Screen. When running a test, it is often convenient to minimize QuickTest so that you can view your application during the test run. When viewing or editing a test, it may be convenient to maximize the QuickTest window, with all panes showing.

QuickTest remembers the size and location of its main window and all of its panes for each mode. When QuickTest enters a mode, the layout reverts to the most recently used layout for that mode. This means that the main QuickTest window and each of its panes are maximized, minimized, or resized, based on the previous layout of the current mode.

Tasks

How to Customize the QuickTest Window

This section describes how to customize and modify the QuickTest window layout, menu items, toolbars and panes.

This task includes the following steps:

- "Move Panes" on page 1282
- "Show and Hide Panes" on page 1287
- "Float and Dock Panes" on page 1288
- "Float and Dock Toolbars" on page 1288
- "Float and Dock Menus" on page 1288
- "Add a Command to a Toolbar or Menu" on page 1288
- "Add or Remove Default Buttons From an Existing Toolbar" on page 1289
- "Set the QuickTest Layout for Record Mode" on page 1289
- "Set the QuickTest Layout for Run Mode" on page 1289
- "Restore the Default Layout of the QuickTest Window" on page 1290
- "Restore Default Toolbar Buttons" on page 1290
- "Restore Default Menu Items" on page 1290
- "Restore the Default Layout of a Toolbar" on page 1290

Move Panes

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

- 1 In the QuickTest window, drag the title bar or tab of the pane you want to move. (If the required pane is not displayed in the QuickTest window, you can select it from the **View** menu).

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

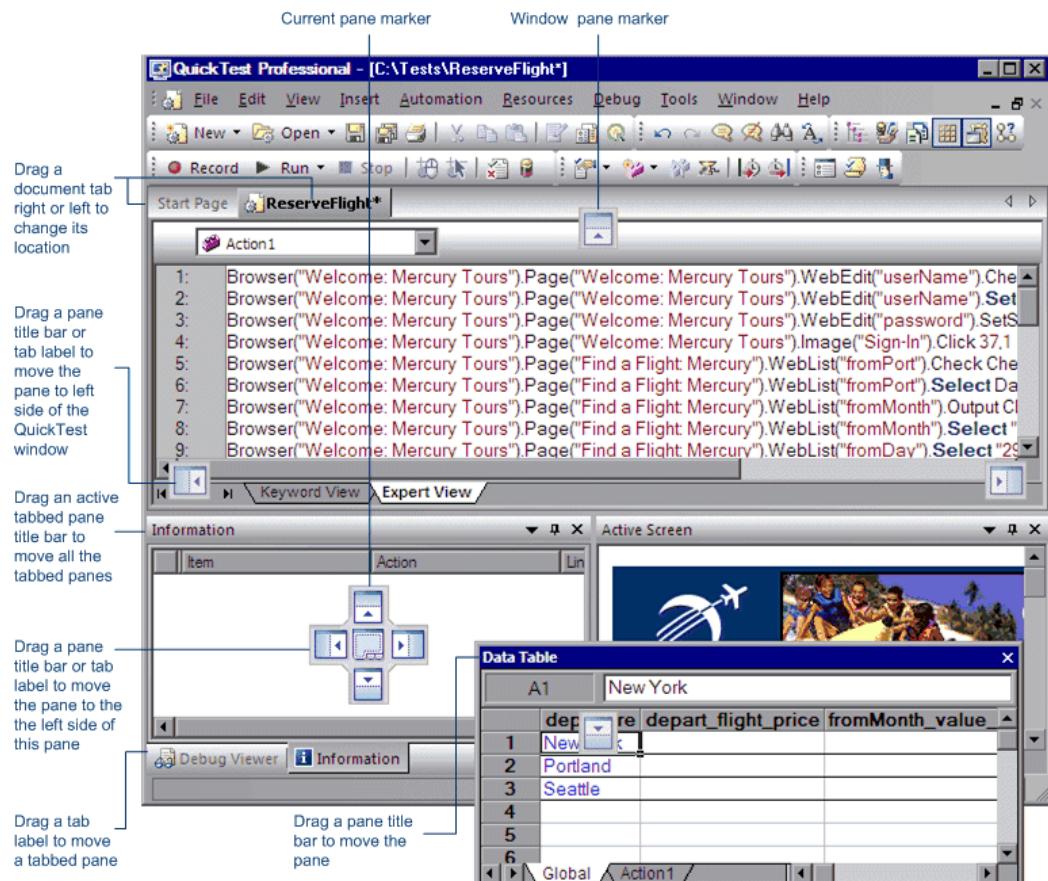
- 2 Repeat this procedure for each pane you want to move.

During the moving process, the following markers are displayed:

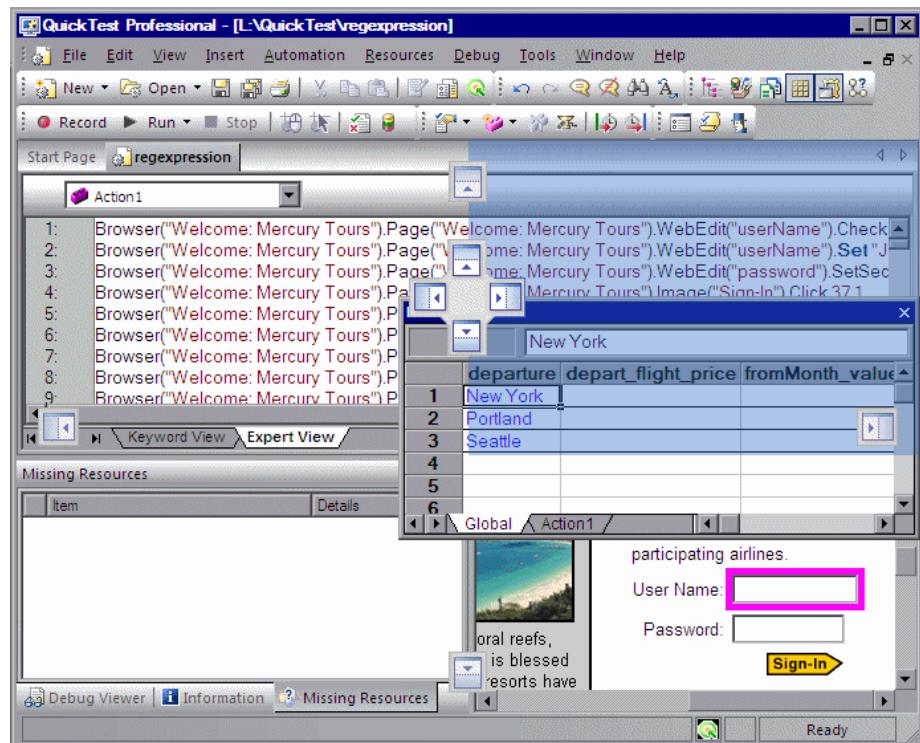
Type	Marker	Description
Current pane markers		<p>Enables you to:</p> <ul style="list-style-type: none"> ➤ position the pane as a new pane in the top, bottom, left or right half, or middle of the active pane, according to the arrow marker selected when you release the mouse button. ➤ position the pane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

Type	Marker	Description
Window pane markers		Enables you to position the pane across the top of the QuickTest window.
		Enables you to position the pane across the right side of the QuickTest window.
		Enables you to position the pane across the bottom of the QuickTest window.
		Enables you to position the pane across the left side of the QuickTest window.

The following illustrates an example of the QuickTest window at the beginning of the moving process.

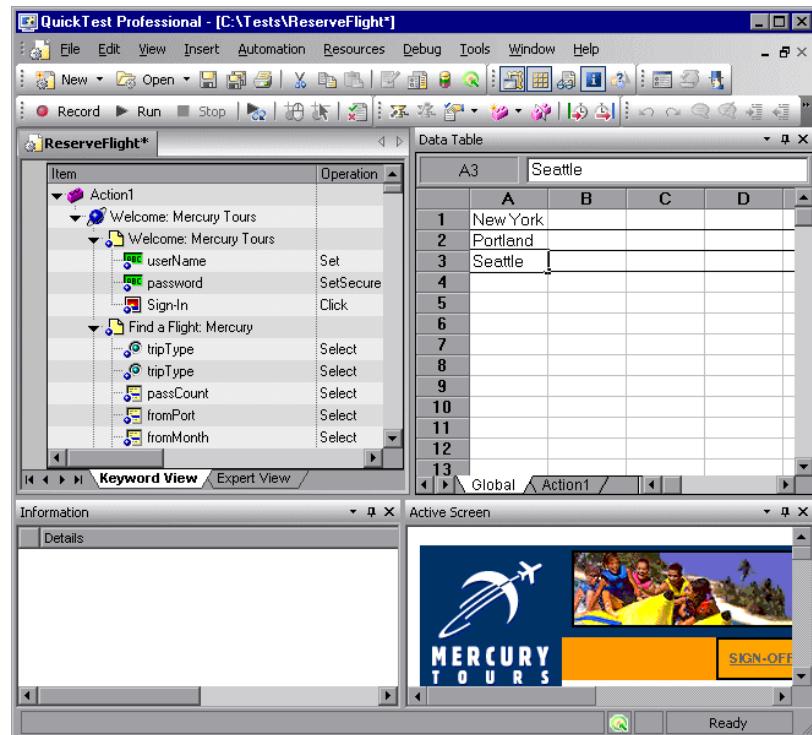


The following illustrates an example of the QuickTest window during the moving process.



Chapter 35 • QuickTest Window Layout

The following illustrates an example of the QuickTest window after the moving process.



Show and Hide Panes

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

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

Button	Description
	<p>The Menu button enables you to select any of the following:</p> <ul style="list-style-type: none"> ➤ Floating. Opens the pane on top of all the other windows and panes, with its own title bar ➤ Docking. Docks the pane to the QuickTest window. ➤ Auto-hide. Displays the pane as a side-tab either at the top or bottom of the QuickTest window, or on one of the edges, according to its position in the QuickTest window. ➤ Hide. Closes the pane.
	<p>The Auto Hide button hides the pane. The pane is displayed as a side-tab either at the top or bottom of the QuickTest window, or on one of the edges, according to its position in the QuickTest window. To display the pane, hold the cursor over the side-tab. The button toggles to the Dock button, shown below.</p>
	<p>The Dock button docks the pane to the QuickTest window. The pane returns the position it was in before it was hidden, and the button toggles to the Auto Hide button, shown above.</p>
	<p>The Close button closes the pane. The pane is removed from the QuickTest window. To reopen the pane, select it from the View menu.</p> <p>Tip: You can also close a pane by right-clicking and choosing Hide from the context menu.</p>

Float and Dock Panes

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

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

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

Float and Dock Toolbars



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

Float and Dock Menus



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

Add a Command to a Toolbar or Menu

You can add or remove commands to be included in a toolbar or menu as shortcuts.

- 1 Open the **Customize** dialog box and select the **Commands** tab.
- 2 In the **Categories** list, find and select the menu name that contains the command you want to add to the toolbar. To view all the available commands in alphabetical order select **All Commands**.
- 3 In the **Commands** list, select the command you want to add, drag it to a toolbar or the menu bar, and release the mouse button.

While dragging, a marker is displayed indicating the location where the command will be placed.

Add or Remove Default Buttons From an Existing Toolbar

- 1 Right-click the customize toolbar button .
- 2 Select **Add or Remove Buttons**.
- 3 Select the menu whose buttons you want to modify.
- 4 Select or deselect the specific button.

Set the QuickTest Layout for Record Mode

- 1 Open a new or existing test.
- 2 Start a recording session.
- 3 Record one step.
- 4 Set all of your layout preferences for the recording mode.
- 5 Stop the recording session.

Set the QuickTest Layout for Run Mode

- 1 Enter a breakpoint before the first step in the test. This enables you to arrange the layout during the run session. For information on how to set a breakpoint, see "Breakpoints" on page 1213.
- 2 Run your test.
- 3 When QuickTest reaches the breakpoint, set all of your layout preferences for the run mode.
- 4 Stop the run session.

Restore the Default Layout of the QuickTest Window

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

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

Restore Default Toolbar Buttons

- 1 Click the customize toolbar button  while in normal mode.
- 2 Select **Add or Remove Buttons**.
- 3 Select the menu whose button you want to restore and select **Restore Toolbar**.

Restore Default Menu Items

Select the **Menu Bar** in the Toolbars tab and click the **Restore Selected** button.

Restore the Default Layout of a Toolbar

- 1 Right-click the customize toolbar button  (Not available for the menu bar).
- 2 Select **Add or Remove Buttons**.
- 3 Select the toolbar whose layout you want to restore.
- 4 Select **Restore Toolbar**.

Toolbars are listed in the **Add or Remove Buttons** selection per row of toolbars.

Reference

Tips and Considerations for Customizing the QuickTest Window Layout

This section includes:

- "Tips for Customizing Toolbars and Menus" on page 1291
- "Considerations for Customizing Toolbars and Menus" on page 1292
- "Moving Panes" on page 1292
- "Showing, Hiding, and Floating Panes" on page 1293

Tips for Customizing Toolbars and Menus

- Toolbar and menu customization settings are created and saved for each Windows user.
- You can delete any button or command while the Customize Dialog box is open. Drag the toolbar button you want to remove from the toolbar to any location outside the toolbars area. The toolbar button is removed.
- You can restore the default buttons and layout for a selected toolbar or for all toolbars using the **Restore Selected** or **Restore All** buttons in the Toolbars tab. You can also restore the default buttons and layout for a toolbar by right-clicking the customize toolbar button , selecting **Add or Remove Buttons**, selecting the toolbar whose settings you want to restore, and then selecting **Restore Toolbar**.
- While the Customize Dialog box is open you can drag toolbar buttons from one toolbar to another toolbar and drag and drop to change the order of items in a menu.

Considerations for Customizing Toolbars and Menus

Some QuickTest add-ins add commands or menus to the QuickTest window. If you are working with add-ins and you customize the toolbars, consider the following:

- QuickTest remembers the customizations for each add-in, even if you close and reopen QuickTest.
- When QuickTest runs without these add-ins, all commands and menus added by the add-ins are removed from the QuickTest window.
- If you customize the toolbars first and then run QuickTest with add-ins, the additional commands and menus will be placed as close as possible to their intended locations, according to adjacent items.
- To create a new menu, select **New Menu** in the **Categories** list and drag the **New Menu** item to the menu bar or a toolbar. To create a name for the new menu, see "Button Appearance Dialog Box" on page 1294.
- To add a command to an existing menu, drag the command over the menu item. The menu item expands. Drag the marker to the location in the menu where you want to add the command, and release the mouse button.

Moving Panes

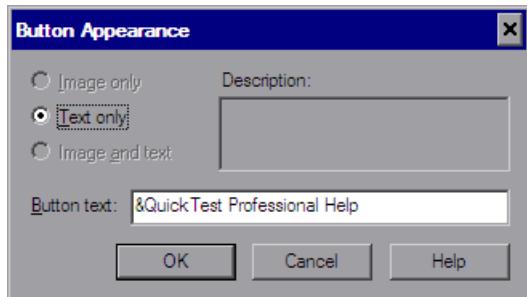
- To move a dockable pane without snapping it into place, press CTRL while dragging it to the required location.
- To move a single, tabbed pane, drag the tab label. When you start dragging the tabbed pane, the tab is removed, and its title bar is displayed.
- To move all tabbed panes simultaneously, drag the title bar of the active tabbed pane.

Showing, Hiding, and Floating Panes

- To auto-hide all the tabbed panes, select the title bar of the active tabbed pane, right-click and select **Auto Hide**. The tabbed panes are displayed as a group of side-tabs on the edge of the QuickTest window, and each pane is displayed only when you hold the cursor over that side-tab.
- If you auto-hide the Information pane, it is automatically displayed when syntax errors are detected in a test script.
- You cannot auto-hide floating panes or individual tabbed panes.
- By default, auto-hidden panes open across the QuickTest window, according to their position in the QuickTest window. For example, if the docked pane was positioned on the right side of the QuickTest window, it is displayed as a side tab on the right edge of the QuickTest window, and opens across the right side of the QuickTest window when selected.
- To float a pane, right-clicking the title bar and choosing **Floating** from the context menu. The pane opens on top of all the other windows and panes, with its own title bar. To dock the pane, double-click the title bar, or right-click the title bar and select **Docking**. The pane returns to its previous position in the QuickTest window.

Button Appearance Dialog Box

This dialog box enables you to modify the appearance of a button or menu.



To access	<p>Do one of the following:</p> <ul style="list-style-type: none">➤ Select the Tools > Customize menu command, right-click a button or menu, and select Button Appearance.➤ Click the customize toolbar button , select Add or Remove Buttons > Customize, then right-click a button or menu, and select Button Appearance.➤ Right-click on the menu bar or any toolbar and select Customize, then right-click a button or menu, and select Button Appearance.
------------------	---

User interface elements are described below:

UI Elements	Description
Image only	Displays the image for the button in the toolbar or menu. This radio button is disabled for items that have no default image.
Text only	Displays the text label for the button in the toolbar or menu.
Image and text	Displays the image and text label for the button or menu in the toolbar or menu. This radio button is disabled for items that have no default image.
Description	The description of the button.
Button text	<p>The text label for the button or menu. The text label for the button can be modified when either the Text only or Image and text radio buttons are selected.</p> <p>You can create a mnemonic (an underlined character for keyboard navigation) for any button text. Add the & character to the text label for the button before the letter you want to define as the mnemonic. Each button text can have only one mnemonic.</p> <p>To create button text that contains a '&' symbol specify the symbol twice. For example: "Help && Support" will generate the following text: "Help & Support".</p>

Customization Mode - Context Menu Options

While the Customize dialog box is open, QuickTest is in customization mode. The following options are available in the context menu when you right-click the menu bar or toolbar buttons in customization mode:

Option	Description
Restore Default	Restores the default setting for the button. This selection is disabled for menu options.
Copy Button Image	Copies the button image to the clipboard. This selection is disabled for items that have no default image.
Delete	Deletes the menu or button. Note: Any customizations of the menu bar will be lost. Tip: For information about restoring buttons and menu items, see "Restore Default Toolbar Buttons" on page 1290 For more information, see "Toolbars Tab (Customize Dialog Box)" on page 1300.
Button Appearance	Opens the Button Appearance dialog box. For more information, see "Button Appearance Dialog Box" on page 1294.
Image	Displays the image for the button in the toolbar or menu. This selection is disabled for items that have no default image.
Text	Displays the text label for the button in the toolbar or menu. This selection is disabled for menu options.
Image and Text	Displays the image and text label for the button or menu option in the toolbar or menu. This selection is disabled for items that have no default image.
Start Group	Places a divider in the toolbar or menu before the current button to indicate a new group of buttons.

Customize Dialog Box

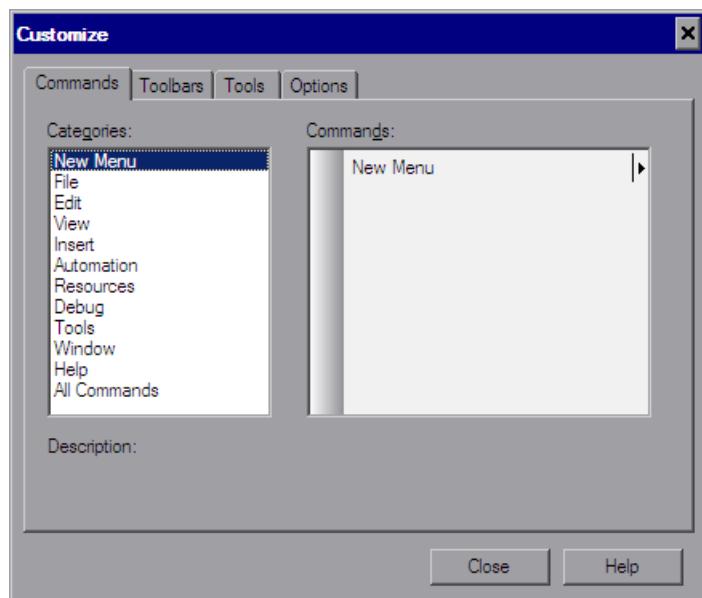
This dialog box enables you to enable you to customize your QuickTest toolbars, menus, shortcut keys, tooltips, and so on.

This dialog box contains the following tabs:

- "Commands Tab (Customize Dialog Box)" on page 1297
- "Options Tab (Customize Dialog Box)" on page 1299
- "Toolbars Tab (Customize Dialog Box)" on page 1300
- "Tools Tab (Customize Dialog Box)" on page 1302

Commands Tab (Customize Dialog Box)

This tab enables you to customize commands in toolbars and menus, and create new menus.



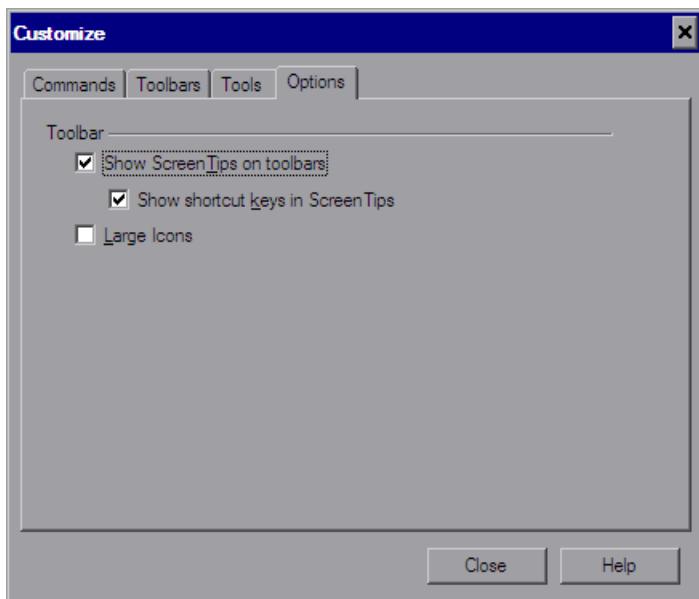
To access	<p>Do one of the following:</p> <ul style="list-style-type: none"> ▶ Select the Tools > Customize menu command and then click the Commands tab. ▶ Click the customize toolbar button , select Add or Remove Buttons > Customize, and then click the Commands tab. ▶ Right-click on the menu bar or any toolbar and select Customize and then click the Commands tab.
Relevant tasks	"Add a Command to a Toolbar or Menu" on page 1288
See also	<ul style="list-style-type: none"> ▶ "Customization Mode - Context Menu Options" on page 1296 ▶ "Tips for Customizing Toolbars and Menus" on page 1291 ▶ "Options Tab (Customize Dialog Box)" on page 1299 ▶ "Toolbars Tab (Customize Dialog Box)" on page 1300 ▶ "Tools Tab (Customize Dialog Box)" on page 1302

User interface elements are described below:

UI Elements	Description
Categories	A list of all of the menu items in the menu bar, with the addition of New Menu and All Commands .
Commands	A list of all the commands available in the menu item selected in the Categories list. Commands that appear in drop-down lists or sub-menus are listed as individual commands in the Commands section. For example, Test in the New drop-down list in the Standard toolbar is listed as the individual command New : Test in the File category.
Description	A description of the selected command in the Command list.

Options Tab (Customize Dialog Box)

This tab enables you to display ScreenTips (tooltips), shortcut keys and large or small icons in the QuickTest display.



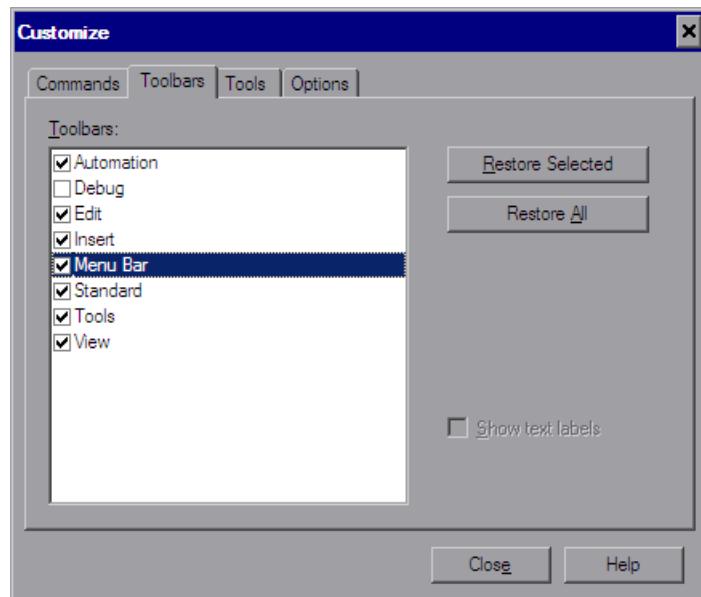
To access	<p>Do one of the following:</p> <ul style="list-style-type: none"> ➤ Select the Tools > Customize menu command and then click the Options tab. ➤ Click the customize toolbar button , select Add or Remove Buttons > Customize, and then click the Options tab. ➤ Right-click on the menu bar or any toolbar and select Customize and then click the Options tab.
See also	<ul style="list-style-type: none"> ➤ "Customization Mode - Context Menu Options" on page 1296 ➤ "Tips for Customizing Toolbars and Menus" on page 1291 ➤ "Commands Tab (Customize Dialog Box)" on page 1297 ➤ "Toolbars Tab (Customize Dialog Box)" on page 1300 ➤ "Tools Tab (Customize Dialog Box)" on page 1302

User interface elements are described below:

UI Elements	Description
Show ScreenTips on toolbars	Turns ScreenTips (tooltips) on or off. Select this check box to display ScreenTips in the QuickTest display.
Show shortcut keys in ScreenTips	Display shortcut keys in the ScreenTips. Available only when the Show ScreenTips on Toolbar check box is selected.
Large Icons	Turns large icons on or off. Select this check box to display large icons in the QuickTest display.

Toolbars Tab (Customize Dialog Box)

This tab enables you to show or hide toolbars or the menu bar. You can also restore the default setting for one or all toolbars or the menu bar, and display text labels for toolbar buttons.



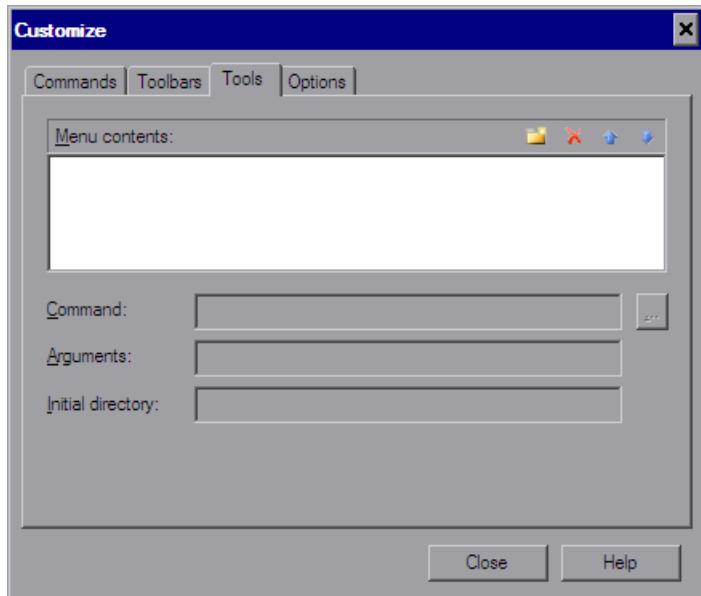
To access	<p>Do one of the following:</p> <ul style="list-style-type: none"> ➤ Select the Tools > Customize menu command and then click the Toolbars tab. ➤ Click the customize toolbar button , select Add or Remove Buttons > Customize, and then click the Toolbars tab. ➤ Right-click on the menu bar or any toolbar and select Customize and then click the Toolbars tab.
Important information	You can also show or hide toolbars using the View > Toolbars menu option or by right-clicking the toolbars area and selecting or deselecting a toolbar from the context menu.
Relevant tasks	<ul style="list-style-type: none"> ➤ "Float and Dock Toolbars" on page 1288 ➤ "Add a Command to a Toolbar or Menu" on page 1288
See also	<ul style="list-style-type: none"> ➤ "Customization Mode - Context Menu Options" on page 1296 ➤ "Tips for Customizing Toolbars and Menus" on page 1291 ➤ "Commands Tab (Customize Dialog Box)" on page 1297 ➤ "Options Tab (Customize Dialog Box)" on page 1299 ➤ "Tools Tab (Customize Dialog Box)" on page 1302

User interface elements are described below:

UI Elements	Description
Toolbars	A list of the toolbars in the QuickTest window, with the addition of Menu Bar . Select or deselect a check box to show or hide a toolbar.
Restore Selected	Restores the default layout for the selected toolbar or menu bar.
Restore All	Restores the default layout for all toolbars.
Show text labels	<p>Displays text labels for the buttons in the currently highlighted toolbar. For buttons that have a text label by default (for example, the Run button), clearing this check box restores the default display, and the text labels are still displayed.</p> <p>To turn off text labels for a toolbar, highlight the toolbar in the Toolbars area and deselect the check box.</p> <p>This check box is disabled for the menu bar toolbar.</p>

Tools Tab (Customize Dialog Box)

This tab enables you to add an item to the Tools menu so you can launch an application from the QuickTest menu.



To access	<p>Do one of the following:</p> <ul style="list-style-type: none">▶ Select the Tools > Customize menu command and then click the Tools tab.▶ Click the customize toolbar button , select Add or Remove Buttons > Customize, and then click the Tools tab.▶ Right-click on the menu bar or any toolbar and select Customize and then click the Tools tab.
See also	<ul style="list-style-type: none">▶ "Customization Mode - Context Menu Options" on page 1296▶ "Tips for Customizing Toolbars and Menus" on page 1291▶ "Commands Tab (Customize Dialog Box)" on page 1297▶ "Options Tab (Customize Dialog Box)" on page 1299▶ "Toolbars Tab (Customize Dialog Box)" on page 1300

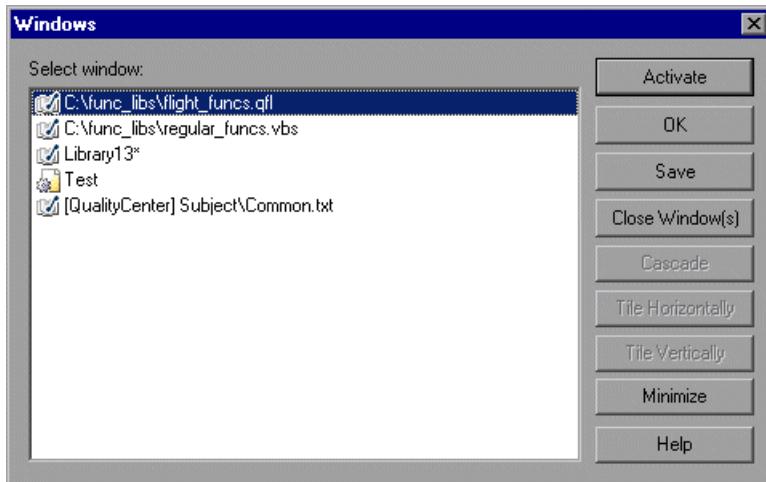
User interface elements are described below:

UI Elements	Description
	New. Adds a new item to the Tools menu, by adding a blank line to the Menu Contents area. Enter a name and functionality details for the new item.
	Delete. Deletes the item selected in the Menu Contents list.
	Move Item Up. Moves the selected item up in the Tools menu.
	Move Item Down. Moves the selected item down in the Tools menu.
Menu contents	A list of the items added to the tools menu.
Command Note: The Command box should contain only the file name and path for the application. If you want to add command line arguments, use the Arguments box. Tip: You can specify a document or other file associated with an application in the file system, for example, c:\tmp\a.txt. In this case, QuickTest automatically opens the specified file in the associated application (Notepad in this example). If you use this option, QuickTest ignores any defined program arguments.	The application for which you want to add an item to the Tools menu. Click the browse button and navigate to the application you want to add.
Arguments	Optional. Instructs QuickTest to open the application using the specified command line arguments.
Initial directory	Optional. Specifies the current working folder for the application. The initial directory is used by the application to search for related files. If an initial directory is not specified, the executable folder is used as the initial directory.

Windows Dialog Box

The **Windows** dialog box enables you to:

- Locate and activate (bring into focus) an open document window
- Select how the open document windows are arranged in the QuickTest window
- Select and save multiple open documents
- Select and close multiple windows, such as function library windows and the Start page



To access	Select Window > Windows .
Important information	<ul style="list-style-type: none">➤ QuickTest enables you to open and work on one test at a time.➤ You can open and work on multiple function libraries simultaneously.➤ You can open any function library, regardless of whether it is associated with the currently open test.

User interface elements are described below:

UI Elements	Description
Activate	Brings the selected document into focus in the QuickTest window.
OK	Closes the Windows dialog box.
Save	Saves the selected documents.
Close Window(s)	Closes the selected window(s). Note: You cannot close a single open test window.
Cascade	Arranges the selected documents in a cascading order that overlaps.
Tile Horizontally	Arranges the selected documents side-by-side horizontally, without overlapping.
Tile Vertically	Arranges the selected documents side-by-side vertically, without overlapping.
Minimize	Minimizes the selected documents.
Help	Displays the QuickTest Professional Help topic for this dialog box.

Troubleshooting and Limitations - QuickTest Window Layout

This section describes troubleshooting and limitations for the QuickTest layout on your screen.

Dual monitor support. When QuickTest is displayed on a secondary monitor, some drop-down lists may appear on the primary monitor.

Workaround: If you want to ensure that QuickTest is displayed on only one monitor, display QuickTest on your primary monitor.

36

Active Screen Pane

This chapter includes:

Concepts

- Active Screen Overview on page 1308

Tasks

- How to Use the Active Screen in Your Test on page 1312
- How to Modify the Active Screen Settings on page 1314

Reference

- Active Screen Pane User Interface on page 1316

Concepts

Active Screen Overview

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. For information on setting Active Screen recording options, see "Enhancing Your Tests" on page 387.

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.

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 "Active Screen Pane (Options Dialog Box)" on page 1434.

This section includes:

- "Active Screen Capture Settings" on page 1309
- "Active Screen Settings for QuickTest Add-ins" on page 1309
- "Active Screen Maintenance" on page 1310
- "Tips for Improving Active Screen Performance" on page 1311

Active Screen Capture Settings

- 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.
- You can decide 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.

Active Screen Settings for QuickTest Add-ins

Working with Web-based Applications

- QuickTest 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*.
- 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 "Active Screen Pane (Options Dialog Box)" on page 1434.

- The Active Screen stores DOM information in Internet Explorer format. Therefore, if you insert the **Object** property for a Web object when only Active Screen object data is available (the object is not displayed in an open application), then QuickTest IntelliSense displays the available native operations and properties for the Internet Explorer DOM, even if you recorded the object in another browser.

Working with Non-Web-based Applications

- 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.)

Active Screen Maintenance

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.

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.

For more information, see "How to Modify the Active Screen Settings" on page 1314.

It is also possible to update all Active Screen captures saved with a test using the Update Run Mode. For more information, see "Update Options Tab (Update Run Dialog Box)" on page 1271.

Tips for Improving Active Screen Performance

- **For 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 "Active Screen Pane (Options Dialog Box)" on page 1434.
- **For Web-based applications:** You can disable the screen capture of all steps in the Active Screen to improve recording time. For details, see "Active Screen Pane (Options Dialog Box)" on page 1434.
- 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.
- Tests without Active Screen files use significantly less disk space. For more information, see "How to Modify the Active Screen Settings" on page 1314.

Tasks

How to Use the Active Screen in Your Test

This task describes the different operations you can perform in the Active Screen pane, and includes the following steps:

- "View object information as it appeared during a recording session" on page 1312
- "Insert steps on objects in the application" on page 1312
- "Insert output values on objects in the application" on page 1313
- "Insert checkpoints on objects in the application" on page 1313

View object information as it appeared during a recording session

- 1 Right-click the object in the Active Screen, and select **View/Add Object**. The Object Selection dialog box opens. For details, see "Object Selection Dialog Box" on page 155.
- 2 Select the object you want to view and click **OK**. The Object Properties dialog box opens. For details, see "Object Properties Dialog Box" on page 234.

Insert steps on objects in the application

- 1 Right-click the object in the Active Screen, and select **Insert Output Value**. The Object Selection dialog box opens. For details, see "Object Selection Dialog Box" on page 155.
- 2 Select the object you want to view and click **OK**. The Output Value Properties dialog box opens. For details, see "Step Generator Dialog Box" on page 916.

Insert output values on objects in the application

- 1 Right-click the object in the Active Screen, and select **Step Generator**. The Object Selection dialog box opens. For details, see "Object Selection Dialog Box" on page 155.
- 2 Select the object you want to view and click **OK**. The Step Generator dialog box opens. For details, see "Step Generator Dialog Box" on page 916.

Insert checkpoints on objects in the application

- 1 Right-click the object in the Active Screen, and select one of the following:

- **Insert Standard Checkpoint**
- **Insert Bitmap Checkpoint**
- **Insert Text Checkpoint**

The Object Selection dialog box opens. For details, see "Object Selection Dialog Box" on page 155.

- 2 Select the object you want to view and click **OK**. The relevant Checkpoint Properties dialog box opens. For details, see "How to Insert a Checkpoint Step in a Test" on page 598.

For a user interface description, see "Active Screen Pane User Interface" on page 1316.

How to Modify the Active Screen Settings

The following steps describe how to modify different settings of the Active Screen:

- "Increase or decrease the Active Screen information saved with a test" on page 1314
- "Stop saving Active Screen information and reduce the disk space used by your test" on page 1315
- "Update a single Active Screen capture" on page 1315
- "Disable Active Screen capture to improve recording time" on page 1315

Increase or decrease the Active Screen information saved with a test

- 1 Modify the Active Screen capture preference in the Active Screen pane of the Options dialog box to capture the amount of information you need. For more information, see "Active Screen Pane (Options Dialog Box)" on page 1434.
- 2 Perform one of the following:
 - **Update Run Mode.** This operation enables you 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 "Update Options Tab (Update Run Dialog Box)" on page 1271.
 - **Re-record the steps.** This enables you to specify the objects 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 "Breakpoints" on page 1213.

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 "Save Test Dialog Box" on page 412.

Update a single 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 on using the pointing hand feature, see "Tips for Using the Pointing Hand" on page 157.
- 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**.

Disable Active Screen capture to improve recording time

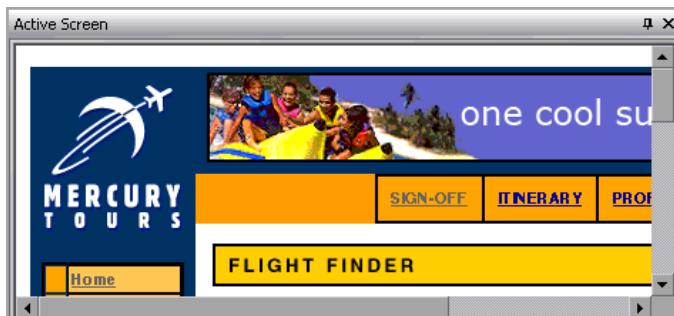
Click **Custom Level** to open "Custom Active Screen Capture Settings Dialog Box" on page 1438, and select the **Disable Active Screen Capture** option.

For more information, see "Tips for Improving Active Screen Performance" on page 1311.

Reference

Active Screen Pane User Interface

This pane enables you to view snapshots of your application as it appeared during a step in a recording session. It also enables you to parameterize object values and insert steps, checkpoints, methods, and output values for almost any object in the page at any time after the recording session.



To access	Do one of the following: <ul style="list-style-type: none"> ▶ Select View > Active Screen. ▶ Click the Active Screen button  in the toolbar.
Important information	<ul style="list-style-type: none"> ▶ You can increase or decrease the Active Screen information saved with your test. For details, see "Active Screen Pane (Options Dialog Box)" on page 1434. ▶ Additional context menu options may be available depending on the test object selected in the Active Screen.
Relevant tasks	<ul style="list-style-type: none"> ▶ "How to Modify the Active Screen Settings" on page 1314 ▶ "Adding and Deleting Test Objects in a Local or Shared Object Repository" on page 160
See also	<ul style="list-style-type: none"> ▶ "Active Screen Overview" on page 1308 ▶ "Custom Active Screen Capture Settings Dialog Box" on page 1438

User interface elements and context menu options are described below (unlabeled elements are shown in angle brackets):

UI Elements	Description
<active screen area>	The snapshot of the application at the time of recording the selected step. The object used in the step is highlighted.
Insert Standard Checkpoint	Inserts a standard checkpoint on the selected object. For details, see "How to Insert a Checkpoint Step in a Test" on page 598.
Insert Output Value	Inserts an output value on the selected object. For details, see "Output Values" on page 781.
View / Add Object	Opens the Object Properties dialog box, which enables you to view object properties and add the selected object to your local object repository. For details, see "Object Properties Dialog Box" on page 234.
Step Generator	Opens the Step Generator dialog box, which enables you to create a step using the selected object. For details, see "Step Generator Dialog Box" on page 916.
Insert Bitmap Checkpoint	Inserts a bitmap checkpoint on the selected object. For details, see "Bitmap Checkpoint Properties Dialog Box" on page 626.
Insert Text Checkpoint	Inserts a text checkpoint on the selected object. For details, see "Text / Text Area Checkpoint Properties Dialog Box" on page 660.

37

Available Keywords Pane

This chapter includes:

Concepts

- Available Keywords Pane Overview on page 1320

Tasks

- How to Work with the Available Keywords Pane on page 1321

Reference

- Available Keywords Pane User Interface on page 1322

Concepts

Available Keywords Pane Overview

The Available Keywords pane displays the keywords available to your test. It enables you to view the available objects and calls to functions, and also enables you to drag and drop them into your test. When you drag and drop an object into your action, QuickTest inserts a step with the default operation for that object. When you drag and drop a function into your test, QuickTest inserts a call to that function.

Examples of usage

- ▶ If you drag and drop a button object into your action, a step is added using the button with a **Click** operation (the default operation for a button object).
- ▶ If you drag and drop a function into your test, a comment and call to that function is added. The comment indicates that a call to the function was added to your test and indicates any necessary arguments. You then provide the arguments for that function to your test. QuickTest indicates the required arguments by displaying a tooltip in the Keyword view, and IntelliSense in the Expert view.

Tasks

How to Work with the Available Keywords Pane

This task describes how to work with the Available Keywords pane:

Add a step using a keyword

Drag the keyword into the document area. QuickTest inserts a step with the default operation for that object. Modify the operation and enter arguments for the step, as needed.

Open the keyword's resource file

Right-click the keyword and select **Open Resource**.

Open the keyword's resource file and point to the selected keyword in that file

Double-click the keyword.

Copy the selected keyword to the clipboard

Right-click the keyword and select **Copy Keyword**.

Sort keywords by resource



Click the **Sort by Resource** button.

Sort keywords alphabetically regardless of the resource file

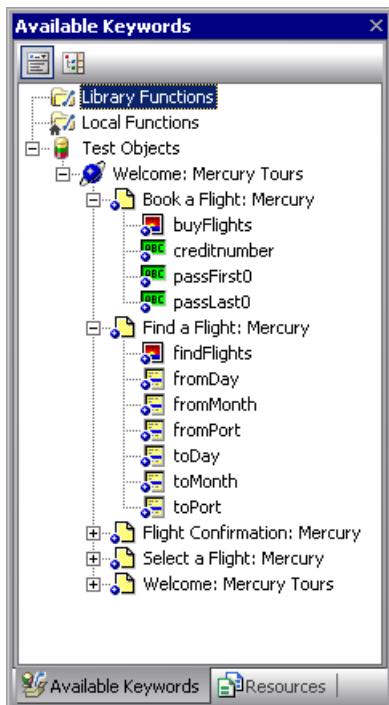


Click the **Sort by Keyword** button.

Reference

Available Keywords Pane User Interface

This pane displays the keywords available to your test sorted by resource or sorted by keyword, enabling you to view the available objects and calls to functions, and dragging and dropping them into your test.



To access	Use one of the following: <ul style="list-style-type: none"> ► Click the Available Keywords Pane button . ► Select View > Available Keywords.
Important information	Checkpoint and output value objects are not displayed in this pane, even if they are included in an associated object repository.

See also	For information on dragging and dropping test objects from other locations, see: <ul style="list-style-type: none"> ➤ "Object Repository Window" on page 237 ➤ "How to Manage Objects in Shared Object Repositories" on page 263
-----------------	---

User interface elements are described below:

UI Elements	Description
	<p>Sort by Keyword. Groups keywords by their type (library functions, local functions, objects) regardless of their resource.</p> <ul style="list-style-type: none"> ➤ All available functions are sorted alphabetically under the main Library Functions or Local Functions nodes. ➤ All available objects are grouped by the page or window in which they appear in the application, sorted by object type, and sorted alphabetically <p>Note: If two keywords have the same name, they are displayed according to the priority of their resources.</p>
	<p>Sort by Resource. Groups keywords by their type (library functions, local functions, objects) and then by the specific resource for that type. This enables you to view the resource within its resource file.</p> <ul style="list-style-type: none"> ➤ Function libraries are sorted alphabetically under the Library Functions or Local Functions nodes. The functions in each function library are then sorted alphabetically under the relevant function library node. ➤ Object repository files are sorted alphabetically under the Test Objects node. Objects in each object repository file are then grouped by the page or window in which they appear in the application, sorted by object type, and sorted alphabetically.
Open Resource	(Context-menu.) Opens the resource file in which the keyword is stored. You can also activate this option by double-clicking a keyword.
Copy Keyword	(Context-menu.) Copies the selected keyword to the clipboard.

38

Data Table Pane

This chapter includes:

Concepts

- Data Table Overview on page 1326
- Data Table Sheets on page 1327
- Data Table - Save Options on page 1330
- Data Table Objects, Methods, and Properties on page 1331
- Formulas in Data Tables on page 1332

Tasks

- How to Define a Data Table in Your Test on page 1334
- How to Manage Data Tables in Your Test on page 1336
- How to Insert Formulas into Data Tables for Use in Checkpoints on page 1338
- How to Import Data Using Microsoft Query on page 1339

Reference

- Data Table Pane User Interface on page 1340
- Database Query Wizard on page 1351

Troubleshooting and Limitations - Data Table on page 1354

Concepts

Data Table Overview

QuickTest enables you to insert and run steps that are driven by data stored in the data table. The data your test uses is stored in the **design-time** data table, which is displayed in the Data Table pane while you insert and edit steps.

The data table has the characteristics of a Microsoft Excel spreadsheet, enabling you to store and use data in its cells and also perform mathematical formulas within the cells.

You can use the data table provided with QuickTest, or you can use any Microsoft Excel (.xls) file.

You can use the DataTable, DTSheet and DTParameter utility objects to manipulate the data in any cell in the data table. For more information on these objects, see the **Utility** section of the *HP QuickTest Professional Object Model Reference*.

You can insert data table parameters and output values into your test. Using data table parameters and/or output values in a test enables you to create a **data-driven** test or action that runs several times using the data you supply.

How QuickTest Uses Data Tables

In each repetition, or **iteration**, of a run session, QuickTest uses a different value from the data table.

During the run session, QuickTest creates a **run-time** data table—a live version of the data table associated with your test. During the run session, QuickTest displays the run-time data in the Data Table pane so that you can see any changes to the data table as they occur.

When the run session ends, the run-time data table closes, and the Data Table pane again displays the stored design-time data table. Data entered in the run-time data table during the run session is not saved with the test. The final data from the run-time data table is displayed in the **Run-Time Data Table** in the Run Results Viewer. For more information on the run-time data table, see "Data Table Pane (Run Results Viewer)" on page 1131.

Tip: If QuickTest is integrating with HP ALM, you can use the data awareness feature to:

- use a different data table for each run iteration
- use the same data table in multiple tests
- perform other tasks.

To use this feature, make sure to save your data table parameters in the **Global** sheet.

For details, see "How to Data Drive a Test in HP ALM" on page 1621.

Data Table Sheets

When working with tests, the data table has two types of data sheets—**Global** and **Action**. You can access the different sheets by clicking the appropriate tabs below the data table.

- Global Sheet. You store data in the Global tab when you want it to be available to all actions in your test and you want the data to control the number of test iterations.

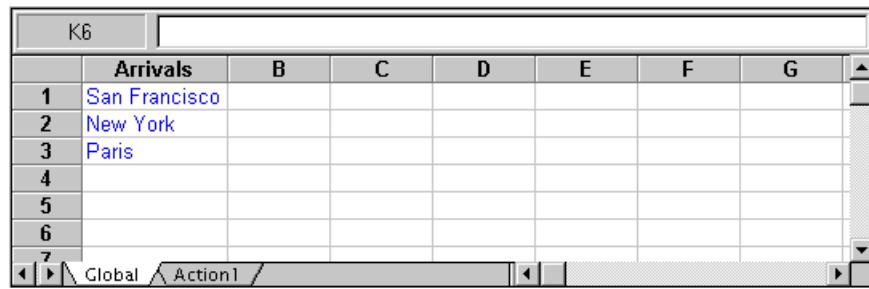
You must also store data in the Global tab if your QuickTest test is stored in HP ALM, and you want to use the data awareness feature. For details, see "Data Awareness in HP ALM" on page 1613.

- Action Sheets. You store data in the action's tab when you want to use the data in data table parameters for that action only and you want the data to control the number of action iterations.

For example, suppose you are creating a test on the sample Mercury Tours Web site. You might create one action for logging in, another for booking flights, and a third for logging out. You may want to create a test in which the user logs onto the site once, and then books flights for five passengers. The data about the passengers is relevant only to the second action, so it should be stored in the action sheet corresponding to that action.

Global Sheet

The Global sheet contains the data that replaces parameters in each iteration of the test. If you create a Global parameter called Arrivals, the Global sheet might look like this:



	Arrivals	B	C	D	E	F	G
1	San Francisco						
2	New York						
3	Paris						
4							
5							
6							
7	Global	Action1					

For more information on creating global parameters, see Chapter 22, "Parameterizing Values."

Action Sheets

Each time you add a new action to the test, a new **action sheet** is added to the data table. Action sheets are automatically labeled with the exact name of the corresponding action. The data contained in an action sheet is relevant for data table parameters in the corresponding action only. For example, if a test had the data table below, QuickTest would use the data contained in the Purchase sheet when running iterations on action parameter steps within the **Purchase** action.

H6		B	C	D	E	F	G
	Departure						
1	New York						
2	Paris						
3	Los Angeles						
4							
5							
6							
7							

Global Purchase

For more information on creating action parameters, see Chapter 22, "Parameterizing Values."

Note: If QuickTest is integrating with HP ALM, make sure to save your data table parameters in the Global Sheet (described on page 1328). For details, see "Data Awareness in HP ALM" on page 1613.

Data Table - Save Options

The data table contains the values that QuickTest substitutes for data table parameters when you run a test, as well as any other values or formulas you enter. Whenever you save your test, QuickTest automatically saves its data table as an .xls file.

When working with tests, the data table is saved with your test by default. You can save the data table in another location and instruct the test to use this data table when running a test. You specify a name and location for the data table in the Resources pane of the Test Settings dialog box. For more information on the Test Settings dialog box, see Chapter 46, "Individual Test Settings."

When to save the data table in a different location

- If you want to run the same test with different sets of input values. For example, you can test the localization capabilities of your application by running your test with a different data table file for each language you want to test. You can also vary the user interface strings that you check in each language by using a different environment parameter file each time you run the test. For more information, see Chapter 22, "Parameterizing Values."
- If you need the same input information for different tests. For example, you can test a Web version and a standard Windows version of the same application using different tests, but the same data table file.

When to save a run-time data table

If it is important for you to save the resulting data from the run-time data table, you can insert a `DataTable.Export` statement to the end of your test to export the run-time data table to a file. You can then import the data to the design-time data table using the data table's **File > Import From File** menu. Alternatively you can add a `DataTable.Import` statement to the beginning of your test to import the run-time data table that was exported at the end of the previous run session. For more information on these methods, see the *HP QuickTest Professional Object Model Reference*.

Saving data tables in a Quality Center project

When working with Quality Center, you must save the data table file in the Test Resources module in your Quality Center project before you specify the data table file in the Resources pane of the Test Settings dialog box. For more information, see "Resources Pane (Test Settings Dialog Box)" on page 1475.

You can add a new or existing data table file to your Quality Center project. Note that adding an existing data table file from the file system to a Quality Center project creates a copy of the file. Thus, once you save the file to the project, changes made to the Quality Center data table file will not affect the data table file in the file system and vice versa.



Data Table Objects, Methods, and Properties

QuickTest provides several data table methods that enable you to retrieve information on the run-time data table and to set the value of cells in the run-time data table. You enter these statements manually in the Expert View. For more information on working in the Expert View, see Chapter 27, "Working in the Expert View and Function Library Windows."

From a programming perspective, the data table is made up of three types of objects—DataTable, DTSheet (sheet), and DTPParameter (column). Each object has several methods and properties that you can use to retrieve or set values.

For more details on the data table methods, see the *HP QuickTest Professional Object Model Reference*.

Formulas in Data Tables

You can use Microsoft Excel formulas in your data table. This enables you to create contextually relevant data during the run session. You can also use formulas as part of a checkpoint to check that objects created on-the-fly (dynamically generated) or other variable objects in your Web page or application have the values you expect for a given context.

When you use formulas in a data table to compare values (generally in a checkpoint), the values you compare must be of the same type, for example integers, strings, and so forth. When you extract values from different places in your applications using different functions, the values may not be of the same type. Although these values may look identical on the screen, a comparison of them will fail, since, for example, 8.2 is not equal to "8.2".

For more information on using worksheet functions, see the Microsoft Excel documentation.

Formulas in Data Tables for Use in Iterations

You can enter formulas rather than fixed values in the cells of a parameter column.

For example, suppose you want to parameterize the value for a WebEdit object that requires a date value no earlier than today's date. You can set the cells in the **Date** column to the date format, and enter the =NOW() Excel formula into the first row to set the value to today's date for the first iteration.

Then you can use another formula in the remaining rows to enter the above date plus one day, as shown below. By using this formula you can run the test on any day and the dates will always be valid.

A2	=A1+1
	Date
1	1/3/2006
2	1/4/2006
3	1/5/2006
4	1/6/2006

TEXT or VALUE functions Used to Convert Values From One Type to Another

- **TEXT(value, format)**. Returns the textual equivalent of a numeric value in the specified format, so that, for example the formula =TEXT(8.2, "0.00") is "8.20".
- **VALUE(string)**. Returns the numeric value of a string, so that, for example, =VALUE("\$8.20") is 8.20.

For more information on using parameters, see Chapter 22, "Parameterizing Values."

Formulas in Data Tables for Use in Checkpoints

You can use a formula in a checkpoint to confirm that an object created on-the-fly (dynamically generated) or another variable object in your Web page or application contains the value it should for a given context. For example, suppose a shopping cart Web site displays a price total. You can create a text checkpoint on the displayed total value and use a data table formula to check whether the site properly computes the total, based on the individual prices of the products selected for purchase in each iteration.

When you use the data table formula option with a checkpoint, QuickTest creates two columns in the data table. The first column contains a default checkpoint formula. The second column contains the value to be checked in the form of an output parameter. The result of the formula is Boolean—TRUE or FALSE.

A1	= \$B1 = "337"
1	TRUE
2	337

A FALSE result in the checkpoint column during a test run causes the test to fail.

After you finish adding the checkpoint, you can modify the default formula in the first column to perform the check you need.

Tasks

How to Define a Data Table in Your Test

This task describes the different options for defining a new data table for your test.

This task includes:

- "Add an external data table file" on page 1334
- "Import data from a database" on page 1335
- "Manually enter or import information in the local data table" on page 1335
- "Add a data table file to your Quality Center project" on page 1335

Add an external data table file

Configure the location of the data table in the Resources pane of the Test Settings dialog box (**File > Settings > Resources** node). For more information, see "Resources Pane (Test Settings Dialog Box)" on page 1475.

If you select an external file as your data table, make sure that:

- The column names in the external data table match the parameter names in the test.
- The sheets in the external Data Table match the action names in the test.

Import data from a database

Import data into a data table sheet using Microsoft Query, or by specifying an SQL statement. For more information, see the "How to Import Data Using Microsoft Query" on page 1339.

Manually enter or import information in the local data table

Edit information in the data table by typing directly into the table cells. You can also import data saved in Microsoft Excel, tabbed text file (.txt), or ASCII format.

To import an Excel file select **File > Import from file** from the data table commands.

To import a single sheet or a tabbed text file, select **Sheet > Import > From file** from the data table commands.

For more details on data table commands, see "Data Table Pane User Interface" on page 1340.

Add a data table file to your Quality Center project

- 1 Make sure you have a Microsoft Excel file in your file system with an **.xls** extension.
- 2 In Quality Center, create a new data table resource and then upload the **.xls** file you created in the previous step to the project's Test Resources module. For more information, see the *HP Application Lifecycle Management User Guide*.
- 3 In QuickTest Test Settings dialog box (**File > Settings > Resources** node), select **Other location** and click the browse button to locate the data table file.
- 4 Create your test. When you save the test, QuickTest saves the data table file to the Quality Center project.

How to Manage Data Tables in Your Test

This task describes how to manage the data table in your test.

This task includes:

- "Define the number of iterations for an action or test" on page 1336
- "Change a column name" on page 1337
- "Use an autofill list" on page 1337

Define the number of iterations for an action or test

- **For actions:** Open the **Run** tab of the **Action Call Properties** dialog box (**Edit > Action > Action Call Properties > Run** tab).
- **For tests:** Open the **Run** pane of the **Settings** dialog box (**File > Settings > Run** pane).
- If you want to prevent QuickTest from running an iteration on a row when the **Run on all rows** option is selected, you must delete the entire row from the data table. To do this, do one of the following:
 - Select the row, right-click in the table, and select **Edit > Delete** from the data table's context menu (or use **CTRL+K**). This restores the bottom grid line from black to gray.
 - Use the **Clear** option from the table's **Edit** menu (or **CTRL+X**).
 - Select a cell and press **Delete** on the keyboard. The data is deleted from the cells, but the row is not deleted and the black line remains. This means that QuickTest will run an iteration for this row even though there is no data in it.

Change a column name

For details, see "Change Parameter Name Dialog Box (Data Table)" on page 1350.

Use an autofill list

- 1 Select the type of autofill list from the AutoFill Lists Dialog Box (Data Table), described on page 1348.
- 2 Enter the first item into a cell in the table (item names are case-sensitive).
- 3 Fill the cells by doing one of the following:
 - ▶ Drag the cursor, from the bottom right corner of the cell up or right to fill the next row or column in the sheet.
 - ▶ Highlight the item and press ENTER to automatically fill the next row of the sheet.
 - ▶ Highlight the item and press TAB to automatically fill the next column of the sheet.

How to Insert Formulas into Data Tables for Use in Checkpoints

This task describes how to insert formulas into the data table for use in a checkpoint.

To use a formula in a checkpoint:

- 1 Select the object or text for which you want to create a checkpoint and open the Insert Checkpoint dialog box as described in Chapter 15, "Checkpoints Overview."
- 2 In the **Configure value** area, click **Parameter**.
- 3 Set the parameter options, as described in the "Parameter Options Dialog Box (Data Table)" on page 763.
- 4 Specify your other checkpoint setting preferences as described in Chapter 15, "Checkpoints Overview."
- 5 Highlight the value in the first (formula) column to view the formula and modify the formula to fit your needs.
- 6 If you want to run several iterations, add the appropriate formula in subsequent rows of the formula column for each iteration in the test or action.

Tip: You can encode passwords to use the resulting strings as method arguments or data table parameter values. For more information, see "Password Encoder Tool" on page 519.

You can also encrypt strings in data table cells using the **Encrypt** option in the Data Table menu. For more information, see "Data Menu (Data Table)" on page 1345.

How to Import Data Using Microsoft Query

You can use Microsoft Query to choose a data source and define a query within the data source. For information on supported versions of Microsoft Query, see the *HP QuickTest Professional Product Availability Matrix*, available from the Documentation Library Home page or the root folder of the QuickTest Professional DVD.

To choose a data source and define a query in Microsoft Query:

- 1** Open the Database Query Wizard (described on page 1351) and select **Create query using Microsoft Query**.
- 2** When Microsoft Query opens, choose a new or an existing data source.
- 3** Define a query.
- 4** In the Finish screen of the Query Wizard, select one of the following:
 - **Exit and return to QuickTest > Finish** to exit Microsoft Query.
 - **View data or edit query in Microsoft Query > Finish** to view the data.

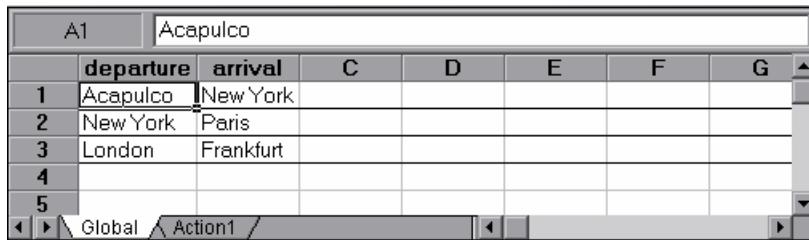
After viewing or editing the data, select **File > Exit and return to QuickTest** to close Microsoft Query and return to QuickTest.

For more information on working with Microsoft Query, see the Microsoft Query documentation.

Reference

Data Table Pane User Interface

You use the data table in the same way as a Microsoft Excel spreadsheet, including inserting formulas into cells.



To Access	<p>In the main QuickTest window, do one of the following:</p> <ul style="list-style-type: none"> ▶ Select View > Data Table. ▶ Click the Data Table  toolbar button.
Important information	<ul style="list-style-type: none"> ▶ You access the data table menus and commands by right-clicking anywhere in the data table. ▶ Each row in the table represents the set of values that QuickTest submits for the parameterized arguments during a single iteration of the test or action. ▶ Each column in the table represents the list of values for a single parameterized argument. The column header is the parameter name. ▶ Combo box and list cells, conditional formatting, and other special cell formats are not supported in the data table. ▶ You can also enter data and formulas in cells in the columns that are not intended for use with data table parameters (the columns that do not have a parameter name in the column header).

Relevant tasks	<ul style="list-style-type: none"> ➤ "How to Define a Data Table in Your Test" on page 1334 ➤ "How to Manage Data Tables in Your Test" on page 1336 ➤ "How to Insert Formulas into Data Tables for Use in Checkpoints" on page 1338
See also	<ul style="list-style-type: none"> ➤ "Data Table Parameters" on page 731 ➤ For information on supported versions of Microsoft Excel, see the <i>HP QuickTest Professional Product Availability Matrix</i>, available from the Documentation Library Home page or the root folder of the QuickTest Professional DVD.

The Data Table pane provides the following shortcut menu (right-click) commands:

- "General Commands (Data Table)" on page 1342
- "File Menu (Data Table)" on page 1342
- "Sheet Menu (Data Table)" on page 1343
- "Edit Menu (Data Table)" on page 1343
- "Data Menu (Data Table)" on page 1345
- "Format Menu (Data Table)" on page 1345

This section also includes:

- "Data Table Specifications" on page 1346
- "Guidelines for Working with the Data Table" on page 1347
- "AutoFill Lists Dialog Box (Data Table)" on page 1348
- "Change Parameter Name Dialog Box (Data Table)" on page 1350

 **General Commands (Data Table)**

The **General** shortcut (right-click) menu includes the following commands:

Command	Shortcut Key	Description
Switch between Data Table sheets	CTRL+PAGE UP/ PAGE DOWN	Switches through the data table sheets when the data table is in focus.

 **File Menu (Data Table)**

The **File** shortcut (right-click) menu includes the following commands:

Command	Description
Import From File	<p>Imports an existing Microsoft Excel into the data table. This command will import all the sheets in the selected Microsoft Excel file. If you want to import only one sheet from an existing Microsoft Excel file or a tabbed text file, use the Sheet > Import > From File command described below.</p> <p>Notes:</p> <ul style="list-style-type: none"> ► The table file you import replaces all data in all sheets of the table. The first row in each Microsoft Excel sheet also replaces the column headers in the corresponding data table sheet. It is therefore essential that the first row of your Microsoft Excel sheet exactly matches the parameter names in your test, and that the file contains at least the same number of sheets as the current data table. ► If you import a Microsoft Excel table containing combo box or list cells, conditional formatting, or other special cell formats, the formats are not imported and the cells are displayed in the data table with a fixed value.
Export	Exports the table to a specified Microsoft Excel (.xls) file.
Print	Prints the entire table or the selected sheet.

 **Sheet Menu (Data Table)**

The **Sheet** shortcut (right-click) menu includes the following commands:

Command	Description
Import > From File	<p>Imports a tabbed text file or a single sheet from an existing Microsoft Excel file into the table.</p> <p>Note: The sheet you import replaces all data in the currently selected sheet of the table, and the first row in the Excel sheet replaces the column headers in the corresponding data table sheet. It is therefore essential that the first row of your Microsoft Excel sheet exactly matches the parameter names in your test.</p>
Import > From Database	Imports data from the specified database to the current sheet.
Export	Exports the current sheet of the data table to a specified Microsoft Excel (.xls) file.

 **Edit Menu (Data Table)**

The **Edit** shortcut (right-click) menu includes the following commands:

Command	Shortcut Key	Description
Cut	CTRL+X	Cuts the table selection and places it on the Clipboard.
Copy	CTRL+C	Copies the table selection and places it on the Clipboard.
Paste	CTRL+V	Pastes the contents of the Clipboard to the current table selection.
Clear > All	N/A	Clears contents and formatting from the current selection.
Clear > Formats	N/A	Clears the formatting from the current selection.
Clear > Contents	CTRL+DEL	Clears the contents from the current selection.

Command	Shortcut Key	Description
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. Note that this option is available only when a row or column heading is selected.
Delete	CTRL+K	Deletes the entire current row or column selection. Cells adjacent to the deleted cells are shifted to fill the space left by the vacated cells. Note that this option is available only when a row or column heading is selected.
Fill Right	CTRL+R	Copies data in the left-most cell of a selected range to all the cells to the right of that left-most cell within the selected range.
Fill Down	CTRL+D	Copies data in the top cell of a selected range to all cells below that top cell within the selected range.
Find	CTRL+F	Finds a cell containing specified text. You can search by row or column in the table and specify to match case and/or find entire cells only. You can also search for formulas or values.
Replace	CTRL+H	Finds a cell containing specified text and replaces it with different text. You can search by row or column in the table and specify to match case and/or to find entire cells only. You can also search for formulas or values. You can also replace all instances of the found text.
Go To	N/A	Goes to a specified cell. This cell becomes the active cell. You must enter the column and row number of the cell.

 **Data Menu (Data Table)**

The **Data** shortcut (right-click) menu includes the following commands:

Command	Shortcut Key	Description
Recalc	F9	Recalculates any formula cells in the table.
Sort	N/A	Sorts a selection of cells by row or column and keys in ascending or descending order.
AutoFill List	N/A	Opens the AutoFill Lists dialog box (described on page 1348).
Encrypt	N/A	<p>Encodes the text in the selected cells. Note that you cannot decrypt data that has been encrypted.</p> <p>You can also use the Password Encoder to encrypt any text string. This can be useful for entering encrypted strings as method arguments in the Expert View. For more information, see "Password Encoder Tool" on page 519.</p>

 **Format Menu (Data Table)**

The **Format** shortcut (right-click) menu includes the following commands:

Command	Description
General	Sets format to General. The General format displays numbers with as many decimal places as necessary and no commas.
Currency(0)	Sets format to currency with commas and no decimal places. Note that QuickTest uses the currency symbol defined in your Windows Regional Settings dialog box.
Currency(2)	Sets format to currency with commas and two decimal places. Note that QuickTest uses the currency symbol defined in your Windows Regional Settings dialog box.
Fixed	Sets format to fixed precision with commas and no decimal places.

Command	Description
Percent	Sets format to percent with no decimal places. Numbers are displayed as percentages with a trailing percent sign (%).
Fraction	Sets format to fraction in numerator denominator form, e.g. 1/2.
Scientific	Sets format to scientific notation with two decimal places.
date (dynamic)	Sets format to Date with the M/D/YY format.
Time: h:mm AM/PM	Sets format to Time with the h:mm AM/PM format.
Custom Number	Sets format to a custom number format that you specify. This option enables you to set special and customized formats for percentages, currencies, dates, times, and so forth.

Data Table Specifications

The main limitations for working with the data table are listed below:

Item	Details
Maximum worksheet size	65,536 rows by 256 columns
Maximum number of worksheets	256 (255 sheets in addition to the Global data table).
Column width	0 to 255 characters
Text length	16,383 characters
Formula length	1024 characters
Number precision	15 digits
Largest positive number	9.99999999999999E307
Smallest positive number	1E-307
Largest negative number	-1E-307
Smallest negative number	-9.99999999999999E307
Maximum number of names per workbook	Limited by available memory

Item	Details
Maximum length of name	255
Maximum length of format string	255
Maximum number of tables (workbooks)	Limited by system resources (windows and memory)
Limitations	<ul style="list-style-type: none"> ▶ The use of colors and formatting in the data table is not supported. ▶ Complex and/or nested formulas are not supported in the data table. ▶ Combo box and list cells, conditional formatting, and other special cell formats are not supported in the data table.

Guidelines for Working with the Data Table

- ▶ When you add data to the data table, you must enter the data in rows from top to bottom and left to right—you cannot leave a gap of an entire row or column. For example, if there is data in row 1, you cannot enter data in a cell in row 3 until you have entered data in row 2. Similarly, if there is data in column A, you cannot enter data in column C until you have entered data in column B.
- ▶ The value returned from the data table is always converted to a string. If you want the value to be converted to something other than a string, you can use VBScript conversion functions, such as CInt, CLng, CDbl, and so forth. For example:

```
Window("Flight Reservation").WinComboBox("Fly From:").Select
CInt(DataTable("ItemNumber", dtGlobalSheet))
```

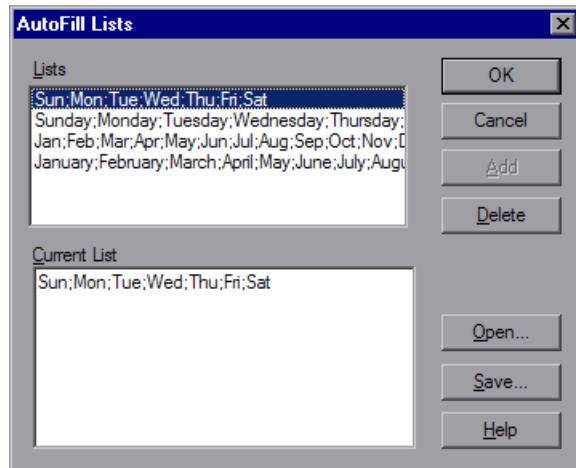
- When you add content to a data table cell, QuickTest changes the color of the row's bottom grid line from gray to black. When you run your test using the **Run on all rows** option (defined in **File > Settings > Run** pane, or **Edit > Action > Action Call Properties > Run** tab), QuickTest runs one iteration for each row whose bottom grid line is black. For details on modifying the number of iterations, see "How to Manage Data Tables in Your Test" on page 1336.
- If you change a data parameter (column header), you must also update the relevant name of the corresponding parameter wherever it is used in QuickTest, for example, parameterized argument values, checkpoint or output values, action parameters, and repository parameters.

If your test is stored in HP ALM, you must also update the relevant Quality Center parameter mappings in each configuration defined for your test.

For details, see "Change Parameter Name Dialog Box (Data Table)" on page 1350.

AutoFill Lists Dialog Box (Data Table)

This dialog box enables you to create, edit, or delete an autofill list. An autofill list contains frequently-used series of text such as months and days of the week.



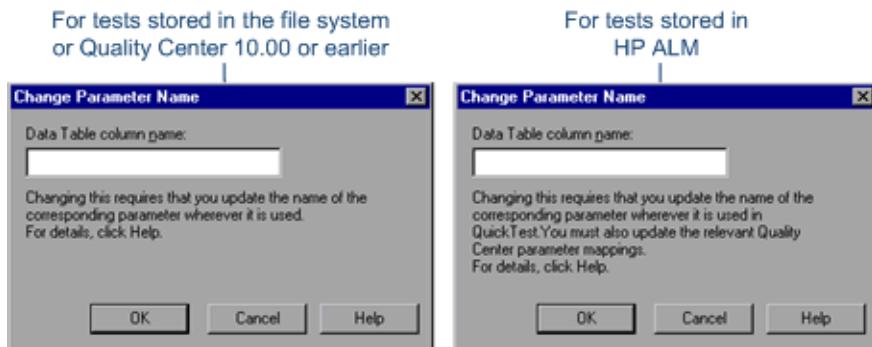
To Access	In the data table (described on page 1340), right-click a cell or cell range and select Data > AutoFill List .
Important information	AutoFill items are case-sensitive.
Related Tasks	"How to Manage Data Tables in Your Test" on page 1336
See Also	"Data Table Overview" on page 1326

User-interface elements are described below:

UI Element	Description
Lists	The lists that are available in your project. Four default lists are included.
Current List	The selected list. This pane can be used to create a new list. Separate the items in a new list with a semi-colon.
Add	Adds a new list to the Lists box.
Delete	Deletes a list from the Lists box.
Open	Opens the Open dialog box, in which you can browse to a previously created list.
Save	Opens the Save As dialog box, in which you can save a new list.

Change Parameter Name Dialog Box (Data Table)

This dialog box enables you to change the data table parameter name, which is defined in the column header in the data table.



To access	In the data table (described on page 1340), double-click a column header cell.
-----------	--

User-interface elements are described below:

UI Element	Description
Data Table column name	<p>The data table parameter name, which is defined in the column header in the data table.</p> <p>For a list of naming conventions, see "Naming Conventions" on page 1779.</p> <p>Note: If you modify the column header, you must also:</p> <ul style="list-style-type: none"> ▶ Change the name of the corresponding data table parameter wherever it is used (for example, parameterized argument values, checkpoint or output values, action parameters, and repository parameters). ▶ For tests stored in HP ALM: In the Test Plan module, you must also update the parameter mappings in each configuration defined for your test.

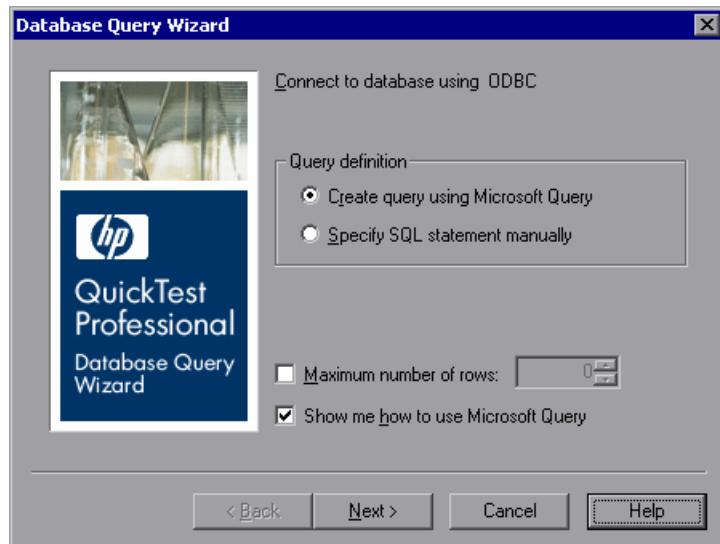
 **Database Query Wizard**

This wizard enables you to import data from a database to a Data Table sheet.

To access	In the Data Table Pane User Interface (described on page 1340), right-click the sheet to which you want to import the data and select Sheet > Import > From Database .
Important information	<ul style="list-style-type: none"> ▶ You can install Microsoft Query from the custom installation option of Microsoft Office. ▶ QuickTest takes several seconds to capture the database query and restore the QuickTest window. The resulting data from the database query is displayed in the data table. ▶ Contrary to importing an Excel file (File > Import From File), existing data in the data table is not replaced when you import data from a database. If the database you import contains a column with the same name as an existing column, the database column is added as a new column with the column name followed by a sequential number. For example, if your data table already contains a column called departures, a database column by the same name would be inserted into the data table as departures1.
Relevant tasks	<ul style="list-style-type: none"> ▶ "How to Manage Data Tables in Your Test" on page 1336 ▶ "How to Import Data Using Microsoft Query" on page 1339
Wizard map	The Database Query Wizard contains: Connect to database using ODBC (page 1352) > Specify SQL statement (page 1353)

Connect to Database Using ODBC Page (Database Query Wizard)

This wizard page enables you to select a data source to import from.



Wizard map	The Database Query Wizard contains: Connect to database using ODBC > Specify SQL statement
-------------------	---

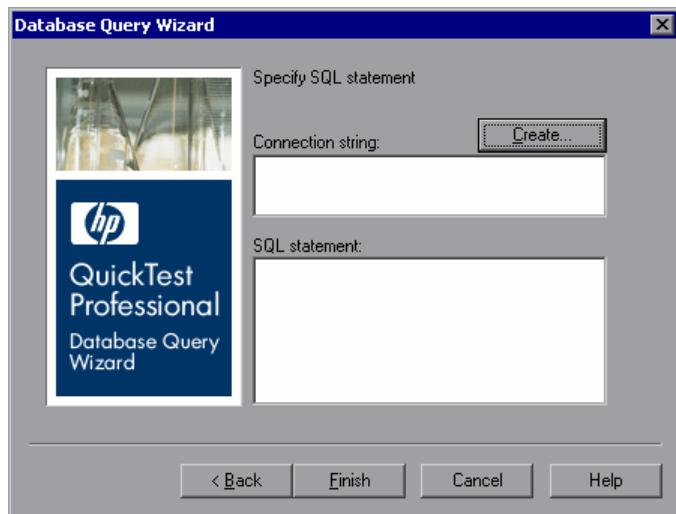
User interface elements are described below:

UI Elements	Description
Create query using Microsoft Query	Opens Microsoft Query, enabling you to create a new query. After you finish defining your query, you exit back to QuickTest. This option is only available if you have Microsoft Query installed on your computer.
Specify SQL statement manually	Opens the Specify SQL statement page in the wizard, which enables you to specify the connection string and an SQL statement.

UI Elements	Description
Maximum number of rows	The maximum number of database rows to import. You can specify a maximum of 32,000 rows.
Show me how to use Microsoft Query	Displays an instruction screen before opening Microsoft Query when you click Next . (Enabled only when Create query using Microsoft Query is selected).

Specify SQL Statement Page (Database Query Wizard)

This wizard page enables you to define the SQL statement to import into the data table.



Wizard map	The Database Query Wizard contains: Connect to database using ODBC > Specify SQL statement
-------------------	--

User interface elements are described below:

UI Elements	Description
Connection string	The connection string to import.
Create	Opens the ODBC Select Data Source dialog box. You can select a .dsn file in the ODBC Select Data Source dialog box or create a new .dsn file to have it insert the connection string in the box for you.
SQL statement	The details of the SQL statement.

Troubleshooting and Limitations - Data Table

This section describes troubleshooting and limitations for working with the QuickTest data table.

Incompatible Microsoft Excel Files

Microsoft Excel files may not be compatible with QuickTest. This may occur because the **.xls** file you are attempting to import contains features or functionality that is unavailable from QuickTest.

Workaround 1: Create and import **.xls** files that contain only functionality that is available from the QuickTest Data Table pane itself. For a list of the available functionality in the Data Table pane, see the relevant shortcut menu in "Data Table Pane User Interface" on page 1340.

Workaround 2: If you need to alter an existing **.xls** file, you can use one of the following options:

- Determine what are the most recent changes to the **.xls** file, and undo or remove them.
- Remove any existing macros.

- In each worksheet in the .xls file, select all rows and columns, and select **Edit > Clear > Formats** to remove all formatting from the worksheet.
 - Create a new .xls file, and copy the data from the non-compatible file. You can do this by using the standard **copy** and **paste** operations, or you can do this by saving each worksheet as a tab delimited .txt file (**File > Save As**), opening each .txt file in a text editor, and copying the data from that file to a new .xls file.
-

Caution:

- Make sure to back up your .xls file before changing or removing any data or functionality.
 - Even though some of the operations in this workaround can be automated, due to the way that Microsoft Excel exports data to tab delimited text files, the actual exported data may not exactly match the original data. Therefore, make sure to verify all exported data.
-

Importing/Exporting a Data Table

If you import a Microsoft Excel table containing a combo box, list cells, conditional formatting, or other special cell formats, the formats are not imported and the cells are displayed in the data table with fixed values. Make sure to verify all imported data from cells containing any of these special formats because the original formatting may affect the values, and the values may not be imported correctly.

Data Table Cell Content

Entering a very large number in the data table may cause unexpected behavior.

39

Information Pane

This chapter includes:

Concepts

- Information Pane Overview on page 1358

Tasks

- How to Resolve VBScript Syntax Errors in the Information Pane on page 1359

Reference

- Information Pane User Interface on page 1360

Concepts

Information Pane Overview

The Information pane lists the VBScript syntax errors in the active test document.

When you switch from the Expert View to the Keyword View or when you save a test with the Expert View displayed, 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 view a description of each of the VBScript errors in the VBScript Reference. For more information, select **Help > QuickTest Professional Help > VBScript Reference > VBScript > Reference > Errors > VBScript Syntax Errors**.

Tasks

How to Resolve VBScript Syntax Errors in the Information Pane

This task describes the different operations that can be performed in the Information Pane.

Check VBScript syntax errors using the Check Syntax option

In the Expert View, do one of the following:



- Click the **Check Syntax** button.
- Select **Tools > Check Syntax**.

Display the currently incorrect syntax

Move the pointer over the description of a syntax error.

Navigate to the line containing a specific syntax error

Double-click the syntax error in the Information pane.

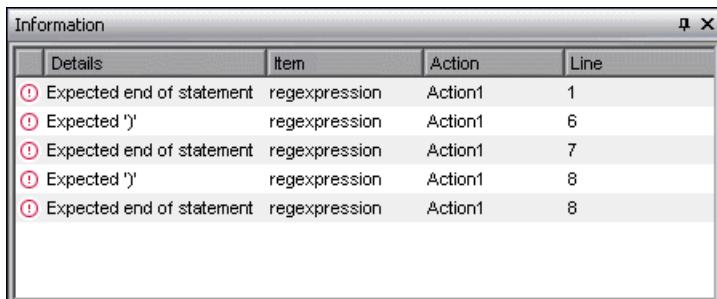
Customize the Information Pane display

- Resize the columns in the Information pane to make the information more readable by dragging the splitter between the column headers.
- Sort the details in the Information pane in ascending or descending order by clicking the column header.

Reference

Information Pane User Interface

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



To access	Select View > Information Pane .
Important information	You can double-click a syntax error to locate the error in the test script or function library, and then correct it. For more information, see "Handling VBScript Syntax Errors" on page 972.
Relevant tasks	"How to Resolve VBScript Syntax Errors in the Information Pane" on page 1359
See also	"Basic VBScript Syntax" on page 970

User interface elements are described below:

UI Elements	Description
Details	<p>The description of the syntax error. For example, if you opened a conditional block with an If statement but did not close it with an End If statement, the description is Expected 'End If'.</p> <p>Note: In certain cases, QuickTest is unable to identify the exact error and displays a number of possible error conditions, for example: Expected 'End Sub', or 'End Function', or 'End Property'. Check the statement at the specified line to clarify which error is relevant in your case.</p>
Item	The name of the test or function library containing the problematic statement.
Action	The name of the action containing the problematic statement.
Line	The line containing the syntax error. Lines are numbered from the beginning of each action or function library.

40

Missing Resources Pane

This chapter includes:

Concepts

- ▶ Missing Resources Overview on page 1364

Tasks

- ▶ How to Handle a Missing Resource on page 1367
- ▶ How to Locate a Missing Action on page 1368

Reference

- ▶ Missing Resources Pane User Interface on page 1371

Concepts

Missing Resources Overview

Each time you open a test, QuickTest verifies that the resources specified for the test are available.

If one or more resources cannot be found, QuickTest opens the Missing Resources pane, if the pane is not already open. The Missing Resources pane provides a list of all resources that are currently unavailable, along with the location where QuickTest expected to find the resource, when available. If one of the resources listed in this pane is unavailable during a run session, the test may fail.

Note: If a test is associated with a resource that is located on a password-protected network host, the resource will be listed as missing in the Missing Resources pane unless you open the resource prior to opening the test.

The Missing Resources pane enables you to locate or remove the missing resources from your test.

After you successfully handle a missing resource, QuickTest removes it from the pane.

This section also includes:

- "Missing Actions" on page 1365
- "Missing Function Libraries" on page 1366
- "Missing Shared Object Repositories" on page 1366
- "Missing Recovery Scenarios" on page 1366
- "Unmapped Shared Object Repository Parameter Values" on page 1366

Missing Actions

If your test contains a call to one or more actions that cannot be found, QuickTest lists these actions in the Missing Resources pane.

If your test contains calls to more than one missing action, when you locate the missing action in another test, QuickTest may identify additional missing actions that are found in the same test. This can occur, for example, if the source test containing the actions that are being called was renamed or was moved to another folder. In such cases, QuickTest displays a message box prompting you to map these missing actions as well.

You can instruct QuickTest to locate these actions simultaneously, or you can handle each call to a missing action individually.

Similarly, if you select to remove an action, and your test contains additional missing actions in the same test, QuickTest opens a message box asking whether you want to remove all the actions with the same path.

Note: If a test is opened in read-only format, you cannot view or map its missing actions.

For more information, see "How to Locate a Missing Action" on page 1368.

Missing Function Libraries

If you choose to remove rather than locate a missing function library, keep in mind that removing this association does not remove calls to those functions from your test steps. Therefore, make sure that you handle any calls to functions in removed function libraries.

Missing Shared Object Repositories

You use the Associate Repositories dialog box to resolve a missing object repository by associating a new object repository with your test. The missing object repository will still be associated with your test and will still appear in the Missing Resources pane. To remove the missing object repository from the Missing Resources pane and your test, you must use the Remove Repository option of the Associate Repositories dialog box.

For more details, see "Associate Repositories Dialog Box" on page 229.

Missing Recovery Scenarios

If your test contains more than one missing recovery scenario, then when you locate a missing scenario in a recovery file, QuickTest may identify additional missing scenarios in that file. You can instruct QuickTest to locate these missing recovery scenarios simultaneously, or you can handle each missing scenario individually.

Unmapped Shared Object Repository Parameter Values

Every repository parameter used in your test must have a specified value. This can be either a default value that was specified when the parameter was created, or it can be a value that you specify in your test. (For more information on repository parameters, see "Working with Repository Parameters" on page 252.)

When you open a test that uses an object repository with a repository parameter without a value, QuickTest indicates this by displaying a **Repository Parameters** item in the Missing Resources pane.

For example, suppose your application contains an edit box whose name property changes depending on a selection made in a previous screen. If you parameterized the value of the name property in the object repository using a repository parameter, but a default value was not defined for the repository parameter, you need to define a value for it. You can map it to a Data Table parameter, an environment variable, a random number, or a test or action parameter. You can also define a constant value for it, and so forth.

Tasks



How to Handle a Missing Resource

This task describes what to do if the Missing Resources Pane indicates that your test has one or more missing resources.

This task includes the following steps:

- "Determine the cause of the problem" on page 1367
- "Determine whether to resolve the problem or remove the association of this resource with your testing asset" on page 1367
- "Use the available options in the Missing Resources pane to handle the problem" on page 1368
- "Results" on page 1368

1 Determine the cause of the problem

Look at the details of the item you want to resolve in the Missing Resources pane and analyze what caused the problem. For example, the resource file may have been renamed, move to a different folder, or deleted accidentally or intentionally.

2 Determine whether to resolve the problem or remove the association of this resource with your testing asset

Decide whether the missing resource is still necessary for your test.

3 Use the available options in the Missing Resources pane to handle the problem

In the Missing Resources Pane, right-click the missing resource item and select **Locate**, **Resolve**, or **Remove** as appropriate. For more information, see "Missing Resources Pane User Interface" on page 1371.

4 Results

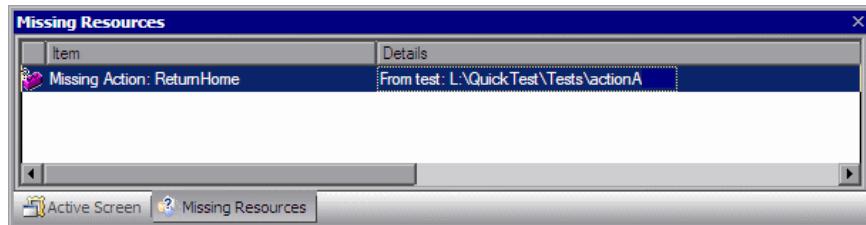
After you address the problem, the resource is removed from the Missing Resources pane.

A missing object repository will still be associated with your test and will still appear in the Missing Resources pane. To remove the missing object repository from the Missing Resources pane and your test, you must use the Remove Repository option of the Associate Repositories dialog box.

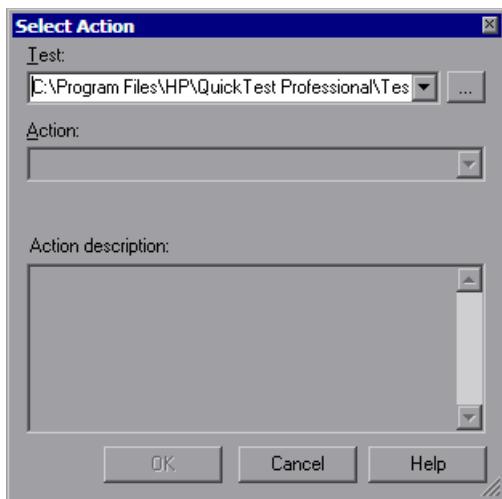
How to Locate a Missing Action

To locate a missing action:

- 1 In the Missing Resources pane, double-click the action you want to locate or right-click the action and select **Locate** from the context-sensitive menu.



The Select Action dialog box opens.



When the Select Action dialog box opens, the **Test** box displays either the name of the test containing the missing action (if QuickTest can identify the source test), or <**Current Test**>.

Note: If the missing action is a nested action that is called from another test, you cannot use the **Locate** button to browse to that action. Instead, you must resolve the missing action from within the external test. For example, if ActionAA (in TestA) calls ActionBB (from TestB), and ActionBB calls ActionCC (from TestC), if you open TestA and the call to ActionCC is missing, then you can only resolve the missing action by opening TestB and locating ActionCC. (You cannot resolve it from within TestA.)

- 2 Click the **Browse** button to find the test that contains the action you want to locate. The **Action** box displays all reusable actions in the test you select.
-

Notes:

- When you select a test, the **Test** box is renamed to **From test**. If the test you select contains reusable actions, these are listed in the **Action** box.
 - You can enter a Quality Center folder or a relative path in the **Test/From test** box. If you enter a relative path, QuickTest searches for the test in the folders listed in the Folders pane of the Options dialog box. For more information, see "Folders Pane (Options Dialog Box)" on page 1431 and "Relative Paths in QuickTest" on page 391. If you are working with the Resources and Dependencies model with Quality Center 10.00 or HP ALM, specify an absolute Quality Center path. For more information, see "Relative Paths and Quality Center" on page 1656.
-

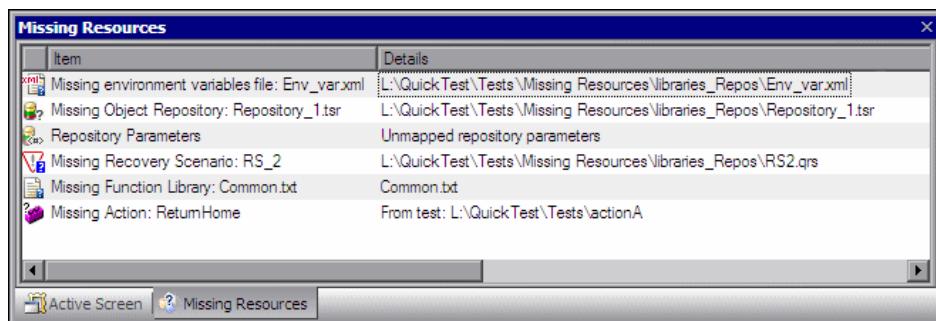
- 3 In the **Action** list, select the action you want to call. When you select an action, its type (Reusable Action) and description, if one exists, are displayed. This helps you identify the action you want to call. For more information on action descriptions, see "General Tab (Action Properties Dialog Box)" on page 559.
 - 4 Click **OK**. QuickTest updates the test with your changes and removes the action from the Missing Resources pane.
-

Note: If your test contains additional missing actions that can be located in the same test, QuickTest opens a message box asking you if you want to map these actions as well. Click **Yes** to map all relevant actions, or click **No** to map only the action you specified.

Reference

Missing Resources Pane User Interface

The Missing Resources pane provides a list of the resources that are referenced in your test but cannot be found.



To access	Select View > Missing Resources .
Important information	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.
Relevant tasks	"How to Handle a Missing Resource" on page 1367

The following sections describe:

- "Main User Interface Elements" on page 1372
- "Shortcut (Right-Click) User Interface Elements" on page 1373

Main User Interface Elements

The main user interface elements are described below (unlabeled elements are shown in angle brackets):

UI Elements	Description
<icon column>	<p>Displays icons indicating the type of missing resource. These icons are also sometimes displayed in QuickTest where a missing resource is used (for example, the Keyword View).</p> <p>The following icons may be displayed:</p> <ul style="list-style-type: none">  Missing action  Missing environment variable file  Missing function library  Missing object repository  Missing recovery scenario  Unmapped Repository parameters
Item	The type and name of the missing resource.
Details	Details about the missing resource. In most cases this is the path where QuickTest is looking for the resource.

Shortcut (Right-Click) User Interface Elements

The user interface elements described below may be available when you right-click an item in the Missing Resources pane:

UI Elements	Description
Locate	<p>Opens the relevant dialog box for the selected missing resource, enabling you to browse to the correct path for the resource.</p> <p>Tip: Double-clicking a missing resource item has the same effect as selecting the Locate option from the right-click menu.</p> <p>Relevant for missing actions, missing environment variables, missing function libraries, and missing recovery scenarios.</p>
Resolve	<p>Enables you to update details for the resource in order to solve the problem. Relevant only for missing object repositories and unmapped repository parameters.</p>
Remove	<p>Removes the association of this resource with your action or test.</p>
Filter	<p>Enables you to display only missing resources of a certain type or to display all missing resources.</p> <p>When the missing resource pane is filtered, an indication of the applied filter is shown at the bottom of the pane, for example:</p> <div data-bbox="623 1119 947 1154" style="border: 1px solid black; padding: 2px;">  = 'Missing Object Repository' </div> <p>Tip: You can cancel the filter and show all missing resources again by clicking the  icon on the left of the filter indication.</p>

Chapter 40 • Missing Resources Pane

41

Process Guidance Panes

This chapter includes:

Concepts

- Process Guidance Overview on page 1376

Tasks

- How to Manage Process Guidance on page 1377

Reference

- Process Guidance Management Dialog Box on page 1379
- Process Guidance Panes User Interface on page 1380

Concepts

Process Guidance Overview

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

A process is a collection of activities, or sub-processes. Each process walks you step-by-step through the activities that are required for that process. As you navigate through the activities for each process and perform the tasks described in each activity, you become acquainted with the way in which a particular process should be performed.

QuickTest provides a built-in package that comprises several processes. These processes provide introductory information and tips on how to perform the most common QuickTest tasks, such as planning and creating a test.

Your organization can also create its own custom processes to guide users through specific requirements and best practices relevant to your organization. For more information, see "Custom Process Guidance Packages" on page 1811.

How Processes are Stored

Processes are stored in process guidance packages. QuickTest provides a built-in package containing several processes. This package is listed by default in the Process Guidance Management dialog box.

Your organization may provide additional packages that include processes that are specific to your organization, your team, your role in your organization, and so on. For more information, see "Custom Process Guidance Packages" on page 1811.

You can select to include or exclude a package in the set of packages available in QuickTest.

When you select to include a package, QuickTest adds all of the processes in that package to the **Process Guidance List** on the Start Page (excluding processes for QuickTest add-ins that are not currently loaded). The processes that are available for the currently open document type and for the currently loaded QuickTest add-ins are also added to the **Process Guidance List** in the **Automation** menu, and can be opened after you refresh the list by closing and reopening the current document or by opening a new document of the same type.

If your organization has its own processes, you can add them to the **Process Guidance List** on the Start Page. You do this by adding the relevant package to the Process Guidance Management dialog box and selecting to show it.

Tasks

How to Manage Process Guidance

This task described the available operations for managing your process packages in the Process Guidance Management Dialog Box (described on page 1379).

This task includes the following:

- "Include and exclude available process guidance packages" on page 1377
- "Add new process guidance packages" on page 1378

Include and exclude available process guidance packages

In the Process Guidance Management Dialog Box (described on page 1379), do one of the following:

- To include a package, select the check box adjacent to the package whose processes you want to include.
- To exclude a package, clear the check box adjacent to the package whose processes you want to exclude.

When you click **Close**, QuickTest adds or removes the relevant processes in the **Process Guidance List**.

Add new process guidance packages

- 1 In the Process Guidance Management Dialog Box (described on page 1379), click **Add**. The Open dialog box opens.
- 2 Browse to the process guidance package file and click **Open**. The package is added to the list of available packages.
- 3 (Optional) Select the check box adjacent to the package to include its processes in the **Process Guidance List** on the Start page.

Open an existing process

You can open a process from the following locations:

- The Start Page: Select a process from the **Process Guidance List**.
- The **Automation** menu: Select **Automation > Process Guidance List**.
- The Process Guidance activities area in the Process Guidance Panes User Interface (described on page 1380).

Reference

Process Guidance Management Dialog Box

This dialog box enables you to manage the list of processes that are available in QuickTest.



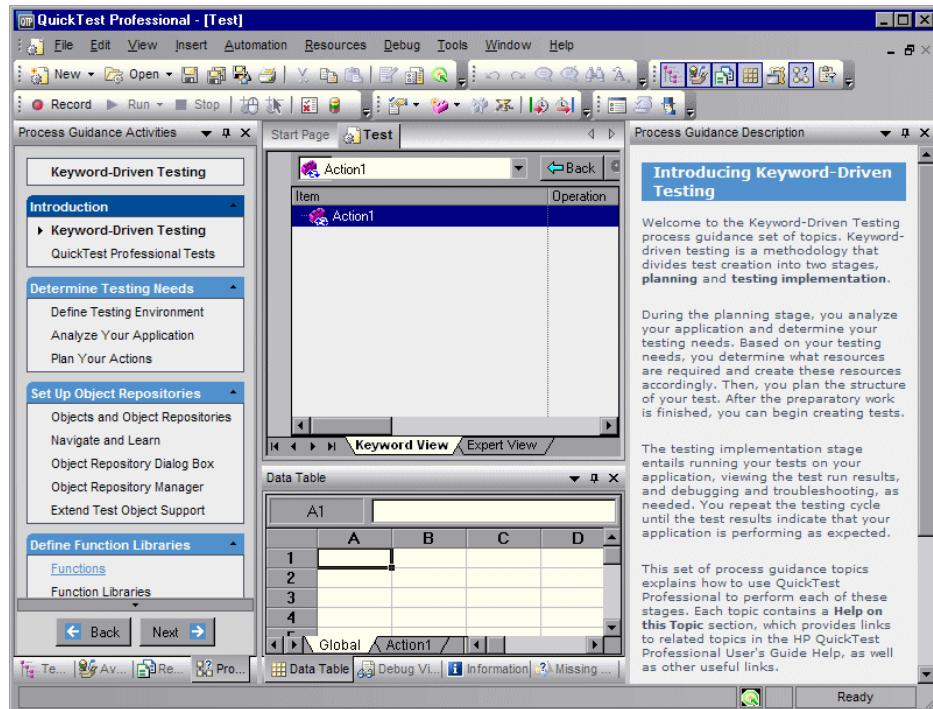
To access	Select File > Process Guidance Management .
Important information	<ul style="list-style-type: none">➤ You cannot remove the built-in QuickTest package.➤ You cannot include or exclude individual processes from within a package.
Related tasks	"How to Manage Process Guidance" on page 1377
See also	"Process Guidance Panes User Interface" on page 1380

User interface elements are described below:

UI Elements	Description
Add	Enables you to add processes specific to your organization to the Process Guidance List on the Start Page.
Remove	Enables you to remove processes from the Process Guidance List on the Start Page.

Process Guidance Panes User Interface

The Process Guidance panes enable you to view activities for a selected process and navigate between activities and processes.

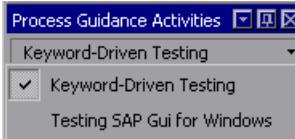


To access	In the QuickTest window, do one of the following: <ul style="list-style-type: none"> ▶ Select View > Process Guidance. ▶ Click the Process Guidance panes toggle button .
Important information	"Considerations for Working with Process Guidance" on page 1383
Relevant tasks	"How to Manage Process Guidance" on page 1377
See also	<ul style="list-style-type: none"> ▶ "Process Guidance Overview" on page 1376 ▶ "Process Guidance Management Dialog Box" on page 1379

The Process Guidance panes contain the following key elements:

Process Guidance Activities Pane

User interface elements are described below (unlabeled elements are shown in angle brackets):

UI Elements	Description
<process list bar>	The list of available processes. If the currently open testing document has more than one process available, you can navigate between these process by selecting the required process from the drop-down list in the process title. 
<activities list area>	The list of available activities for the selected process. Activities are often grouped, enabling you to navigate directly to the sub-process that interests you. Click an activity to open the relevant topic in the Process Guidance Description pane . Note: The example image in this section illustrates some of the groups and activities in the Keyword-Driven Testing process. For example, the Determine Testing Needs group contains three activities: Define Testing Environment, Analyze Your Application, and Plan Your Actions.

UI Elements	Description
Back / Next	<p>Enable you to navigate up and down between activities and to display the topic for the previous or next activity in the Process Guidance Description pane.</p> <p>Tip: If the Process Guidance Pane is not large enough to display all of the activities, you can position the cursor over the Up or Down arrows to scroll through the list of activities. (The up arrow is located at the top of the pane, directly below the Process Guidance Activities title bar; the down arrow is located at the bottom of the pane, directly above the Back and Next buttons.)</p>

Process Guidance Description Pane

User interface elements are described below (unlabeled elements are shown in angle brackets):

UI Elements	Description
<description area>	<p>The topic description for the selected activity. Each description introduces you to a specific activity and provides links to locations in which you can find more information about how to perform that activity.</p> <p>Additionally, many of the descriptions include interactive links that open dialog boxes or other relevant features, enabling you to directly access the features that are being described.</p>



Considerations for Working with Process Guidance

- If the **Process Guidance List** is empty, open the Process Guidance Management dialog box and select at least one process. For more information, see "Process Guidance Management Dialog Box" on page 1379.
- If you want to open a process that is not relevant for the current testing document or loaded QuickTest add-in, you need to open the process from **Process Guidance List** in the Start Page.
- The **Process Guidance List** on the Start Page displays all available processes. Some processes may be available only under certain conditions. For example, the Business Components process guidance is available only if you are connected to a Quality Center project that supports business process testing. Additionally, some processes may be visible only if you have a specific add-in loaded. For example, the **Testing SAP GUI for Windows** built-in process is visible only if the SAP add-in is loaded.
- When you select a QuickTest process from the list, the relevant document type opens. For example, if a test document is open and you select the **Application Areas** process, a new application area opens, enabling you to navigate through the application area as you navigate through the selected process (provided that you are connected to a Quality Center project with business process testing support).

42

Resources Pane

This chapter includes:

Concepts

- Resources Pane Overview on page 1386

Tasks

- How to Modify Associations Between Resources and Your Test or Action on page 1386

Resources

- Resources Pane User Interface on page 1391

Concepts

Resources Pane Overview

Tests and actions are associated with resources, such as function libraries, recovery scenarios, and object repositories.

QuickTest enables you to use the Resources pane to associate, remove, open, change the priority of, and otherwise manage the associations of these resource files in your test.

Tasks

How to Modify Associations Between Resources and Your Test or Action

The following sections describe how to use the Resources pane to view and manage the resource file associations in your test or action:

- ▶ "Resources - General Options" on page 1387
- ▶ "Function Libraries" on page 1388
- ▶ "Recovery Scenarios" on page 1388
- ▶ "Object Repositories" on page 1389
- ▶ "Actions" on page 1390

Resources - General Options

To view the resources associated with your test or action:

In the Resource pane, expand the relevant node of the Resources tree.

To open a resource file from the Resources pane:

Do one of the following:

- Double-click the resource node.
- Right-click the resource node and select the relevant command:
 - **Open Function Library.** Opens the selected function library in the QuickTest Function Library window, enabling you to view and modify it (if in read-write mode). For more information, see "Working in the Expert View and Function Library Windows" on page 929.
 - **Recovery Scenario Properties.** Opens the Recovery Scenario Properties dialog box, enabling you to view properties for the recovery scenario in read-only, such as trigger events and recovery operations. For more information, see "Recovery Scenario Properties Dialog Box" on page 1537.
 - **Open Repository.** Opens the Object Repository Window-Local Object Repository for local object repositories and the Object Repository Manager for shared object repositories. For more information, see "Object Repository Window" on page 237 and "Object Repository Manager Main Window" on page 266.

To remove the association of a resource file with the test or action:

Right-click the resource node and select the relevant command:

- **Remove Function Library from List.** Removes the association of the selected function library with your test.
- **Remove Recovery Scenario from List.** Removes the selected recovery scenario from your test.
- **Remove Repository from List.** Removes the selected object repository from the action.

Removing a resource node severs the association between the resource and the test or action. The resource node is deleted from the Resources pane and any other location that indicates associations, such as the Test Settings dialog box (for function libraries and recovery scenarios) and the Associate Repositories dialog box (for shared object repositories).

To change the priority of a resource file:

Right-click the resource node and select **Move Up** or **Move Down**.

Function Libraries

To associate a function library with a test:

- 1 (Optional) To associate a function library stored in a Quality Center project, connect QuickTest to the relevant Quality Center project. For more information, see "How to Work with Tests in Quality Center" on page 1618.
- 2 In the Resources pane, right-click **Associated Function Libraries** and select **Associate Function Library**. The Open Function Library dialog box opens.
- 3 Browse to and select a function library. For details, see "Open <Resource> Dialog Box" on page 406. The function library is associated with the test.

Recovery Scenarios

To associate a recovery scenario with a test:

- 1 (Optional) To associate a recovery scenario stored in a Quality Center project, connect QuickTest to the relevant Quality Center project. For more information, see "How to Work with Tests in Quality Center" on page 1618.
- 2 In the Resources pane, right-click **Associated Recovery Scenarios** and select **Associate Recovery Scenario**. The Add Recovery Scenario dialog box opens.

- 3 Browse to and select a recovery scenario file. Then select the specific recovery scenarios you want to associate. For more information, see "Add Recovery Scenario Dialog Box" on page 1532.

The recovery scenarios are associated with the test. You can view the associated recovery scenarios in the Resources pane and in the Recovery pane of the Test Settings dialog box (**File > Settings > Recovery** node).

To disable or enable a recovery scenario:

Right-click the recovery scenario node and select **Disable Recovery Scenario / Enable Recovery Scenario**. This toggle command enables or disables the selected recovery scenario depending on whether it was previously enabled or disabled.

Object Repositories

To associate a shared object repository with an action stored in the current test (not relevant for external actions):

- 1 (Optional) To associate an object repository stored in a Quality Center project, connect QuickTest to the relevant Quality Center project. For more information, see "How to Work with Tests in Quality Center" on page 1618.
- 2 In the Resources pane, right-click **Associated Repositories per Action** and select **Associate Repository with Action**. The Open Shared Object Repository dialog box opens.
- 3 Browse to and select a shared object repository. For details, see "Open <Resource> Dialog Box" on page 406. The shared object repository is associated with the action.

Actions

To view or modify an action's properties:

Right-click the action node and select **Action Properties**. The Action Properties dialog box opens, enabling you to define options for the stored action. You can:

- Modify an action name.
- Add or modify an action description.
- Set an action as reusable or non-reusable.
- Set the data table definitions (for external actions).

For more information, see "How to Use Actions in Your Test" on page 542.

To delete an action from a test:

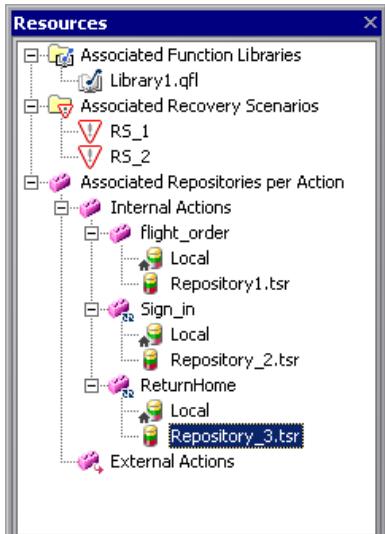
Right-click the action node and select **Delete Action**. The action is removed from the test. For information on the effects of removing various types of actions from tests, see "Remove a call to an action or delete an action" on page 544.

Reference

Resources Pane User Interface

This pane enables you to manage the resources associated with your test in one pane.

You can associate, remove, open, change the priority, and otherwise manage the function libraries, recovery scenarios, and object repositories in your test.



To Access	Do one of the following: <ul style="list-style-type: none"> ► Select View > Resources. ► Click the Resources Pane toolbar button .
Relevant tasks	"How to Modify Associations Between Resources and Your Test or Action" on page 1386
See also	<ul style="list-style-type: none"> ► "Resources Pane Overview" on page 1386 ► "Missing Resources Pane User Interface" on page 1371

User interface elements are described below:

Associated Function Libraries Node

Lists all of the function libraries currently associated with your test.

Node	Context Menu Option
Associated Function Libraries	Associate Function Library. Opens the Open Function Library dialog box, enabling you to associate a function library with your test.
<Function library node>	<ul style="list-style-type: none"> ➤ Open Function Library. Opens the selected function library in the QuickTest Function Library window. ➤ Remove Function Library from List. Removes the selected function library from your test. ➤ Move Up or Move Down. Moves the selected function library up or down the priority list of associated function libraries.

See also: "User-Defined Functions and Function Libraries" on page 1011

Associated Recovery Scenarios Nodes

Lists all of the recovery scenarios currently associated with your test.

Node	Context Menu Option
Associated Recovery Scenarios	Associate Recovery Scenario. Opens the Add Recovery Scenario dialog box. For information, see "Add Recovery Scenario Dialog Box" on page 1532.
<Recovery scenario node>	<ul style="list-style-type: none"> ➤ Recovery Scenario Properties. Opens the Recovery Scenario Properties dialog box. For more information, see "Recovery Scenario Properties Dialog Box" on page 1537. ➤ Remove Recovery Scenario from List. Removes the selected recovery scenario from your test. ➤ Move Up or Move Down. Moves the selected recovery scenario up or down the priority list of associated recovery scenarios. ➤ Disable Recovery Scenario / Enable Recovery Scenario. Disables or enables the selected recovery scenario.

See also: "Recovery Scenarios" on page 1523

Associated Repositories per Action Nodes

Lists all of the object repositories currently associated with all of the actions in your test.

The nodes in this part of the Resources pane are organized according to the following hierarchy:

Associated Repositories per Action node

Internal Actions and External Actions nodes

individual actions (includes all of the actions stored with your test, even when they are not called by your test)

local and shared object repositories associated with the individual action (if any)

Node	Context Menu Option
Action	<ul style="list-style-type: none"> ➤ Associate Repository with Action. Opens the Open Shared Object Repository dialog box, enabling you to associate an object repository with the selected action. This option is disabled for external actions. ➤ Action Properties. Opens the Action Properties dialog box, enabling you to define options for the stored action. For more information, see "Action Properties Dialog Box" on page 557. ➤ Delete Action. Removes the action from the test.
<Repository node>	<ul style="list-style-type: none"> ➤ Open Repository. Opens the Object Repository Window-Local Object Repository for local object repositories and the Object Repository Manager for shared object repositories. ➤ Remove Repository from List. Removes the selected object repository from the action. ➤ Move Up or Move Down. Modifies the priority of the selected object repository when you move it up or down in the list of associated repositories.

See also: "Managing Test Objects in Object Repositories" on page 159

43

Test Flow Pane

This chapter includes:

Concepts

- [Test Flow Pane Overview on page 1396](#)

Tasks

- [How to Manage Actions in the Test Flow Pane on page 1397](#)

Reference

- [Test Flow Pane User Interface on page 1401](#)

Concepts

Test Flow Pane Overview

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 do the following:

- ▶ Display test, action, and action call properties
- ▶ Manage actions and change their order in the test
- ▶ Work with the object repository
- ▶ Run specific actions

For details, see "How to Manage Actions in the Test Flow Pane" on page 1397.

Note: Nested actions that are commented out are not displayed in the Test Flow pane.

Tasks

How to Manage Actions in the Test Flow Pane

This task describes how to perform various operations in the Test Flow pane.

This task includes the following:

- ▶ "Display an action in the Keyword View and Expert View" on page 1397
- ▶ "Display the test properties" on page 1398
- ▶ "Display the action properties" on page 1398
- ▶ "Display the action call properties" on page 1398
- ▶ "View or hide the sub-nodes of an action" on page 1398
- ▶ "View or hide the sub-nodes in the test" on page 1398
- ▶ "Change the run order of actions" on page 1399
- ▶ "Copy or delete actions" on page 1399
- ▶ "Work with the object repository" on page 1400
- ▶ "Run your test from a specific action" on page 1400
- ▶ "Debug your test from a specific action" on page 1400

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. The following view are available:

- ▶ The Keyword View displays the steps of your test in a modular, table format. For details, see Chapter 13, "Keyword View."
- ▶ The Expert View displays the script for the selected action. For details, see Chapter 27, "Working in the Expert View and Function Library Windows."

You can view and edit the individual steps of an action stored in this test, and view the steps for each selected external action.

Display the test properties

Right-click the **Test** node in the tree and then select **Settings** to open the Test Settings dialog box. Details of the test and its path are displayed. For details, see "Individual Test Settings" on page 1455.

Display the action properties

Right-click an action in the tree and then select **Action Properties** to open the Action Properties dialog box. The name of the action and its path are displayed. For details, see "Action Properties Dialog Box" on page 557.

Display the action call properties

Right-click an action in the tree and then select **Action Call Properties** to open the Action Call Properties dialog box. For details, see "Action Call Properties Dialog Box" on page 550.

View or hide the sub-nodes of an action

Do one of the following:

- Right-click an action in the tree and then select **Expand Sub Tree** or **Collapse Sub Tree**
- Select a sub-node and press + or * on the keyboard to expand the node and - to collapse the node.

View or hide the sub-nodes in the test

Do one of the following:

- Right-click the **Test** node in the tree and select **Expand All** or **Collapse All**.
- Select the **Test** node and press + or * on the keyboard to expand all the nodes in the test, and - to collapse the nodes.

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**.
- Press CTRL+UP arrow or CTRL+DOWN arrow.
- 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.

Note: You can only drag top-level actions. Selecting the parent action automatically includes all of its child actions. You cannot drag a child action, and you cannot drag a parent action together with only some of its child actions.

For details, see "Keyword View User Interface" on page 504.

Copy or delete actions

Right-click an action in the tree and then select one of the following:

- Select **Copy** to open the Select Action dialog box and create a copy of the action in your test. For details, see "Select Action Dialog Box" on page 582.
- Select **Delete** to remove the action from your test. For details, see "How to Use Actions in Your Test" on page 542.

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 details, see Chapter 6, "Shared Object Repositories."

Run your test from a specific action

Right-click an action in the tree and then select one of the following:

- **Run from Action** to start a run session from the beginning of the selected action.
- **Run to Action** to run the test until the beginning of the selected action and then pause the run session.

Debug your test from a specific action

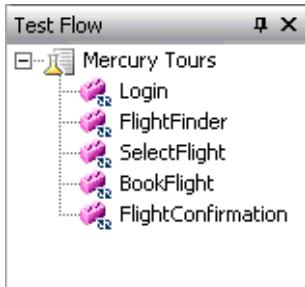
Right-click an action in the tree and then select **Debug from Action** to begin (and pause) a debug session at the beginning of the selected action.

Reference

Test Flow Pane User Interface

This pane enables you to:

- View all the calls to actions in the current test and the order in which they are run
- Manage the order of actions
- Run your test from a selected action or to a selected action
- Display actions in the Keyword View or the Expert View



To access	<p>Do one of the following:</p> <ul style="list-style-type: none"> ➤ Select View > Test Flow. ➤ Click the Test Flow Pane toolbar button .
Important information	<ul style="list-style-type: none"> ➤ The Test Flow pane is displayed by default when you start QuickTest Professional. ➤ For details on working with actions in your test, see Chapter 14, "Actions".
Relevant tasks	"How to Manage Actions in the Test Flow Pane" on page 1397
See also	"Test Flow Pane Overview" on page 1396

This pane includes the following icons and context menu options:

- "Item Type Icons" on page 1402
- "Context Menu Options — Test Node" on page 1403
- "Context Menu Options — Action Node" on page 1403
- "Context Menu Options — Service Test Node" on page 1404

Item Type Icons

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

Context Menu Options — Test Node

Option	Description
Expand All	Opens the sub-nodes in the test.
Collapse All	Hides the sub-nodes in the test.
Settings	Opens the "Properties Pane (Test Settings Dialog Box)" on page 1466, which enables you to view the test settings.

Context Menu Options — Action Node

Option	Description
Expand Sub Tree (action only)	Opens the sub-nodes in the action.
Collapse Sub Tree (action only)	Hides the sub-nodes in the action.
Action Properties	Opens the Action Properties Dialog Box (described on page 557), which enables you to view details of the selected action.
Action Call Properties	Opens the Action Call Properties Dialog Box (described on page 550), enables you to view details of the selected action call.
Object Repository	Opens the Object Repository window, which displays a tree containing all objects in the current test.
Copy	Opens the Select Action Dialog Box (described on page 582), which enables you to copy the selected action.
Delete	Removes the selected action from your test. For details, see "How to Use Actions in Your Test" on page 542.
Run from Action	Starts a run session from the beginning of the selected action.
Debug from Action	Start (and pause) a debug session at the beginning of the selected action.
Run to Action	Runs the test until the beginning of the selected action and then pause the run session.

Option	Description
Move Up	Moves a top-level action (a direct child of the test) up the Test Flow Pane tree.
Move Down	Moves a top-level action (a direct child of the test) down the Test Flow Pane tree.

Context Menu Options — Service Test Node

The Service Test node is available only if your test contains a call to a Service Test test.

Option	Description
Edit Call to Service Test Test	Opens the Call to Service Test Test dialog box, enabling you to view and modify your Service Test test parameters. For details, see "Call to Service Test Test Dialog Box" on page 1741. Note: To replace the current call with a different Service Test test, you must first delete the current call and then insert a new one.
Delete	Removes the selected Service Test call from your test.

44

To Do Pane

This chapter includes:

Concepts

- To Do Pane Overview on page 1406

Tasks

- How to Manage Tasks and TODO Comments on page 1407

Reference

- Task Editor Dialog Box on page 1408
- To Do Pane User Interface on page 1410

Concepts

To Do Pane Overview

QuickTest enables you to create and manage tasks and TODO comments about issues that need to be handled in your tests and function libraries. For example, you can provide instructions to someone else during a handover, or remind yourself to do something.

If needed, you can export your tasks and TODO comments to Microsoft Excel or an XML file.

- **Tasks** are test-related reminders that are linked to the currently open test. You create and manage tasks using the To Do pane and the Task Editor dialog box. For more information, see "Tasks Tab (To Do Pane)" on page 1410.
- **TODO comments** are reminders that are inserted as comment steps adjacent to the relevant steps in an action or function library. You can access TODO comments from the To Do pane or directly from the testing document. For more information, see "Comments Tab (To Do Pane)" on page 1413.

Tasks

How to Manage Tasks and TODO Comments

This task describes how to manage your tasks and TODO comments.

Manage tasks

- Add a new task in the Tasks Tab (To Do Pane) (described on page 1410), by opening the Task Editor Dialog Box (described on page 1408) and clicking the **Add Task** button .
- Use the Task Editor dialog box to edit existing tasks, as needed. Tasks are saved with the test.
- To sort by a specific column, click on the column header.

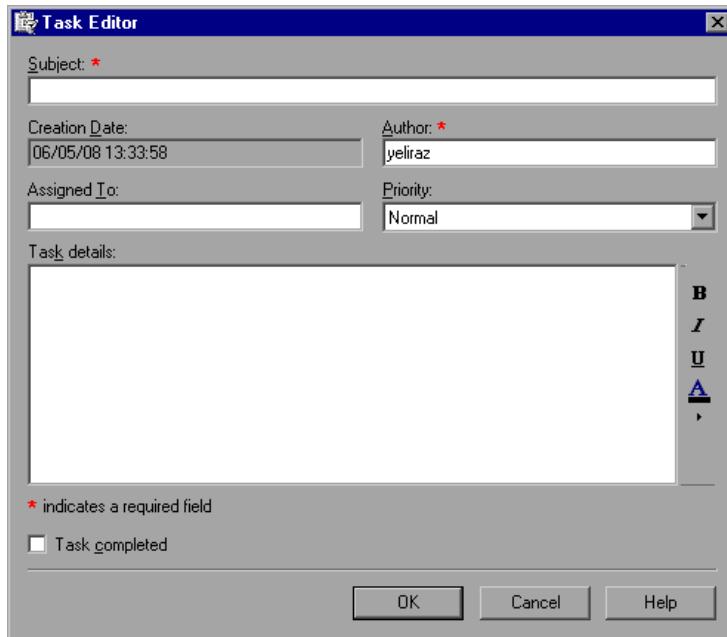
Manage TODO comments

- Add a new TODO comment by inserting a Comment step. For information on Comment steps, see "Comments in the Keyword View" on page 486.
TODO comments are inserted as comment steps adjacent to the relevant steps in an action or function library.
- In the Comments Tab (To Do Pane) (described on page 1413), view any comment step that begins with any of the following permutations of the words **to do:** To Do, todo, to-do, or TODO. To sort by a specific column, click on the column header.

Reference

Task Editor Dialog Box

This dialog box enables you to add a task to the To Do pane, edit an existing task, or to mark a task as complete.



To access	In the Tasks tab of the To Do pane, do one of the following: <ul style="list-style-type: none"> ➤ Click the Add Task button  or Press INSERT. ➤ Select an existing task and click the Edit Task button  or press ENTER.
Important Information	Fields marked with a red asterisk (*) are mandatory.
See also	<ul style="list-style-type: none"> ➤ "To Do Pane Overview" on page 1406 ➤ "To Do Pane User Interface" on page 1410

User interface elements are described below:

Option	Description
Subject	A descriptive topic name for the task (Mandatory field). For naming conventions, see "Naming Conventions" on page 1779.
Creation Date	The date and time that the Task Editor was opened to create the current task. (Read-only)
Author	The automatically generated name of the user who created the task. You can modify the Author field when creating a task but not when modifying it. (Mandatory field upon creation)
Assigned To	The name of the user who is responsible for handling the task.
Priority	<p>The importance of the task. Possible values:</p> <ul style="list-style-type: none"> ➤ High Priority ➤ Normal Priority ➤ Low Priority
Completed	<p>Indicates whether the task was fully implemented. Possible values:</p> <ul style="list-style-type: none"> <input checked="" type="checkbox"/> Task completed <input type="checkbox"/> Task not completed
Task Details	A textual description of the task. You can modify the font style (bold, italic, and underline) and color to highlight various parts of the task details.

To Do Pane User Interface

This pane enables you to view and manage your test-related tasks and TODO comments.

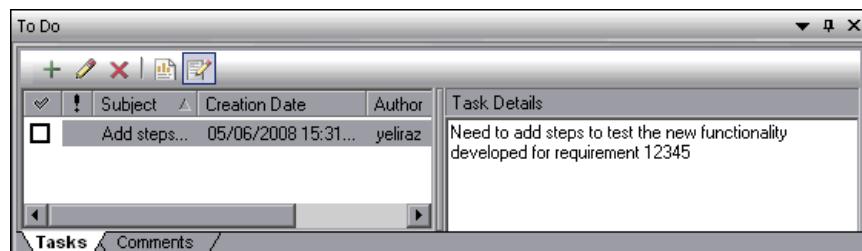
To access	<p>In the main QuickTest window, do one of the following:</p> <ul style="list-style-type: none"> ▶ Select View > To Do. ▶ Click the To Do Pane toolbar button . <p>Note: The To Do pane opens automatically when you open a test that contains tasks.</p>
See also	<ul style="list-style-type: none"> ▶ "To Do Pane Overview" on page 1406 ▶ "Task Editor Dialog Box" on page 1408

The To Do pane contains the following tabs:

- ▶ **Tasks tab.** Enables you to create and manage your test--related tasks. For more information, see "Tasks Tab (To Do Pane)" on page 1410.
- ▶ **Comments tab.** Enables you to view and access TODO comments in an action or currently open function library. For more information, see "Comments Tab (To Do Pane)" on page 1413.

Tasks Tab (To Do Pane)

This tab displays all of the tasks defined for the currently open test. You define tasks using the Task Editor dialog box.



To access	Select View > To Do >Tasks tab .
See also	<ul style="list-style-type: none"> ➤ "To Do Pane Overview" on page 1406 ➤ "Comments Tab (To Do Pane)" on page 1413 ➤ "Task Editor Dialog Box" on page 1408

The Tasks tab includes the following key elements:

- "Toolbar Buttons" on page 1411
- "Columns" on page 1412
- "Context Menu Options" on page 1412

Toolbar Buttons

	Toolbar Option	Shortcut Key	Description
	Add Task	INSERT	Opens the Task Editor dialog box (described on page 1408), enabling you to add a new task to the Tasks tab in the To Do pane.
	Edit Task	ENTER	Opens the Task Editor dialog box (described on page 1408), enabling you to modify the selected task or mark it as complete.
	Delete Task	DELETE	Removes the selected task from the To Do pane.
	Export TODO List	N/A	<p>Saves the tasks to an external file, such as a text file.</p> <p>You can save the tasks in the list in any of the following formats:</p> <ul style="list-style-type: none"> ➤ XML (Extensible Markup Language) ➤ XLS (Microsoft Excel file) ➤ CSV (Comma-Separated Values file)
	Show/Hide Task Details	N/A	Opens or closes the Task Details pane on the right side of the To Do pane, displaying more/less information about a selected task.

Columns

Column	Description
Completed	<p>Indicates whether the task was fully implemented. When you mark a task as complete, a strike-through format is applied to the task in the pane, indicating that the task is completed.</p> <p>Possible values:</p> <ul style="list-style-type: none"> <input checked="" type="checkbox"/> Task completed <input type="checkbox"/> Task not completed <p>Tip: You can also mark a task as complete by selecting the Task completed check box in the Task Editor dialog box.</p>
Priority	<p>Indicates the importance of the task.</p> <p>Possible values:</p> <ul style="list-style-type: none"> High Priority Normal Priority Low Priority
Subject	Indicates the topic of the task.
Creation Date	Indicates the date and time that the Task Editor was opened to create the current task.
Author	Indicates the name of the user who created the task.
Assigned To	Indicates the name of the user who is responsible for handling the task.
Task Details	When enabled, displays a textual description of the task.

Context Menu Options

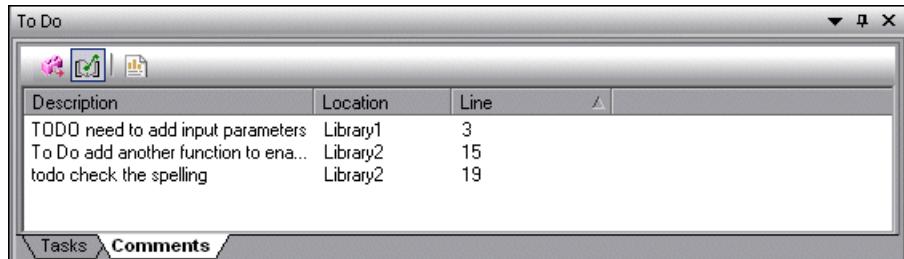
Context Menu Option	Description
Sort By	Enables you to choose a column by which to sort the tasks in the tab.

Context Menu Option	Description
Duplicate	Creates a copy of the selected task and inserts it in the Tasks tab. This is useful if you want to create a new task that is similar to an existing one.
Delete Task	Permanently removes the task from the Tasks tab in the To Do pane.

Comments Tab (To Do Pane)

This tab can display any comment step that begins with any of the following permutations of the words **to do**: To Do, todo, to-do, or TODO (not case-sensitive).

Example: To Do need to ask Sarah to add design steps



To access	Select View > To Do > Comments tab
Important Information	<ul style="list-style-type: none"> ➤ By default, the Comments tab displays all TODO comment steps in the test's local actions. ➤ The text displayed in the Comments tab is limited to 260 characters. If the text exceeds this limit, and you want to view the entire comment, you can jump to the comment in the testing document by double-clicking the comment line in the Comments tab. ➤ You can show TODO comments in: <ul style="list-style-type: none"> ➤ The current test's local actions. ➤ Any external actions associated with the test. ➤ Any open function library.

See also	<ul style="list-style-type: none"> ► "To Do Pane Overview" on page 1406 ► "Tasks Tab (To Do Pane)" on page 1410 ► "Task Editor Dialog Box" on page 1408
-----------------	--

The Comments tab contains the following key elements:

Toolbar Buttons

Toolbar Option	Description
	Comments in External Actions Toggle button that enables you to display or hide any TODO comments from external actions.
	Comments in Open Function Libraries Toggle button that enables you to display or hide any TODO comments from currently open function libraries (in addition to the TODO comments from the local actions).
	Export TODO List Saves the TODO comments to an external file, such as a text file. You can save the list of TODO comments in any of the following formats: <ul style="list-style-type: none"> ► XML (Extensible Markup Language) ► XLS (Microsoft Excel file) ► CSV (Comma-Separated Values file)

Columns

Column	Description
Description	Displays the text of the TODO comment.
Location	Specifies the name of the action or the path of the function library containing the TODO comment.
Line	Specifies the line number of the TODO comment in the action or function library.

Context Menu Options

Context Menu Option	Description
Sort By	Enables you to choose a column by which to sort the TODO comments in the tab.
Go To Comment Line	Moves the cursor to the comment line in the action or function library. Tip: You can also double-click a comment in the Comments tab or press ENTER to move the cursor to the comment line in the action or function library.

Part IX

Configuring QuickTest Settings

45

Global Testing Options

This chapter includes:

Concepts

- ▶ Global Testing Options Overview on page 1420

Reference

- ▶ Options Dialog Box on page 1420
- ▶ General Pane (Options Dialog Box) on page 1422
- ▶ Folders Pane (Options Dialog Box) on page 1431
- ▶ Active Screen Pane (Options Dialog Box) on page 1434
- ▶ Run Pane (Options Dialog Box) on page 1447

Concepts

Global Testing Options Overview

Global testing options affect both how you work with tests and the general appearance of QuickTest. For example, you can choose not to display the Start Page when QuickTest starts, or you can set the timing-related settings used by QuickTest when running a test. The values you set remain in effect for all tests and for subsequent testing sessions. You can set global testing options using the Options dialog box (described on page 1420) or by inserting statements in the Expert View.

You can also set testing options that affect only the test currently open in QuickTest. For more information, see Chapter 46, "Individual Test Settings."

Reference

Options Dialog Box

This dialog box enables you to modify your global testing options. It contains key nodes, which enable you to navigate to the options you want to view or modify. The values you set remain in effect for all subsequent QuickTest sessions.

To access	Do one of the following: ► Select Tools > Options . ► Click the Options toolbar button  .
-----------	--

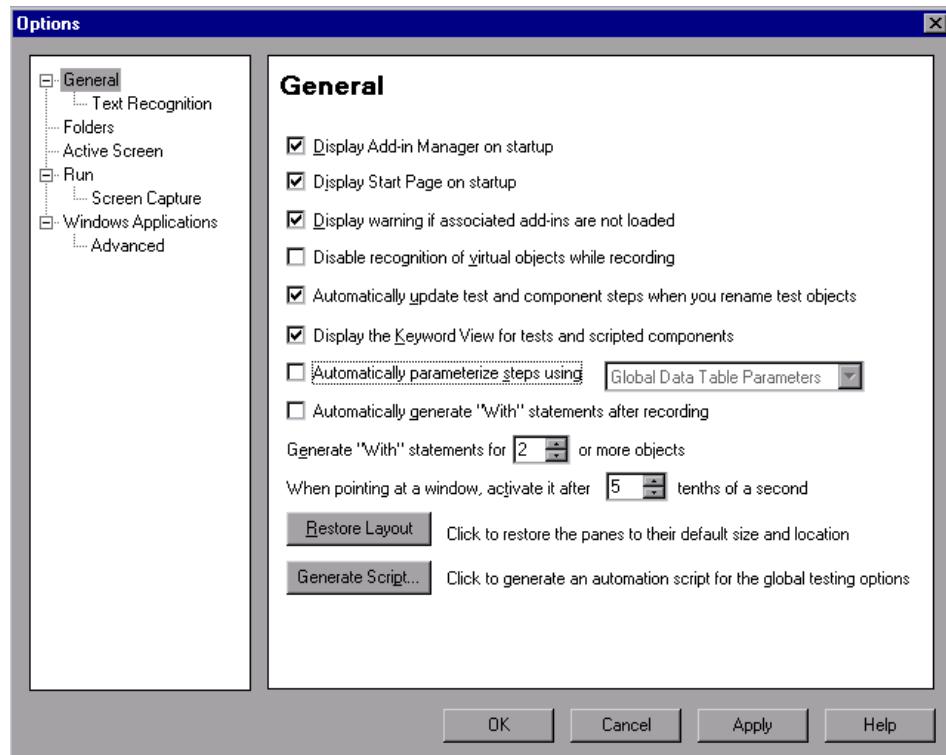
The Options dialog box contains the following key nodes:

Node	Options
General	<p>Options for general test settings. For more information, see "General Pane (Options Dialog Box)" on page 1422.</p> <p>The General node also contains the Text Recognition sub-node. For more information, see "Text Recognition Pane (Options Dialog Box)" on page 1426.</p>
Folders	<p>Options for entering the folders (search paths) in which QuickTest searches for tests, actions, or files that are specified as relative paths in dialog boxes and statements. For more information, see "Folders Pane (Options Dialog Box)" on page 1431.</p> <p>Note: If you are working with the Resources and Dependencies model with Quality Center 10.00 or HP ALM, specify an absolute Quality Center path. For more information, see "Relative Paths and Quality Center" on page 1656.</p>
Active Screen	<p>Options for configuring which information QuickTest saves and displays in the Active Screen while recording. For more information, see "Active Screen Pane (Options Dialog Box)" on page 1434.</p>
Run	<p>Options for running tests. For more information, see "Run Pane (Options Dialog Box)" on page 1447.</p> <p>The Run node also contains the Screen Capture node. For more information, see "Screen Capture Pane (Options Dialog Box)" on page 1450.</p>
Windows Applications	<p>Options for configuring how QuickTest tests interface with Windows applications. The Windows Applications node also includes the Advanced node. For more information, see the section on testing Windows-based applications in the <i>HP QuickTest Professional Add-ins Guide</i>.</p>

The navigation tree may contain additional nodes, depending on the add-ins that are currently loaded. For more information, see the relevant section in the *HP QuickTest Professional Add-ins Guide*.

General Pane (Options Dialog Box)

The options set in this pane affect the general appearance of QuickTest and other general testing options.



To access	Select Tools > Options > General node.
See also	"Text Recognition Pane (Options Dialog Box)" on page 1426

User interface elements are described below:

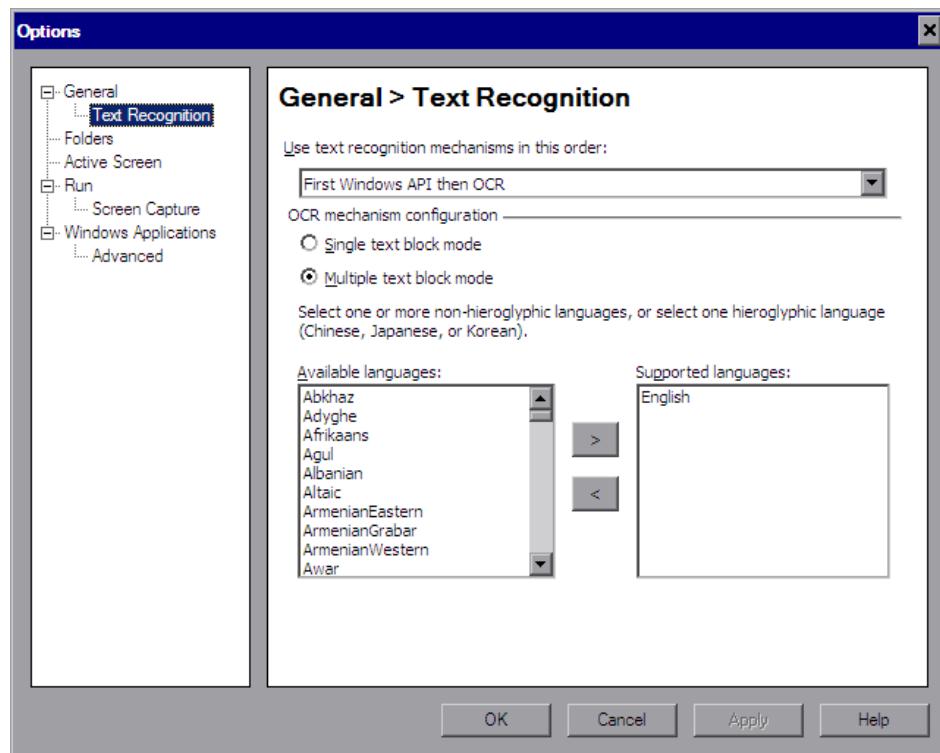
UI Element	Description
Display Add-in Manager on startup	Determines whether the Add-in Manager is displayed when starting QuickTest. For information on working with the Add-in Manager, see the section on loading QuickTest add-ins in the <i>HP QuickTest Professional Add-ins Guide</i> .
Display Start Page on startup	Determines whether the Start Page is displayed when starting QuickTest.
Display warning if associated add-ins are not loaded	Determines whether a warning message is displayed if you open a test that requires add-ins that are not currently loaded.
Disable recognition of virtual objects while recording	Determines whether the defined virtual objects stored in the Virtual Object Manager are recognized while recording. For more information, see Chapter , "Virtual Objects."
Automatically update test and component steps when you rename test objects	Determines whether to automatically update test and component steps when you rename test objects in the local or shared object repository. For more information, see "Renaming Test Objects" on page 170. Note: Applicable only if there are no syntax errors in the test when you rename the test object.
Display the Keyword View for tests and scripted components	Displays the Keyword View in addition to the Expert View when this check box is selected. Clearing this check box hides the Keyword View, which may be useful if you work only in the Expert View. Note: Changes to this option affect testing documents that are opened after this option is applied. Changes do not affect the currently open testing document.

UI Element	Description
Automatically parameterize steps using	<p>Instructs QuickTest to automatically parameterize the test object method arguments in any action in which you recorded one or more steps during a recording session.</p> <p>You can select to parameterize the steps with Global Data Table Parameters or Test Parameters.</p> <p>For details, see "Automatically Parameterizing Steps" on page 738.</p>
Automatically generate "With" statements after recording	<p>Instructs QuickTest to automatically generate With statements when you record. For more information, see "How to Generate With Statements for Your Test" on page 906.</p>
Generate "With" statements for __ or more objects	<p>Indicates the minimum number of identical, consecutive objects for which you want to apply the With statement. This setting is used when QuickTest automatically generates With statements after recording and when you select to generate With statements for an existing action.</p> <p>Default = 2</p> <p>For more information, see "How to Generate With Statements for Your Test" on page 906.</p>
When pointing at a window, activate it after __ tenths of a second	<p>Specifies the time (in tenths of a second) that QuickTest waits before it sets the focus on an application window when using the pointing hand to point to an object in the application (for Object Spy, checkpoints, Step Generator, Recovery Scenario Wizard, and so forth).</p> <p>Default = 5</p>

UI Element	Description
Restore Layout	<p>Restores the layout of the QuickTest window so that it displays the panes and toolbars in their default sizes and positions.</p> <p>Tip: For details on restoring default items in customized menus and toolbars, see "Toolbars Tab (Customize Dialog Box)" on page 1300.</p> <p>Note: QuickTest recalls your most recent window layout for each of its operating modes: view/edit, record, and run. For more information, see "QuickTest Window Layout Customization in Different Modes" on page 1280.</p>
Generate Script	<p>Generates a QuickTest automation script containing the current global testing options.</p> <p>When you click the Generate Script button, a Save As dialog box opens, enabling you to specify the name and file system location to store the generated file.</p> <p>You can use some or all of the script lines from this generated script in an automation script. This can be useful, for example, if you want to create an initialization script that will open QuickTest with a pre-defined set of options applied.</p> <p>For more information, see "QuickTest Automation Scripts" on page 1589 and the <i>QuickTest Professional Automation Object Model Reference</i> (Help > QuickTest Professional Help > HP QuickTest Professional Advanced References > HP QuickTest Professional Automation Object Model).</p>

Text Recognition Pane (Options Dialog Box)

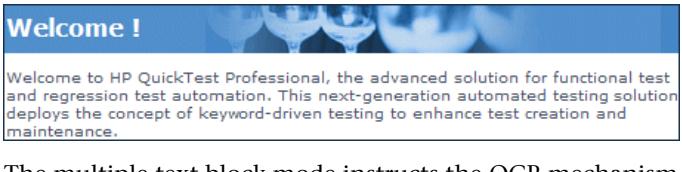
This pane enables you to configure how QuickTest identifies text in your application. You can use this pane to modify the default text capture mechanism, OCR (optical character recognition) mechanism mode, and the language dictionaries the OCR mechanism uses to identify text.



To access	Select Tools > Options > 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.
Related tasks	<ul style="list-style-type: none">➤ "How to Configure Text Recognition Settings" on page 852➤ "How to Create or Modify a Text or Text Area Checkpoint Step" on page 657➤ "How to Create or Modify a Text or Text Area Output Value Step" on page 794
See also	<ul style="list-style-type: none">➤ "Text Recognition for Windows-Based Objects" on page 847➤ "Guidelines for Text Recognition" on page 854➤ "Checking Text in an Image - Use-Case Scenario" on page 848➤ "Text Checkpoints" on page 655

User interface elements are described below:

UI Element	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; the only option available when working with Windows 7 and Windows Server 2008 R2.)</p> <p>For more information, see "Notes and Guidelines" on page 1430.</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="491 1206 659 1240" style="background-color: #0072BD; color: white; padding: 5px; text-align: center;"> Welcome ! </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>

UI Element	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>

Notes and Guidelines

- The Windows API text recognition mechanism is provided as is. If it does not work as you expect, change the option to use OCR.

However, although OCR accuracy is usually very high, keep in mind that as with all OCR technologies, the OCR mechanism does not guarantee 100% accuracy.

For more information, see "Guidelines for Text Recognition" on page 854

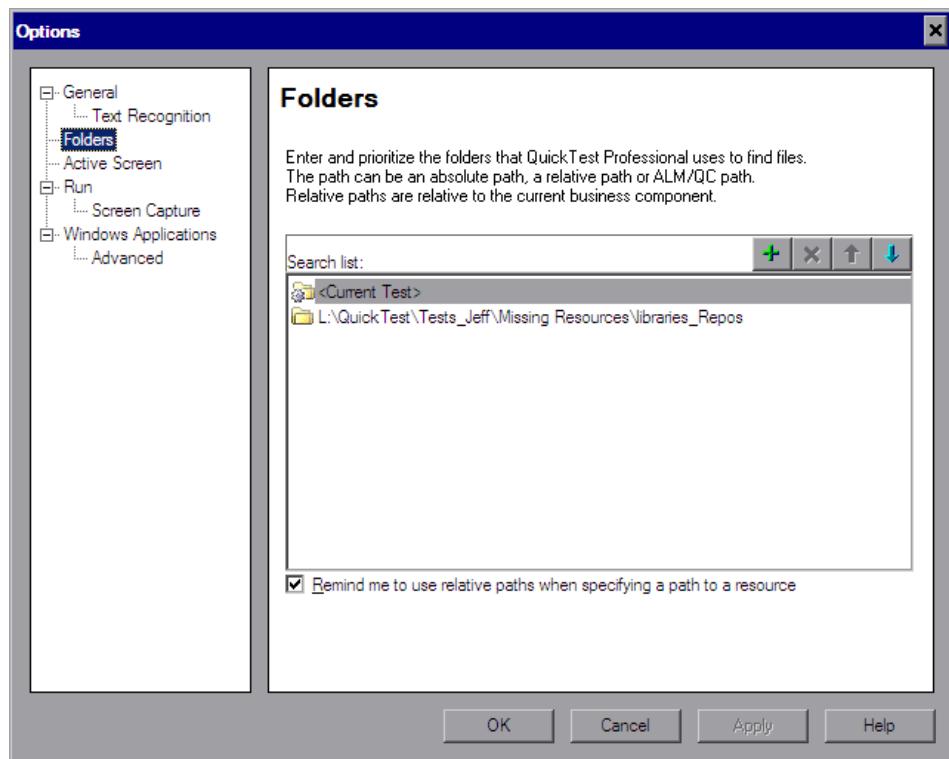
- Guidelines for Windows Vista, Windows 7, Server 2008, and Server 2008 R2 Users:

Windows Vista, Windows 7, Windows Server 2008, and Windows Server 2008 R2 do not support the Windows API test recognition mechanism. Therefore, only the **Use only OCR** option is available when QuickTest is installed on these operating systems.

Folders Pane (Options Dialog Box)

This pane enables you to enter the folders (search paths) in which QuickTest searches for tests, actions, or resource files that are specified as relative paths in dialog boxes and steps.

For example, suppose you add the folder in which all of your tests are stored to the folders list. If you later insert a copy of an action to a test, you only have to enter the name of the test containing the action you want to insert in the Insert Copy of Action dialog box. QuickTest searches for the test's path in the folders you specified in the Folders pane.



To access	Select Tools > Options > Folders node.
Important information	<ul style="list-style-type: none"> ➤ The current test is listed in the Search list by default. It cannot be deleted. ➤ If you are working with the Resources and Dependencies model with Quality Center 10.00 or HP ALM, specify an absolute Quality Center path. For more information, see "Relative Paths and Quality Center" on page 1656. ➤ You can use a <code>PathFinder.Locate</code> statement in your test to retrieve the complete path that QuickTest will use for a specified relative path based on the folders specified in the Folders pane. For more information, see the <i>HP QuickTest Professional Object Model Reference</i>. ➤ QuickTest searches for the specified test, action, or file in the order in which the folders are displayed in the search list. If the same file name exists in more than one folder, QuickTest uses the first instance it finds.
See also	<ul style="list-style-type: none"> ➤ For more information on relative or absolute paths, see "Relative Paths in QuickTest" on page 391.

User interface elements are described below:

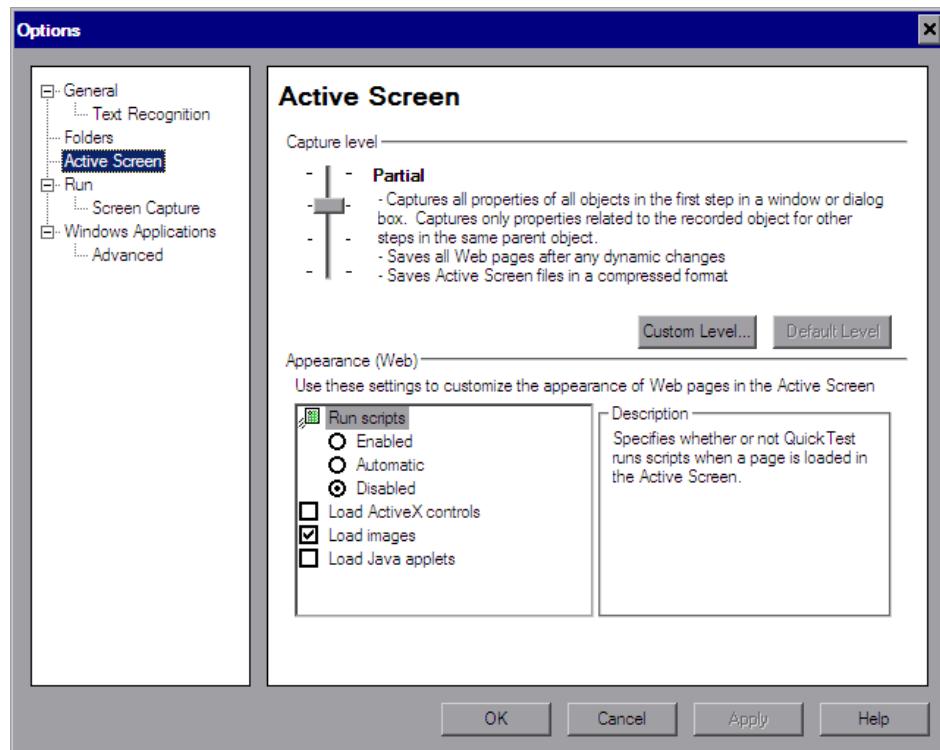
UI Element	Description
Search list	Indicates the folders in which QuickTest searches for tests, actions, or files. If you define folders here, you do not need to designate the full path of a test, action, or file in other dialog boxes or call statements. The order of the search paths in the list determines the order in which QuickTest searches for a specified action or file.

UI Element	Description
	<p>Adds a new folder to the search list.</p> <p>Tips:</p> <ul style="list-style-type: none"> ➤ To add an ALM/QC path when connected to Quality Center, click this button. QuickTest adds [QualityCenter], and displays a browse button so that you can locate the Quality Center path. ➤ When not connected to Quality Center, hold the SHIFT key and click this button. QuickTest adds [QualityCenter], and you enter the path. You can also type the entire Quality Center path manually. If you do, you must add a space after [QualityCenter]. For example: [QualityCenter] Subject\Tests. ➤ Note that QuickTest searches Quality Center project folders only when you are connected to the corresponding Quality Center project.
	Deletes the selected folder from the search list.
	Moves the selected folder up in the list.
	Moves the selected folder down in the list.
Remind me to use relative paths when specifying a path to a resource	<p>When saving a resource, you can choose to be prompted to use a relative path. For more information, see "Relative Paths in QuickTest" on page 391.</p> <p>Note: When QuickTest is connected to a Quality Center 10.00 or HP ALM project, a reminder is displayed only if you select a path in the file system or in a Quality Center 9.2 project.</p>

Active Screen Pane (Options Dialog Box)

This pane enables you to specify which information QuickTest saves and displays in the Active Screen while recording and running tests.

The more information saved in the Active Screen, the easier it is to edit the test after it is recorded. However, more information saved in the Active Screen adds to the recording time and disk space required. This is especially critical with Windows-based add-ins, as they require more disk space to save Active Screen data.



To access	Select Tools > Options > Active Screen node.
Important information	When you are recording on an MDI (Multiple Document Interface) application, the Active Screen does not capture information for non-active child frames.

Related tasks	<ul style="list-style-type: none"> ➤ "How to Modify the Active Screen Settings" on page 1314 ➤ "How to Use the Active Screen in Your Test" on page 1312 ➤ "Updating all Active Screen Captures in a Test Using Update Run Mode" on page 1271
See also	<ul style="list-style-type: none"> ➤ "Custom Active Screen Capture Settings Dialog Box" on page 1438 ➤ "Active Screen Overview" on page 1308

The Active Screen pane contains the following key areas:

- "Capture Level Area" on page 1435
- "Appearance (Web) Area" on page 1436

Capture Level Area

UI Element	Description
Capture level	<p>Specifies the objects for which QuickTest stores data in the Active Screen.</p> <p>Use the slider to select one of the following options:</p> <ul style="list-style-type: none"> ➤ Complete. Captures all properties of all objects in the application's active window/dialog box/Web page in the Active Screen of each step. This level saves Web pages after any dynamic changes and saves Active Screen files in a compressed format. ➤ Partial. (Default.) Captures all properties of all objects in the application's active window/dialog box/Web page in the Active Screen of the first step performed in an application's window, plus all properties of the recorded object in subsequent steps in the same window. This level saves Web pages after any dynamic changes and saves Active Screen files in a compressed format. ➤ Minimum. Captures properties only for the recorded object and its parent in the Active Screen of each step. This level saves the original source HTML of all Web pages (prior to dynamic changes) and saves Active Screen files in a compressed format. ➤ None. Disables capturing of Active Screen files for all applications and Web pages.

UI Element	Description
Custom Level	Enables you to specify custom Active Screen options. For more information, see "Custom Active Screen Capture Settings Dialog Box" on page 1438.
Default Level	Returns the capture level settings to the predefined default level (Partial).

Appearance (Web) Area

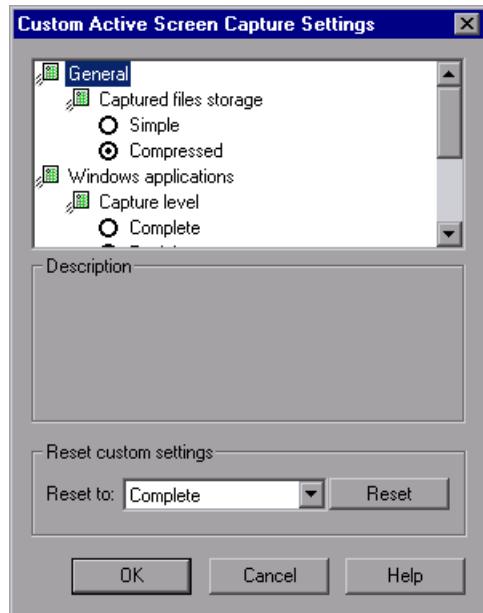
UI Element	Description
Appearance (Web)	<p>Enables you to modify how QuickTest displays captured Web pages in the Active Screen.</p> <p>Notes:</p> <ul style="list-style-type: none"> ➤ QuickTest loads ActiveX controls or Java applets to the Active Screen in view-only mode. You cannot perform operations or retrieve additional information on the loaded ActiveX or Java objects. To perform operations on these items from the Active Screen, you must load the relevant add-in and then record directly on the ActiveX or Java object. ➤ ActiveX controls or Java applets that are loaded to the Active Screen may not work exactly as they do in the application. In some cases, this may cause unexpected behavior, depending on the implementation of the specific controls or applets that are loaded.

UI Element	Description
Run scripts	<p>Specifies whether QuickTest runs scripts when a page is loaded in the Active Screen, according to one of the following options:</p> <ul style="list-style-type: none"> ➤ Enabled. Runs scripts whenever loading a page in the Active Screen. ➤ Automatic. Runs scripts as needed, according to the page that is displayed. ➤ Disabled. Prevents scripts from running when loading a page in the Active Screen. <p>Note: This option refers only to scripts that run automatically when the page loaded. It does not enable you to activate scripts in the Active Screen by performing an operation on the screen.</p>
Load ActiveX controls	<ul style="list-style-type: none"> ➤ Instructs QuickTest to load ActiveX controls from your browser page to the Active Screen, so that for each step you can preview how the page is actually displayed in the application. If this option is cleared, a default ActiveX image is displayed in the Active Screen for all ActiveX control objects.
Load images	<ul style="list-style-type: none"> Instructs QuickTest to load images from your browser page to the Active Screen
Load Java applets	<ul style="list-style-type: none"> Instructs QuickTest to load Java applets from your browser page to the Active Screen, so that for each step you can preview how the page is actually displayed in the application. If this option is cleared, a default Java image is displayed in the Active Screen for all Java applet objects.

Custom Active Screen Capture Settings Dialog Box

This dialog box enables you to customize how QuickTest captures and saves Active Screen information.

When you apply custom Active Screen settings, you override the capture-level setting in the Active Screen pane with all of the settings in the Custom Active Screen Capture Settings dialog box.



To access	Select Tools > Options > Active Screen node > Custom Level .
Important information	<ul style="list-style-type: none">➤ The Custom Active Screen Capture Settings dialog box may also contain options applicable to any QuickTest add-ins installed on your computer.➤ The default settings in the Custom Active Screen Capture Settings dialog box do not reflect the selected capture-level setting in the Active Screen pane of the Options dialog box. If you want to customize only specific settings, use the Reset to option to ensure that all other settings are using the capture-level setting you prefer and then modify the specific settings you need.
Related tasks	<ul style="list-style-type: none">➤ You can specify whether to save screen captures of the Active Screen using the Save Active Screen files option in the Save dialog box. See "Save Test Dialog Box" on page 412.➤ You can use the Update Run Mode option to modify the amount of information saved in the Active Screen after you modify the Active Screen capture settings. See "Update Options Tab (Update Run Dialog Box)" on page 1271.

User interface elements are described below (unlabeled elements are shown in angle brackets):

UI Element	Description
<settings box>	<p>Enables you to select the specific setting options that determine how QuickTest captures and saves Active Screen information. The Settings box may also contain options applicable to QuickTest add-ins installed on your computer.</p> <p>For details, see:</p> <ul style="list-style-type: none"> ▶ "General" on page 1441 ▶ "Capture Level - General" on page 1441 ▶ "Capture Level - Java Applications or Applets" on page 1442 ▶ "Capture Level - SAP GUI for Windows Applications" on page 1443 ▶ "Capture Level - Oracle Applications" on page 1444 ▶ "Capture Level - Windows Applications" on page 1445 ▶ "Capture Level - Terminal Emulator Applications" on page 1446 ▶ "Web" on page 1446
Description	<p>Provides a description of the option selected in the Settings box.</p>
Reset custom settings	<p>Enables you to reset the custom settings to one of the predefined levels provided with QuickTest (Complete, Partial, Minimum, or None) by choosing a level from the Reset to list and clicking the Reset button. For more information on the available capture levels, see "Capture Level - General" on page 1441.</p>

General

By selecting a **Captured files storage** option, you can specify the type of compression QuickTest uses for storing captured Active Screen information:

- **Simple.** Instructs QuickTest to save Active Screen captures in standard uncompressed file formats (for example, **.html** and **.png**).
- **Compressed.** Instructs QuickTest to save Active Screen captures in a compressed (zipped) file format. Using this option saves disk space, but it may affect the time it takes to load images in the Active Screen. This is the default option.

Capture Level - General

The Custom Active Screen Capture Settings dialog box enables you to customize how QuickTest captures and saves Active Screen information.

Capture level options are available for Java applets or applications, SAP GUI for Windows applications, Oracle applications, Windows-based applications, and Terminal Emulator applications. The options available in the Custom Active Screen Capture Settings dialog box depend on the add-ins that are installed.

By selecting a **Capture level** option, you can specify which properties are captured for each object in an application when it is captured for the Active Screen.

Depending on your testing requirements, you can choose between different levels of Active Screen capture. However, you should take into consideration that the less information captured for the Active Screen, the better the performance.

For example, if you select the **Complete** capture level option, you can add checkpoints on every test object displayed in any Active Screen capture after recording, but it will take more time and use more disk space to record a single operation. Selecting **Partial** enables QuickTest to record faster and use less disk space, but there may be limitations on the operations you can perform from the Active Screen after recording.

The following sections describe the capture level options available for different environments.

Capture Level - Java Applications or Applets

The following **Capture level** options are available for Java applications or Java applets:

- **Complete.** Instructs QuickTest to save all identification properties of all objects in the application or applet's open window/dialog box in the Active Screen of each step.
- **Partial.** (Default) Instructs QuickTest to save all identification properties of all objects in the application or applet's open window/dialog box in the Active Screen of the first step performed in that window, plus all properties of the recorded object only, in subsequent steps in the same window.
- **Minimum.** Instructs QuickTest to save all identification properties for the recorded object plus all identification properties for the parent objects in the recording hierarchy.
- **None.** Disables capture of Active Screen files for Java applications or Java applets.

When the **Complete** or **Minimum** capture level is selected, the following setting in the Custom Active Screen Capture Settings dialog box is also relevant for Java applications or Java applets:

Disable capture of the following objects. Prevents QuickTest from capturing the data of steps performed on other objects for the selected test object types in the Active Screen. These objects will be visible in the Active Screen as images only.

By default, **JavaObject** and **JavaMenu** are selected (meaning that identification properties are not captured for these objects).

Note: If you record on a specific test object, its identification properties will be captured even if the **Disable capture of the following objects** option is selected.

Capture Level - SAP GUI for Windows Applications

The following **Capture level** options are available for SAP GUI for Windows applications:

- **Complete.** Instructs QuickTest to save the property values of all objects in the application's open window/dialog box in the Active Screen of each step.

This option makes it possible for you to insert checkpoints and perform other operations on any object in the window/dialog box from the Active Screen of any step. However, it may result in longer recording times and require more disk space.

Note: The properties for inner objects of some container objects (such as table cells or tree nodes) are not captured in the Active Screen. Use the appropriate SAPGuiTable or SAPGuiTree methods to access information for these objects. For more information, see the **SAP GUI for Windows** section of the *HP QuickTest Professional Object Model Reference*.

- **Partial.** (Default) Instructs QuickTest to save properties of the recorded object and of its parent in the Active Screen of each step.

This option enables speedy recording and requires relatively little disk space. However, you can insert checkpoints and perform other operations only on the recorded object and on the window/dialog box itself. You cannot perform operations on the other objects displayed in the Active Screen.

- **None.** Disables the capture of Active Screen files for SAP GUI for Windows applications.

This option allows extremely fast recording and requires only minimum disk space. However, you cannot perform post-recording test editing (such as inserting checkpoints, output values, and so forth) from the Active Screen.

Notes:

- The property values of the objects in the Active Screen reflect the values at the time that the steps are added to your test (when information is sent to the SAP server). These values may potentially be different from the property values at the time that a particular step is performed.
 - The Active Screen captures only the visible part of the SAP GUI for Windows applications window at the time that the step is added to the test.
-

Capture Level - Oracle Applications

The following **Capture level** options are available for Oracle applications:

- **Complete.** Instructs QuickTest to save all identification properties of all objects in the application's open window/dialog box in the Active Screen of each step.
- **Partial.** (Default) Instructs QuickTest to save all identification properties of all objects in the application's open window/dialog box in the Active Screen of the first step performed in that window, plus all identification properties of the recorded object only, in subsequent steps in the same window.
- **Minimum.** Instructs QuickTest to save all identification properties for the recorded object plus all identification properties for the parent objects in the recording hierarchy.
- **None.** Disables capture of Active Screen files for Oracle applications.

Capture Level - Windows Applications

The following **Capture level** options are available for Windows applications.

- **Complete.** Instructs QuickTest to save all properties of all objects in the application's open window/dialog box in the Active Screen of each step.

This option makes it possible for you to insert checkpoints and perform other operations on any object in the window/dialog box, from the Active Screen for any step.

- **Partial (Default).** Instructs QuickTest to save all properties of all objects in the application's open window/dialog box in the Active Screen of the first step performed in an application's window, plus all properties of the recorded object in subsequent steps in the same window.

This option makes it possible for you to insert checkpoints and perform other operations on any object displayed in the Active Screen, while conserving recording time and disk space. Note that with this option the Active Screen information may not be fully updated for subsequent steps.

- **Minimum.** Instructs QuickTest to save properties only for the recorded object and its parent in the Active Screen of each step.

This option enables speedy recording and requires relatively little disk space. However, you can insert checkpoints and perform other operations only on the recorded object and on the window/dialog box itself. You cannot perform operations on the other objects displayed in the Active Screen.

- **None.** Disables capture of Active Screen files for Windows applications.

This option allows extremely fast recording and requires only a minimum of disk space. However, you cannot perform post-recording test editing from the Active Screen.

Capture Level - Terminal Emulator Applications

The following **Capture level** options are available for applications run on terminal emulators:

- **Complete.** Instructs QuickTest to save all properties of all objects in the application's open window/dialog box in the Active Screen of each step. This option makes it possible for you to insert checkpoints and perform other operations on any object in the window/dialog box, from the Active Screen for any step.
- **None.** Disables capture of Active Screen files for Terminal Emulator applications.

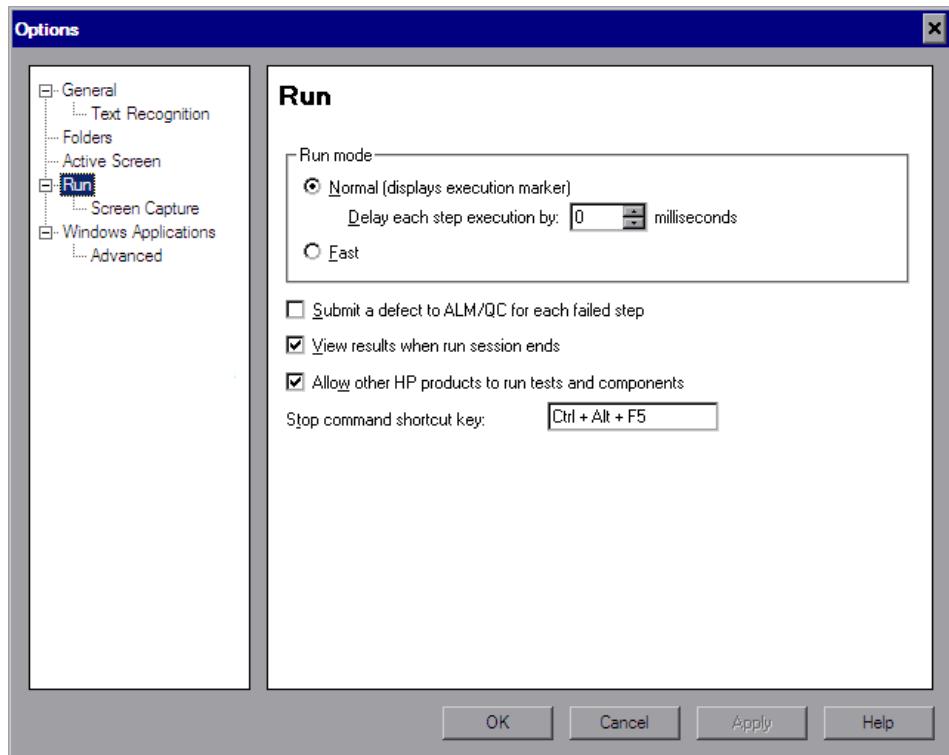
Web

You can specify whether QuickTest captures Web pages for the Active Screen.

- **Disable Active Screen capture.** Disables the screen capture of all steps in the Active Screen.
If you do not select this option, you can also delete Active Screen information after you have finished editing your test by selecting **Save As**, and clearing the **Save Active Screen files** check box. For more information, see "Save Test Dialog Box" on page 412.
- **Capture original HTML source.** Captures the HTML source of Web pages as they appear originally, before any scripts are run. Deselecting this option instructs QuickTest to capture the HTML source of Web pages after any dynamic changes have been made to the HTML source (for example, by scripts running automatically when the page is loaded).

Run Pane (Options Dialog Box)

This pane enables you to determine how QuickTest runs tests.



To access	Select Tools > Options > Run node.
See also	"Screen Capture Pane (Options Dialog Box)" on page 1450

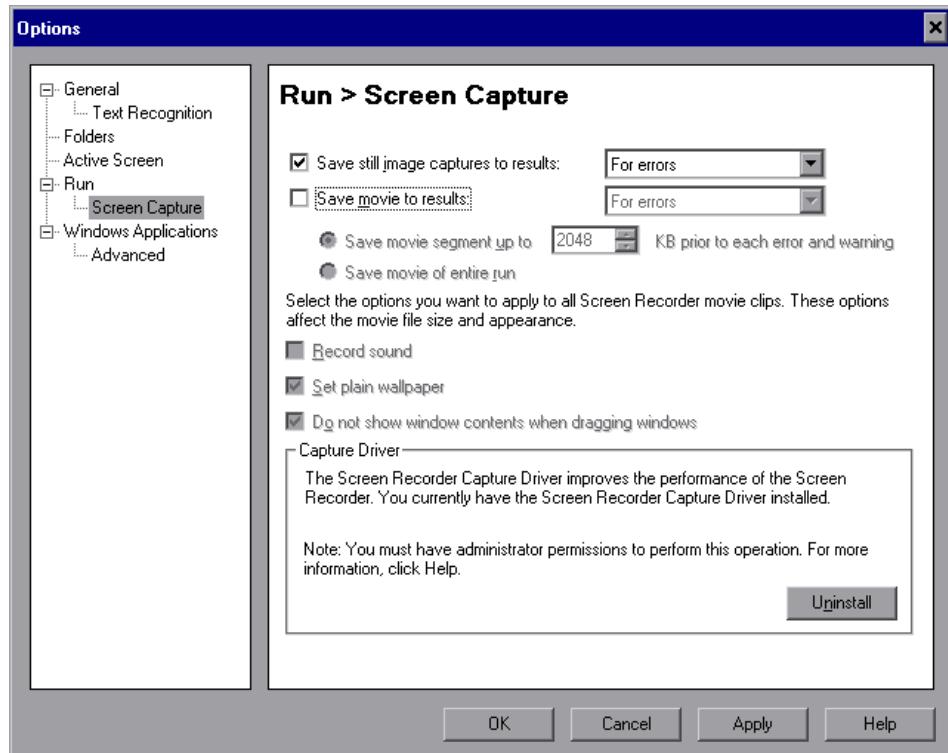
User interface elements are described below:

UI Element	Description
Run mode	<p>Instructs QuickTest how to run your test:</p> <ul style="list-style-type: none"> ➤ Normal (displays execution marker). Runs your test with the execution arrow to the left of the Keyword View or Expert View, marking each step or statement as it is performed. If the test contains multiple actions, the tree in the Keyword View Item column expands to display the steps, and the Expert View displays the script, of the currently running action. Delay each step execution by. You can specify the time in milliseconds that QuickTest should wait before running each consecutive step (up to a maximum of 10000 ms.) The Normal run mode option requires more system resources than the Fast option, described below. <p>Note: You must have Microsoft Script Debugger installed to enable this mode. For more information, see the <i>HP QuickTest Professional Installation Guide</i>.</p> <ul style="list-style-type: none"> ➤ Fast. Runs your test without the execution arrow to the left of the Keyword View or Expert View and does not expand the item tree or display the script of each action as it runs. This option requires fewer system resources. <p>Note: When running a test set from Quality Center, tests and function libraries are automatically run in Fast mode, even if Normal mode is selected.</p>
Submit a defect to ALM/QC for each failed step	<p>Instructs QuickTest to automatically submit a defect to ALM/QC for each failed step in your test. This option is available only when you are connected to a Quality Center project. For more information, see "How to Automatically Submit Defects to a Quality Center Project" on page 1100.</p>
View results when run session ends	<p>Instructs QuickTest to display the results automatically following the run session.</p>

UI Element	Description
Allow other HP products to run tests and components	<p>Enables other HP products such as Quality Center and Test Batch Runner to run QuickTest tests on this computer.</p> <p>Note: This option is not required to enable WinRunner and Service Test to run QuickTest tests.</p>
Stop command shortcut key	<p>Enables you to define a shortcut key or key combination that stops the current QuickTest record or run operation, even if QuickTest is not in focus or is in hidden mode.</p> <p>Click in the field and then press the required key or key combination on the keyboard.</p> <p>The default key combination is Ctrl+Alt+F5.</p> <p>Note: It is important to define a shortcut that is not already defined for some other operation by the application being tested. If this is the case and:</p> <ul style="list-style-type: none"> ➤ you open the application manually before you click Record or Run, the shortcut defined in the application will apply for its original purpose. ➤ you start a record or run session and QuickTest opens the application for you, the shortcut you define in the Run pane will stop the session.

Screen Capture Pane (Options Dialog Box)

This pane enables you to control when and how QuickTest captures screens of the application being tested.



To access	Select Tools > Options > Run node > Screen Capture node.
Important information	Vista users: In addition to the options described below, if your Vista Windows color scheme is set to Aero , QuickTest automatically sets it to Vista Basic while capturing movies of a run session to maximize performance. The color scheme is returned to its previous settings when the run session ends. For movies, see Save movie to results under General , below.
See also	"How to Play a Screen Recorder Movie in the HP Micro Player" on page 1104

This dialog box contains the following key elements:

General

UI Element	Description
Save still image captures to results	<p>Instructs QuickTest when to capture still images of the application during the run session and save them in the run results. When images are available in the run results, QuickTest displays them in the Captured Data pane in the Run Results Viewer.</p> <p>Clear the check box to disable this option, or select an option from the list:</p> <ul style="list-style-type: none">➤ Always. Captures images for all steps in the run.➤ For errors. Captures images only for failed steps. This is the default setting.➤ For errors and warnings. Captures images only for steps that return a failed or warning status. <p>Note: This setting also affects the availability of other information displayed in the bottom pane of the run result details, such as:</p> <ul style="list-style-type: none">➤ XML checkpoint and output value result details➤ Bitmap checkpoint images (expected, actual, and difference)

UI Element	Description
Save movie to results	<p>Instructs QuickTest when to capture a movie of the application during the run session and save it in the run results. When movies are available in the run results, QuickTest displays them in the Screen Recorder tab in the Run Results Viewer.</p> <p>This option is disabled by default.</p> <p>Select the check box to enable this option and then select an option from the list:</p> <ul style="list-style-type: none"> ➤ Always. Captures a movie of all steps in the run. ➤ For errors. Captures movies only for failed steps. ➤ For errors and warnings. Captures movies only for steps that return a failed or warning status. <p>Important:</p> <ul style="list-style-type: none"> ➤ Records only on primary monitor: The Screen Recorder records a movie of the operations performed on your primary monitor. Therefore, if you are working with multiple monitors, make sure that your application is fully visible on your primary monitor when recording or running a test. For more information, see "Save Movie to Results" on page 1453. ➤ Records entire desktop. The Screen Recorder saves a movie of your entire desktop. You can prevent the QuickTest window from partially obscuring your application while capturing the movie by minimizing QuickTest during the run session. For information on how to minimize QuickTest during run sessions, see "QuickTest Window Layout Customization in Different Modes" on page 1280. ➤ Vista Users. For best results when working with Microsoft Windows Vista, set the Window's display settings to the Classic Windows theme. ➤ Unicode Characters in Test Names. The HP Screen Recorder does not support Unicode. When you save a test using a name containing Unicode characters, and the run results contain a Screen Recorder movie, the movie is not saved.

Save Movie to Results

UI Element	Description
Save movie segment up to ___ KB prior to each error and warning	<p>When selected, QuickTest saves movie segments for each error (or warning). Each segment contains the specified number of kilobytes of the movie prior to the failed (or warning) step. You can enter any value from 400 (0.4 MB) to 2097152 (2 GB). If more than one segment is captured for a run session, QuickTest stores a single movie comprised of all the relevant movie segments.</p> <p>Note: This option is enabled only when For errors or For errors and warnings is selected in the Save movie to results option.</p>
Save movie of entire run	<p>When selected, QuickTest saves a movie of the entire run if at least one error (or warning) occurs.</p> <p>Note: This option is enabled only when For errors or For errors and warnings is selected in the Save movie to results option.</p>
Record sound	<p>Instructs QuickTest to save sound with the movie of your application.</p>
Set plain wallpaper	<p>Sets the wallpaper of your desktop to a solid blue color for the duration of the run session.</p>
Do not show window contents when dragging windows	<p>Instructs Windows to display only the outline of a window, without its contents, whenever the window is dragged during the run session.</p>

Capture Driver Area

UI Element	Description
Install/Uninstall button	<p>Installs or uninstalls the Screen Recorder Capture Driver. The Screen Recorder Capture Driver improves the performance of the Screen Recorder during movie recording.</p> <p>Note: The Screen Recorder Capture Driver cannot be installed or uninstalled when running QuickTest via a remote connection.</p>

46

Individual Test Settings

This chapter includes:

Concepts

- [Test Settings Overview on page 1456](#)
- [Add-in Associations in Your Test on page 1458](#)
- [Associated Function Libraries on page 1459](#)
- [Local System Monitor on page 1460](#)
- [Log Tracking on page 1461](#)

Tasks

- [How to Manually Configure Log Tracking Settings on page 1462](#)

Reference

- [Properties Pane \(Test Settings Dialog Box\) on page 1466](#)
- [Run Pane \(Test Settings Dialog Box\) on page 1471](#)
- [Resources Pane \(Test Settings Dialog Box\) on page 1475](#)
- [Parameters Pane \(Test Settings Dialog Box\) on page 1479](#)
- [Environment Pane \(Test Settings Dialog Box\) on page 1483](#)
- [Recovery Pane \(Test Settings Dialog Box\) on page 1490](#)
- [Local System Monitor Pane \(Test Settings Dialog Box\) on page 1495](#)
- [Log Tracking Pane \(Test Settings Dialog Box\) on page 1498](#)

Concepts

Test Settings Overview

You can use the Test Settings dialog box to set testing options that affect how QuickTest works with a specific test. For example, you can instruct QuickTest to run a parameterized test for only certain lines in the data table. The individual testing options that you specify are saved when you save the test.

Note: You can also set testing options that affect all tests. For details, see Chapter 45, "Global Testing Options."

The Test Settings dialog box contains the following panes:

Node	Options
Properties	Options for setting the properties of the test, for example, its description and associated add-ins. For details, see "Properties Pane (Test Settings Dialog Box)" on page 1466.
Run	Options for setting the run session preferences. For details, see "Run Pane (Test Settings Dialog Box)" on page 1471.
Resources	Options for specifying resources you want to associate with your test, such as function libraries and data tables. For details, see "Resources Pane (Test Settings Dialog Box)" on page 1475.
Parameters	Options for specifying input and output parameters for your test. For details, see "Parameters Pane (Test Settings Dialog Box)" on page 1479.

Node	Options
Environment	Options for viewing existing built-in and user-defined environment variables, adding, modifying and saving user-defined environment variables, and selecting the active external environment variables file. For details, see "Environment Pane (Test Settings Dialog Box)" on page 1483.
Recovery	Options for setting how QuickTest recovers from unexpected events and errors that occur in your testing environment during a run session. For details, see "Recovery Pane (Test Settings Dialog Box)" on page 1490.
Local System Monitor	Options for activating and setting preferences for tracking system counters during a run session. For details, see "Local System Monitor Pane (Test Settings Dialog Box)" on page 1495.
Log Tracker	Options for activating and setting run-time preferences for tracking log messages generated by the log framework that monitors events occurring in your application. For details, see "Log Tracking Pane (Test Settings Dialog Box)" on page 1498.

Note: The Test Settings dialog box may also contain additional panes corresponding to any QuickTest add-ins that are installed or loaded. For details on add-ins, see the relevant section in the *HP QuickTest Professional Add-ins Guide*.

Add-in Associations in Your Test

When you open QuickTest, you select the add-ins to load from the Add-in Manager dialog box. You can create and edit tests that work with any environment for which the necessary add-in is loaded.

When you create a new test, the add-ins that are currently loaded are automatically associated with your test.

Choosing to associate an add-in with your test instructs QuickTest to check that the associated add-in is loaded each time you open that test. For details, see "Add-in Manager Dialog Box" on page 113.

When you open a test, QuickTest notifies you if an associated add-in is not currently loaded, or if you have loaded add-ins that are not currently associated with your test. This process ensures that your run session will not fail due to unloaded add-ins and reminds you to add required add-ins to the associated add-ins list if you plan to use them with the currently open test. For details on loading and working with add-ins, see the *HP QuickTest Professional Add-ins Guide*.

Quality Center uses the associated add-ins list to determine which add-ins to load when it opens QuickTest to run or view a test. For details on working with Quality Center, see Chapter 52, "Quality Center Integration."

Associated Function Libraries

The **Associated function libraries** area of the **Resources** pane in the Test Settings dialog box indicates the list of function libraries associated with your test. QuickTest searches these files for the VBScript functions, subroutines, and so forth that are specified in the test.

The order of the function libraries in the list determines the order in which QuickTest searches for a function or subroutine that is called from a step in your test. If there are two functions or subroutines with the same name, QuickTest uses the first one it finds.

Note: If you use the **LoadFunctionLibrary** statement to dynamically load a function library during a run session, and a later step calls a function that has the same name as a function in an associated function library, the function in the dynamically loaded function library is used. For details, see the **Utility** section of the *HP QuickTest Professional Object Model Reference*.

You can enter an associated function library using an absolute or relative path. If you enter it as a relative path, then during a run session, QuickTest searches for the file in the directory for the current test, and then in the folders listed in the Folders pane of the Options dialog box. For details, see "Folders Pane (Options Dialog Box)" on page 1431 and "Relative Paths in QuickTest" on page 391.

Tip:

- ▶ When working with tests, if your function libraries are stored in the file system and you want other users or HP products to be able to run this test on other computers, you can set the file path as a relative path. (Click the path once to highlight it, and then click it again to enter edit mode.) Any users who want to run this test should then specify the drive letter and folder in which QuickTest should search for the relative path in the Folders pane of the Options dialog box (**Tools > Options > Folders** node).
For details, see "Folders Pane (Options Dialog Box)" on page 1431, and "Relative Paths in QuickTest" on page 391.

Caution: If you are working with the Resources and Dependencies model with Quality Center 10.00 or HP ALM, you should store your function libraries in the Quality Center Test Resources module and specify an absolute Quality Center path in the Folders pane. For details, see "Relative Paths and Quality Center" on page 1656.

- ▶ You can also add, delete and prioritize the function libraries associated with your test using the Resources pane. For details, see "Resources Pane" on page 1385.
-

Local System Monitor

You can use the Local System Monitor pane of the Test Settings dialog box (**File > Settings > Local System Monitor** node) to activate and set preferences for tracking system counters during a run session.

The local system monitor tracking options enable you to track application performance counters during a run session. These counters enable you to monitor the resources used by your application.

The system counters that can be monitored are the process counters that are accessible through the performance console (select **Start > Run >** and then enter **Perfmon**). For information on these process counters, see the performance console's Help.

You can also define limits for the counters. If the specified counters exceed these limits, the run session will fail. The results of the system counters are viewed in the Run Results Viewer. For details, see "Run Results Viewer" on page 1085.

For information on enabling system monitoring, see "Local System Monitor Pane (Test Settings Dialog Box)" on page 1495.

Log Tracking

If the Windows-based application you are testing uses a supported Java or .NET log framework that includes a UDP appender, you can enable QuickTest to receive log messages from that framework and send them to the run results.

You do this by configuring your application's log configuration file. You can configure this file using QuickTest, or you can configure the file manually. To configure this file using QuickTest, you define the settings in the Log Tracking pane in the Test Settings dialog box described on page 1461. For details on configuring the log configuration file manually, see "How to Manually Configure Log Tracking Settings" on page 1462.

When you configure your application's log configuration file (via QuickTest or manually), you specify the minimum log message level that you want QuickTest to receive, the log message source and port, and so on. During a run session, QuickTest receives the relevant log messages that are generated for your application and sends them to the run results. In the Run Results Viewer, these log messages are displayed in the Log Tracking Results pane. In the Run Results Viewer, log messages are associated with steps according to their timestamp. For details, see "Log Tracking Pane (Run Results Viewer)" on page 1132.

Analyzing the log messages that were generated as a result of a particular step can help you pinpoint the causes of unexpected behavior in your application, such as application crashes during automated runs.

For more details, see:

- "How to Manually Configure Log Tracking Settings" on page 1462
- "Log Tracking Pane (Test Settings Dialog Box)" on page 1498

Tasks

How to Manually Configure Log Tracking Settings

You can specify your log tracking and collection preferences manually. This is useful, for example, if your test is validating more than one application, and you need to configure a file for each application being tested.

Note: If you want to instruct QuickTest to automatically configure your application's log configuration file, use the Log Tracking Pane (Test Settings Dialog Box) described on page 1498.

The following steps describe how to manually specify your log tracking and collection preferences.

- "Prerequisites" on page 1463
- "Open the relevant log configuration file and specify your preferences" on page 1463
- "Configure the settings in the Log Tracking pane so that QuickTest will use the same settings that you defined in the previous step" on page 1465
- "Results" on page 1465

1 Prerequisites

- a** Your Windows-based application must use a Java or .NET log framework that includes a UDP appender.
- b** Verify the following:
 - Which log framework version is used with your application? Make sure that it includes a UDP appender and that the specific version is supported when working with QuickTest.
 - Make sure that logging is enabled on your application. Find out how to enable and disable logging. (You may want to enable logging for some run sessions and disable logging for others.) Do you need to define an XML parameter or modify a registry key?
 - Where is the log configuration file located? Make sure that it is writable.

2 Open the relevant log configuration file and specify your preferences

This step describes how to manually configure a log configuration file so that QuickTest will be able to receive log messages during a run session.

For a list of supported log frameworks, see the *HP QuickTest Professional Product Availability Matrix*, available from the Documentation Library Home page or the root folder of the QuickTest Professional DVD.

- a** Do the following:

1. Add an **appender-ref ref** attribute with the value of **QtpUdpAppender** to the **root** element.

2. Specify the minimum log message level that you want QuickTest to include in the run results.

Example:

```
...
<root>
<root>
    <level value="DEBUG" />
...
    <appender-ref ref="QtpUdpAppender" />
</root>
```

- b** Add an **appender** element and its attributes, as shown in the following example.

Example:

```
<appender name="QtpUdpAppender"
    type="log4net.Appender.UdpAppender">
    <remoteAddress value="1.1.1.1" />
    <remotePort value="18081" />
    <encoding value="utf-16" />
    <layout type="log4net.Layout.XmlLayoutSchemaLog4j">
        <prefix value="" />
    </layout>
</appender>
```

Note: To enable QuickTest to receive log messages, the **Add log messages to run results** area of the Log Tracking pane of the Test Settings dialog box must also be configured, as described below.

3 Configure the settings in the Log Tracking pane so that QuickTest will use the same settings that you defined in the previous step

This step enables QuickTest to receive log messages during a run session.

In the Log Tracking pane of the Test Settings dialog box:

- Select the **Add log messages to run results** check box.
- Specify the settings in the upper half of the pane:
 - **Log message source**
 - **Port**
 - **Minimum level to add node to results tree**

For details on these fields, see "Log Tracking Pane (Test Settings Dialog Box)" on page 1498.

- Clear the **Auto-configure log mechanism** check box. This prevents QuickTest from modifying the configuration file.

4 Results

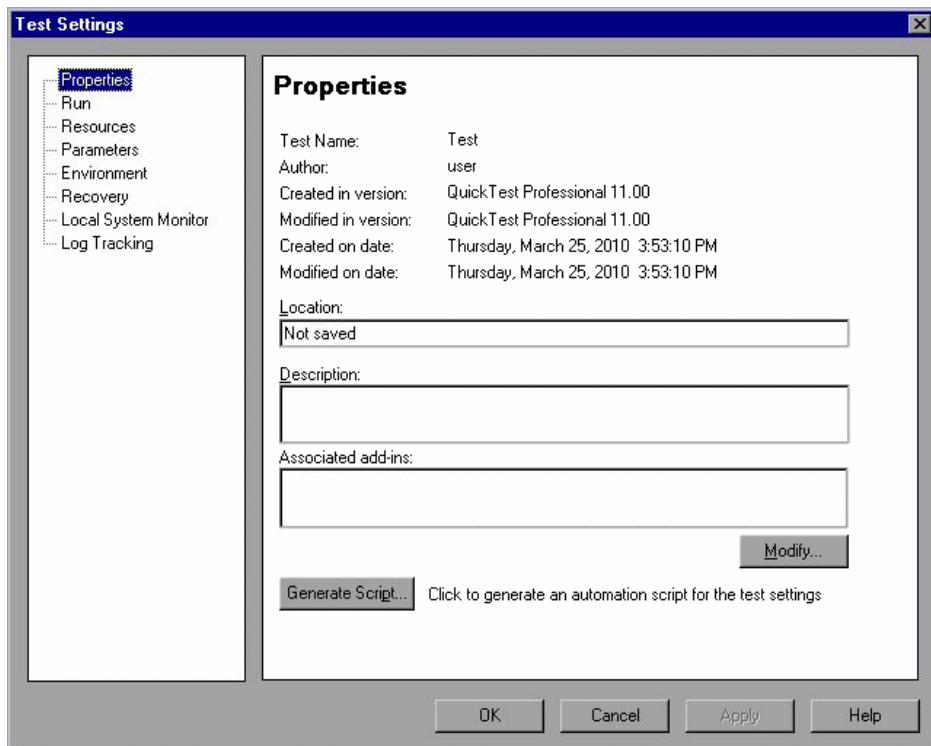
During a run session, QuickTest receives any log messages that match or exceed the minimum log message level that you specified in step on page 1464 and displays them in the run results.

In the run results tree, QuickTest also inserts a node for the first message that matches or exceeds the **Minimum level to add node to results tree**. This node is inserted directly after the step that triggered (or preceded) the relevant log message (according to its timestamp).

Reference

Properties Pane (Test Settings Dialog Box)

This pane enables you to view and define general information about your test, including the add-ins associated with it. You can also choose to generate an automation script for the test settings.



To access	Select File > Settings > Properties .
See also	"Modify Associated Add-ins Dialog Box" on page 1469

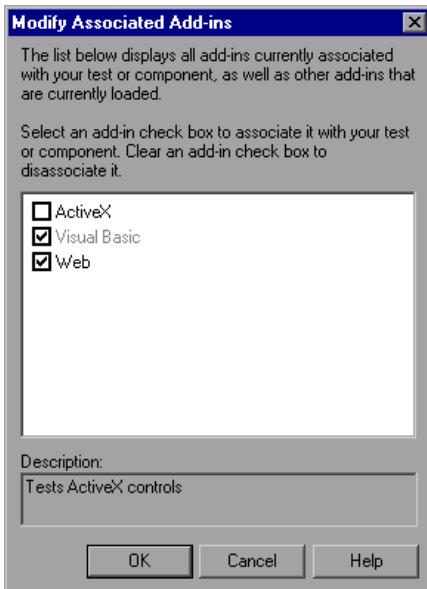
User interface elements are described below:

UI Element	Description
Name	Indicates the name of the test. If the test is saved in a version-controlled project in Quality Center, the version number is also shown.
Author	Indicates the Windows user name of the person who created the test.
Created in version	Indicates the version of QuickTest used to create the test.
Modified in version	Indicates the version of QuickTest last used to modify the test.
Created on date	Indicates the date and time that the test was created.
Modified on date	Indicates the date and time that the test was last modified.
Location	Indicates the path and filename of the test.
Description	Enables you to specify a description for your test.
Associated add-ins	Lists the add-ins associated with the test. For details, see "Add-in Associations in Your Test" on page 1458.
Modify	Enables you to select the add-ins to associate with your test. For details, see "Modify Associated Add-ins Dialog Box" on page 1469.

UI Element	Description
Generate Script	<p>Generates a QuickTest automation script containing the current test settings.</p> <p>When you click the Generate Script button, a Save As dialog box opens, enabling you to specify the name and file system location to store the generated file.</p> <p>You can use some or all of the script lines from this generated script in an automation script. This can be useful, for example, if you want to open a test with required test settings already set or to set the same test settings for multiple tests by looping through the tests in a folder.</p> <p>For details, see "QuickTest Automation Scripts" on page 1589 and the <i>QuickTest Professional Automation Object Model Reference</i> (Help > QuickTest Professional Help > HP QuickTest Professional Advanced References > HP QuickTest Professional Automation Object Model).</p>

Modify Associated Add-ins Dialog Box

This dialog box lists all the add-ins currently associated with your test, as well as any other add-ins that are currently loaded in QuickTest, and enables you to associate or disassociate add-ins with your test.



To access	In the Properties Pane (Test Settings Dialog Box), click Modify .
Important information	<ul style="list-style-type: none"> ➤ If this dialog box contains a child add-in, and you select it, the parent add-in is selected automatically. ➤ If you clear the check box for a parent add-in, the check boxes for its children are also cleared. ➤ If a specific add-in is not currently loaded, but you want to associate it with your test, reopen QuickTest and load the add-in from the Add-in Manager. If the Add-in Manager dialog box is not displayed when you open QuickTest, you can choose to display it the next time you open QuickTest. To do so, select Display Add-in Manager on startup from the General pane of the Options dialog box. For details, see "Global Testing Options" on page 1419.

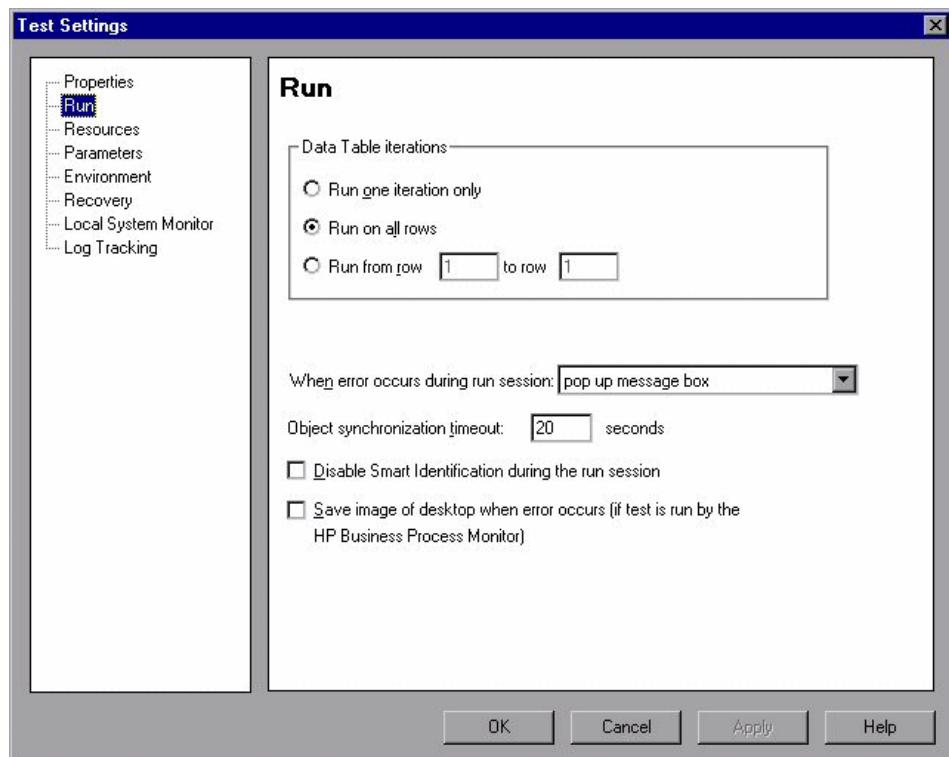
See also	<ul style="list-style-type: none"> ➤ "Add-in Associations in Your Test" on page 1458 ➤ For details on the Add-in Manager, see the section on working with QuickTest add-ins in the <i>HP QuickTest Professional Add-ins Guide</i>.
-----------------	--

User interface elements are described below (unlabeled elements are shown in angle brackets):

UI Elements	Description
<Add-ins list>	<p>The list of the add-ins that are currently loaded in QuickTest. You can select the check boxes for add-ins that you want to associate with your test, or clear the check boxes for add-ins that you do not want to associate with your test.</p> <p>In the image above:</p> <ul style="list-style-type: none"> ➤ Web is loaded and associated with the test. ➤ ActiveX is loaded, but not associated with the test. ➤ Visual Basic is associated with the test, but is not loaded. <p>Note:</p> <ul style="list-style-type: none"> ➤ Add-ins that are associated with your test but not currently loaded are shown dimmed. ➤ This list might also include child nodes representing add-ins that you or a third party developed to support additional environments or controls using add-in extensibility.
Description	<p>The description of the selected add-in.</p>

Run Pane (Test Settings Dialog Box)

This pane enables you to choose what to do when an error occurs during the run session, set the object synchronization timeout, and choose whether or not to disable the Smart Identification mechanism for the test.



To access	Select File > Settings > Run.
-----------	-------------------------------

Important information	<ul style="list-style-type: none"> ➤ This pane applies to the entire test. You can set the run properties for an individual action in a test from the Run tab in the Action Call Properties dialog box of a selected action. For details on action run properties, see "Run Tab (Action Call Properties Dialog Box)" on page 552. ➤ By default, when you run a test with global data table parameters, QuickTest runs the test for each row in the data table using the parameters you specified. For details, see "When to Choose Global or Action Data Table Parameters" on page 737. ➤ You can use this pane to instruct QuickTest to run iterations on a test only for certain lines in the Global tab in the data table.
------------------------------	--

User interface elements are described below:

UI Element	Description
Data Table iterations	<p>Specifies the iterations for the test. Select an option:</p> <ul style="list-style-type: none"> ➤ Run one iteration only. Runs the test only once, using only the first row in the global data table. ➤ Run on all rows. Runs the test with iterations using all rows in the global data table. ➤ Run from row __ to row __. Runs the test with iterations using the values in the global data table for the specified row range.
When error occurs during run session	<p>Specifies how QuickTest responds to an error during the run session. For details, see "Error Response Options" on page 1474.</p> <p>Note: By default, QuickTest automatically continues to the next step when an error occurs in a test that runs from a Quality Center test set or runs using QuickTest automation. If you want to change this setting, contact HP Software Support.</p>

UI Element	Description
Object synchronization timeout	<p>Sets the maximum time (in seconds) that QuickTest waits for an object to load before running a step in the test.</p> <p>Note: When working with Web objects, QuickTest waits up to the amount of time set for the Browser navigation timeout option, plus the time set for the object synchronization timeout. For details on the Browser navigation timeout option, see the <i>HP QuickTest Professional Add-ins Guide</i>.</p>
Disable Smart Identification during the run session	<p>Instructs QuickTest not to use the Smart Identification mechanism during the run session.</p> <p>Note: When you select this option, the Enable Smart Identification check boxes in the Object Properties and Object Repository dialog boxes are disabled, although the settings are saved. When you clear this option, the Enable Smart Identification check boxes return to their previous on or off setting.</p>
Save image of desktop when error occurs (if test is run by the HP Business Process Monitor)	<p>This option is applicable only to tests that are run by the Business Process Monitor component of HP Business Availability Center.</p> <p>Selecting this option instructs QuickTest to capture a snapshot of the desktop if an error occurs during a run session of a test initiated by the Business Process Monitor. The image is saved in Business Availability Center. The Business Process Monitor forwards the run results to the Business Availability Center servers.</p>

Error Response Options

By default, if an error occurs during the run session, QuickTest displays a popup message box describing the error. You must click a button on this message box to continue or end the run session.

You can accept the **popup message box** option or you can specify a different response by choosing one of the alternative options in the list in the **When error occurs during run session** box:

- **proceed to next action iteration.** QuickTest proceeds to the next action iteration when an error occurs.
- **stop run.** QuickTest stops the run session when an error occurs.
- **proceed to next step.** QuickTest proceeds to the next step in the test when an error occurs.

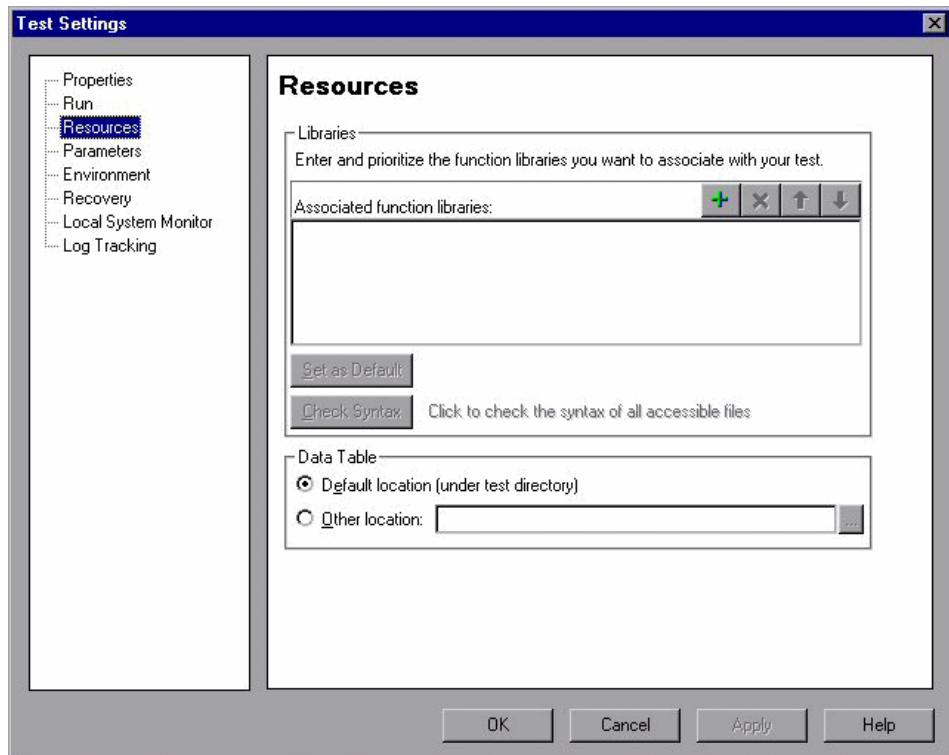
QuickTest first performs any recovery scenarios associated with the test, and performs the option selected above only if the associated recovery scenarios do not resolve the error. For details, see "Recovery Pane (Test Settings Dialog Box)" on page 1490.

Note: If you are working with many tests, you may want to use a QuickTest automation script to set a different value for each test. To access the automation script line that controls this option, you can use the **Generate Script** button in the Properties pane of the Test Settings dialog box.

For details, see "QuickTest Automation Scripts" on page 1589 or the *QuickTest Professional Automation Object Model Reference* (**Help > QuickTest Professional Help > HP QuickTest Professional Advanced References > HP QuickTest Professional Automation Object Model**).

Resources Pane (Test Settings Dialog Box)

This pane enables you to associate specific files with your test, such as VBScript function libraries and data table files. You can also set the currently associated function library settings as the default settings for all new tests.



To access	Select File > Settings > Resources.
-----------	-------------------------------------

Important information	<ul style="list-style-type: none">➤ Object repositories are associated with individual action(s) in your test. You can associate an object repository with an action using the Action Properties dialog box (Edit > Action > Action Properties) and the Associate Repositories dialog box (Resources > Associate Repositories).➤ When running a test, QuickTest uses associated function libraries from Quality Center project folders only when you are connected to the corresponding Quality Center project. For details on working with Quality Center projects, see Chapter 52, "Quality Center Integration."
See also	➤ "Associated Function Libraries" on page 1015

User interface elements are described below:

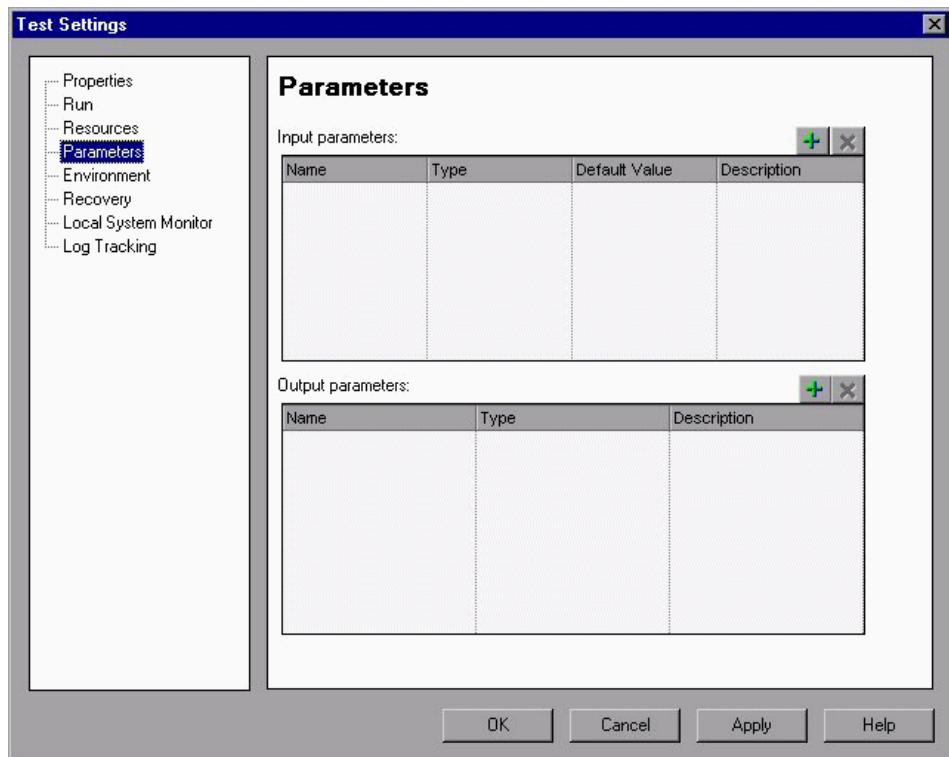
UI Element	Description
Associated function libraries	The list of function libraries associated with your test. You can add, delete, and prioritize the files. You can also set the default function libraries for new tests. For details, see "Associated Function Libraries" on page 1459.

UI Element	Description
	<p>Associates a function library with the test. You can enter the absolute or relative path and filename of the function library, or use the browse button to locate the required file. If the function library contains syntax errors, a message opens stating that your test will fail because of these syntax errors.</p> <p>The function library can be located in the file system or in a Quality Center project folder. For details on associating a function library stored in Quality Center, see "Resources Pane (Test Settings Dialog Box)" on page 1475, below.</p> <p>Note:</p> <ul style="list-style-type: none"> ➤ If you are working with the Resources and Dependencies model with Quality Center 10.00 or HP ALM, specify an absolute Quality Center path. For details, see "Relative Paths and Quality Center" on page 1656. ➤ When you are connected to Quality Center and, QuickTest adds [QualityCenter], and displays a browse button so that you can locate the Quality Center path. ➤ When not connected to Quality Center, you can add a file located in a Quality Center project folder by holding the SHIFT key and clicking this button. QuickTest adds [QualityCenter], and you can enter the path. You can also type the entire Quality Center path manually. If you do, you must add a space after [QualityCenter]. For example: [QualityCenter] Subject\Tests.
	<p>Removes an associated function library from the list.</p>
	<p>Assigns a higher priority to the selected function library.</p>
	<p>Assigns a lower priority to the selected function library.</p>

UI Element	Description
Set as Default	<p>Sets the current list of function libraries as the default list to be associated with new tests.</p> <p>Note: The Set as Default option is available for tests only. It is enabled when the setting for this test is different from the default for all tests.</p> <p>Caution: If the default function library is moved or renamed, QuickTest will not be able to locate it. The function library will be displayed in the Missing Resources pane when new tests are created. For information on resolving missing resources, see Chapter , "Missing Resources Pane."</p>
Check Syntax	<p>Verifies whether any of the associated function libraries contain syntax errors that will prevent the test from running properly. Click the Check Syntax button to check the files for syntax errors before finalizing the test. If any syntax errors are found, the Information pane opens listing the files containing syntax errors. Otherwise, an information box opens confirming that the syntax in all of the function libraries is valid.</p> <p>Note:</p> <ul style="list-style-type: none"> ▶ QuickTest checks only the associated function libraries that can be accessed. For example, if an associated function library is stored in a Quality Center project to which you are not currently connected, its syntax will not be checked. ▶ You cannot instruct QuickTest to check syntax for function libraries that are loaded dynamically during a run.
Data Table	<p>The location of the data table to be used in your test:</p> <ul style="list-style-type: none"> ▶ Default location (under test directory). Instructs QuickTest to use data stored in the default data table location under the test folder. ▶ Other location. Instructs QuickTest to use data stored in the specified data table location. The data table can be any Microsoft Excel (.xls) file. <p>For details on data tables, see "Data Table Overview" on page 1326.</p> <p>Note: You can specify Microsoft Excel files stored in Quality Center as data tables. For details, "How to Manage Data Tables in Your Test" on page 1336.</p>

Parameters Pane (Test Settings Dialog Box)

This pane enables you to define input parameters that pass values into your test and output parameters that pass values from your test to external sources. You can also use the Parameters pane to modify or delete existing test parameters.



To access	Select File > Settings > Parameters .
-----------	--

Important information	<ul style="list-style-type: none"> ➤ You can edit an existing parameter by selecting it in the appropriate list and modifying its details. ➤ If, during a test run, a value is not supplied by QuickTest or Quality Center for one or more input parameters, QuickTest uses the default value for the parameter. ➤ You can directly access test parameters only when parameterizing the value of a top-level action input parameter or when specifying the storage location for a top-level output parameter. To use values supplied for test parameters in steps within an action, you must pass the test parameter to the action containing the step. For details, see "Parameters Tab (Action Properties Dialog Box)" on page 563. ➤ Test parameters are similar to Action parameters. For information on Action parameters, see "Parameters Tab (Action Properties Dialog Box)" on page 563.
See also	For information on using parameter values in steps, see "Parameterizing Values" on page 723.

This pane contains the following key areas:

- "General User Interface Elements" on page 1480
- "Input Parameters Area" on page 1481
- "Output Parameters Area" on page 1482

General User Interface Elements

UI Elements	Description
	<p>Adds a row to the table, enabling you to specify a new parameter.</p> <p>You define test parameters in the same way you define action parameters. For information on defining parameters and parameter types, see "Parameters Tab (Action Properties Dialog Box)" on page 563.</p>
	Removes the selected parameter from the test.

Input Parameters Area

This area displays the test parameters for which values can be received from the source that runs or calls this test.

UI Elements	Description
Name	The parameter name (case-sensitive).
Type	<p>The parameter type. The following types are available:</p> <ul style="list-style-type: none"> ► String ► Boolean ► Date ► Number ► Password ► Any
Default Value	<p>The default value for the parameter.</p> <p>If you do not specify a default value, QuickTest assigns a default value as follows:</p> <ul style="list-style-type: none"> ► String: Empty string ► Boolean: True ► Date: The current date ► Number: 0 ► Password: Empty string ► Any: Empty string <p>When a test runs, the actual values used for parameters are generally those sent by the application (QuickTest or Quality Center) calling the test, as described below:</p> <ul style="list-style-type: none"> ► QuickTest. Input Parameters tab of the Run dialog box. For details, see "How to Run Your Test" on page 1067. ► Quality Center. Test Lab module. For details, see the Quality Center or HP ALM user guide.
Description	A meaningful description for the parameter, for example, the purpose of the parameter.

Output Parameters Area

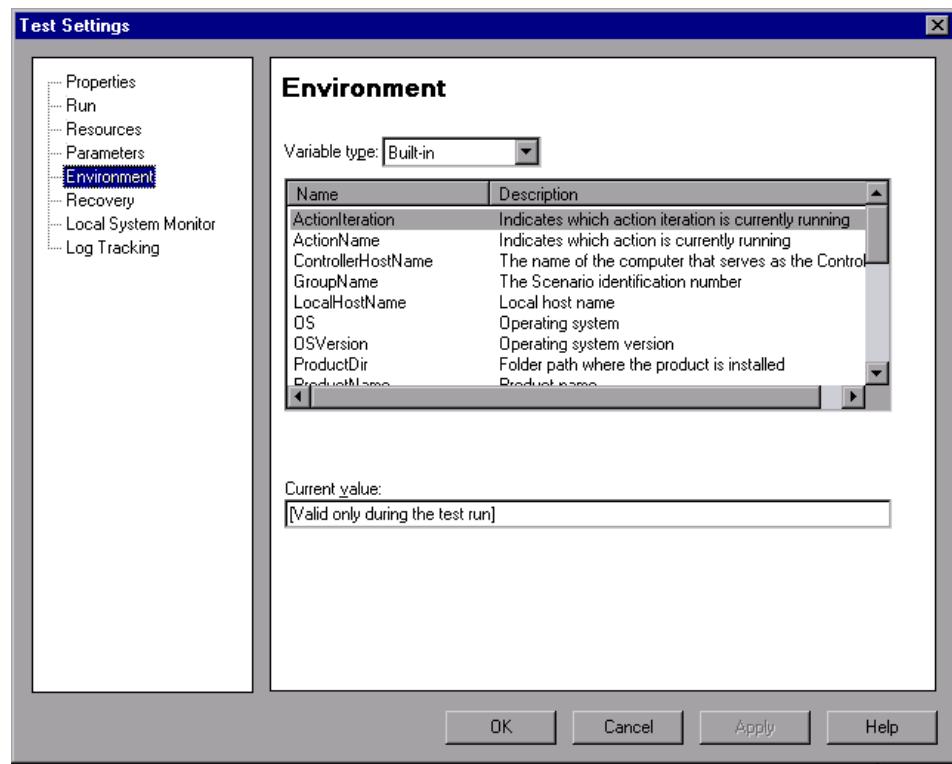
Lists the parameters that the test can pass to the source that runs or calls it.

UI Elements	Description
Name	The parameter name (case-sensitive).
Type	<p>The parameter type. The following types are available:</p> <ul style="list-style-type: none">➤ String➤ Boolean➤ Date➤ Number➤ Password➤ Any
Description	A meaningful description for the parameter, for example, the purpose of the parameter.

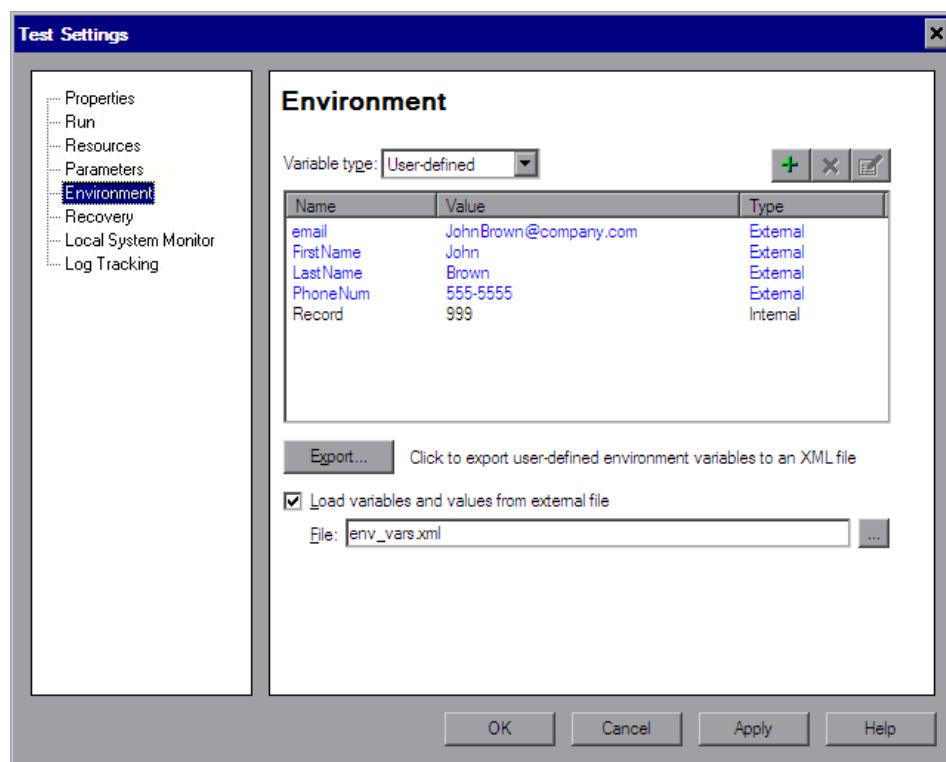
Environment Pane (Test Settings Dialog Box)

This pane displays existing built-in and user-defined environment variables. It also enables you to add, modify, or delete internal user-defined environment variables, export the defined variables to an external .XML file, and retrieve them from a file.

Built-in Environment Variables



User-Defined Environment Variables



To access	Select File > Settings > Environment .
Important information	<ul style="list-style-type: none"> ► Variables from an external environment variables file are displayed in blue. Internal environment variables are displayed in black. ► When you specify a path to a resource in the file system or in Quality Center 9.2, 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).
See also	"Environment Variable Parameters" on page 734

This pane includes the following key areas:

- "Variable Type Selection Area" on page 1485
- "Built-in Environment Variables Area" on page 1485
- "User-Defined Environment Variables Area" on page 1486

Variable Type Selection Area

UI Element	Description
Variable type	Enables you to select either a Built-in environment variable or a User-defined environment variable to define.

Built-in Environment Variables Area

Displays the built-in environment variables defined by QuickTest and their current values.

This area provides the following information:

UI Element	Description
Name	The name of each built-in environment variable.
Description	A short description of each built-in environment variable.
Current value	The current value of the selected environment variable.

User-Defined Environment Variables Area

Displays both internal and external user-defined environment variables and their current values.

This area provides the following information:

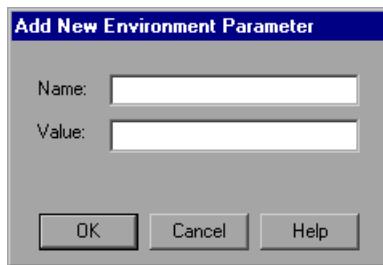
UI Element	Description
Name	The name of each user-defined variable.
Value	The value assigned to each user-defined variable.
Type	The type of each user-defined variable: Internal or External . Internal environment variables are available only to the test in which they are defined.
	Enables you to define a new internal environment variable and add it to the list. For details, see "Add New/View/Edit Environment Parameter Dialog Box", below.
	Deletes a selected internal environment variable from the list. Note: After you confirm the deletion of the environment variable, you cannot retrieve it, even if you click Cancel in the Test Settings dialog box.

UI Element	Description
	Enables you to edit the value of a selected internal environment variable or to view the properties of a selected external environment variable. For details, see "Environment Pane (Test Settings Dialog Box)" on page 1483.
Export	Exports your user-defined environment variables to an external .XML file for use with other tests. You can then use the exported environment variable file with any test by loading them from the file as external user-defined environment variables. For details, see "Save <Resource> Dialog Box" on page 417. <p>Note:</p> <ul style="list-style-type: none"> ▶ If the file is saved to the file system, its values are loaded each time the test runs. If the file is saved to a Quality Center project, its values are loaded when the test is first loaded. ▶ If the values are changed after the test is loaded, the new values will not be used by QuickTest, until the next time the test is loaded. ▶ When you specify a path to a resource in the file system or in Quality Center 9.2, 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 you are working with the Resources and Dependencies model with Quality Center 10.00 or HP ALM, you should specify an absolute Quality Center path. For details, see "Relative Paths and Quality Center" on page 1656.
Load variables and values from external file	Instructs QuickTest to load the variables saved in the .XML file that you specify for use with your test.
File	The environment variable .XML file to load and use when your test runs. Click the Browse button to open the Open XML File dialog box. For details, see "Open <Resource> Dialog Box" on page 406.

Add New/View/Edit Environment Parameter Dialog Box

This dialog box enables you to add, edit and internal user-defined environment variables from the Environment pane of the Test Settings dialog box. For external user-defined environment variables, you can view the variable details in read-only format.

The following image is an example of the dialog box when adding a new environment parameter. The actual name of the dialog box may differ depending on the operation you are performing.



To access	In the user-defined environment variable area of the Environment Pane (Test Settings Dialog Box), do one of the following: ► To add: Click the New  button. ► To view, modify, or copy: Select the environment variable and click the View/Edit Environment Variable button  .
Important information	Internal environment variables are available only to the test in which they are defined.

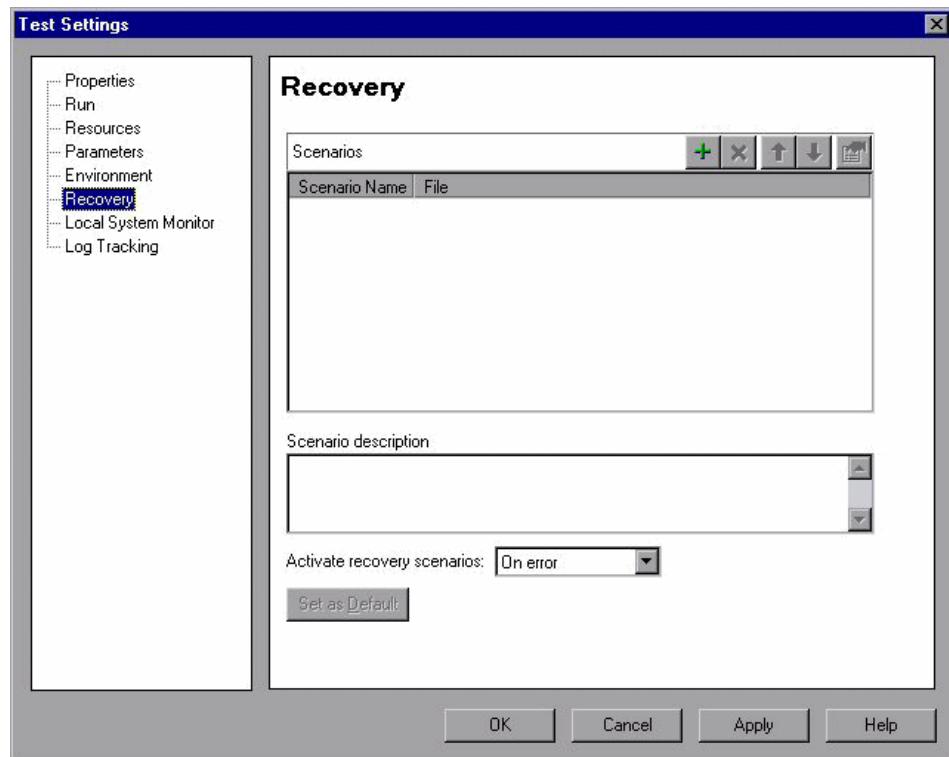
User interface elements are described below:

UI Elements	Description
Name	The name of the variable.
Value	The value of the variable. Tip: To copy the value of the variable to the Clipboard, select the value text, right-click, and select Copy .
	View/Edit Complex Value. Enables you to view the contents of the value in a multi-line edit box. Available only when the value cannot be displayed entirely in the Value box.

Recovery Pane (Test Settings Dialog Box)

This pane displays a list of all recovery scenarios associated with the current test. Recovery scenario settings enable you to specify how a test recovers from unexpected events and errors during a run session.

This pane also enables you to associate additional recovery scenarios with the test, create recovery scenarios, remove scenarios from the test, change the order in which they are applied to the run session, and specify the default list of scenarios to associate with all new tests.



To access	Select File > Settings > Recovery .
Important information	<ul style="list-style-type: none"> ➤ The default recovery scenarios provided with QuickTest are installed in your QuickTest installation folder. The paths specifying the default recovery scenarios in the Recovery pane use an environment variable (%ProductDir%) in the file path. This enables QuickTest to locate these recovery scenarios when tests associated with them are run on different computers or by different HP products. Do not modify the file paths of these default recovery scenarios or attempt to use the environment variable for any other purpose. ➤ You can also associate, remove, enable, disable, prioritize, and view the properties of the recovery scenarios associated with your test in the Resources pane. For details, see "Resources Pane" on page 1385.
See also	"Recovery Scenarios" on page 1523

This pane includes the following key elements:

- "Scenarios Area" on page 1491
- "Scenario Description and General Options Area" on page 1493
- "Scenario Type Icons" on page 1494

Scenarios Area

UI Element	Description
	Opens the Add Recovery Scenario dialog box, which enables you to associate one or more recovery scenarios with the test. For details, see "Add Recovery Scenario Dialog Box" on page 1532.
	Removes the selected recovery scenario from the test.
	Moves the selected scenario up in the list, giving it a higher priority.

UI Element	Description
	Moves the selected scenario down in the list, giving it a lower priority.
	Displays summary properties for the selected recovery scenario in read-only format. For details, see "Recovery Scenario Properties Dialog Box" on page 1537.
Scenario Name	The name of each recovery scenario associated with your test. You can add, delete, and prioritize the scenarios in the list.
File	<p>The file path for each recovery scenario associated with your test. You can edit the recovery scenario file path by clicking the path once to highlight it, and then clicking it again to enter edit mode. For details on scenario type icons, see "Scenario Type Icons" on page 1494.</p> <p>For example, you may want to modify an absolute file path to be a relative file path.</p> <p>If you modify a recovery scenario file path, ensure that the recovery scenario exists in the new path location before running your test.</p> <p>Note:</p> <ul style="list-style-type: none"> ➤ When working with tests, if your recovery files are stored in the file system and you want other users or HP products to be able to run this test on other computers, you should set the recovery file path as a relative path. (Click the path once to highlight it, and then click it again to enter edit mode.) Any users who want to run this test should then specify the drive letter and folder in which QuickTest should search for the relative path in the Folders pane of the Options dialog box (Tools > Options > Folders node). For details, see "Folders Pane (Options Dialog Box)" on page 1431 and "Relative Paths in QuickTest" on page 391. ➤ If you are working with the Resources and Dependencies model with Quality Center 10.00 or HP ALM, you should store your recovery files in the Quality Center Test Resources module and specify an absolute Quality Center path in the Folders pane. For details, see "Relative Paths and Quality Center" on page 1656.

Scenario Description and General Options Area

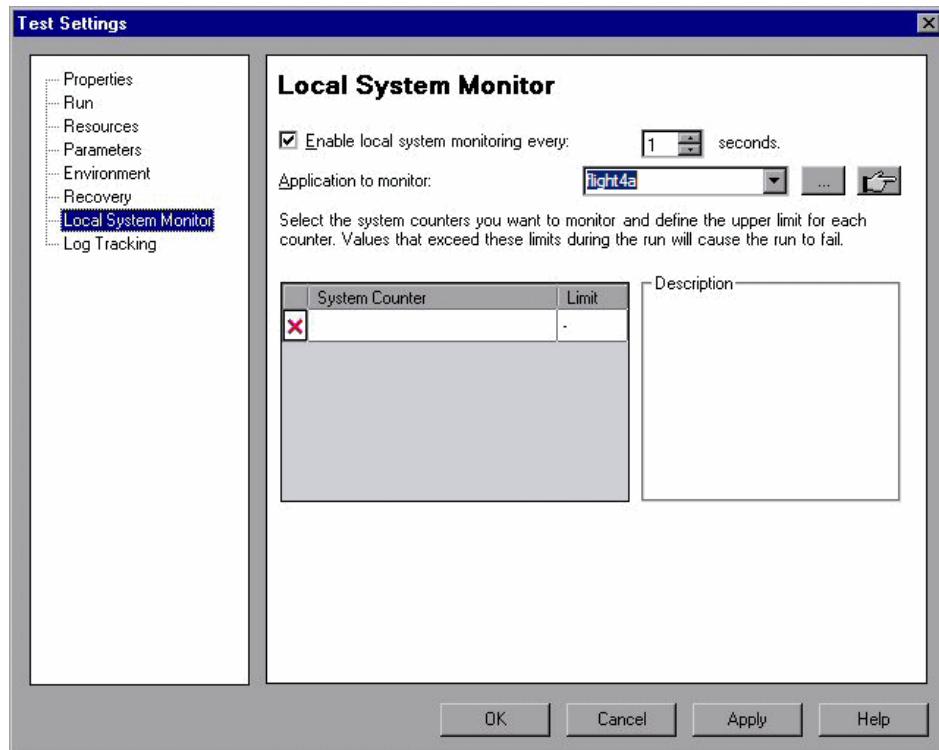
UI Elements	Description
Scenario description	Displays the textual description of the scenario selected in the Scenarios box. You can select or clear the check box next to each scenario to enable or disable it for the current test.
Activate recovery scenarios	<p>Indicates how often QuickTest should activate the recovery mechanism:</p> <ul style="list-style-type: none"> ▶ On every step. The recovery mechanism is activated after every step. ▶ On error. The recovery mechanism is activated only after steps that return an error return value. ▶ Never. The recovery mechanism is disabled. <p>Note: Choosing On every step may result in slower performance during the run session.</p>
Set as Default	<p>Sets the current list of recovery scenario files as the default list to be associated with new tests.</p> <p>Note: The Set as Default option is available for tests only. It is enabled when the setting for this test is different from the default for all tests.</p> <p>Caution: If the file containing the recovery scenarios is moved or renamed, QuickTest will not be able to locate it. The recovery scenario file will be displayed in the Missing Resources pane when new actions or tests are created. For information on resolving missing resources, see Chapter , "Missing Resources Pane."</p>

Scenario Type Icons

Icon	Description
	Indicates that the recovery scenario is triggered by a specific pop-up window in an open application during the run session.
	Indicates that the recovery scenario is triggered when the property values of an object in an application match specified values.
	Indicates that the recovery scenario is triggered when a step in the test does not run successfully.
	Indicates that the recovery scenario is triggered when a specified application fails during the run session.
	Indicates that the recovery scenario is no longer available for the test—possibly because the recovery file has been renamed or moved, or can no longer be accessed by QuickTest. When an associated recovery file is not available during a run session, a message is displayed in the run results.

Local System Monitor Pane (Test Settings Dialog Box)

This pane enables you to activate system monitoring, and define the system counters you want to track during a run session. These counters enable you to monitor the resources used by your application.



To access	Use one of the following: <ul style="list-style-type: none">► Select File > Settings > Local System Monitor.► Click the Settings  toolbar button.
-----------	--

Important information	<ul style="list-style-type: none"> ➤ The Local System Monitor data that is captured during a run session is displayed in the Run Results Viewer. For details, see "System Monitor Pane (Run Results Viewer)" on page 1137. ➤ If more than one process with the same name runs during a run session, and you monitor a counter for that process (for example, you select to monitor a counter for the <code>iexplorer.exe</code> process, and more than one Internet Explorer browser is open on your desktop during the run session), the counter is sampled from the application that contains at least one test object from the test. If more than one application meets this criterion, only one application is monitored.
See also	"Local System Monitor" on page 1460

User interface elements are described below:

UI Element	Description
Enable local system monitoring every: _____ seconds	<p>The frequency in seconds, by which the system counters for this application will be checked.</p> <p>Use the up and down arrows or enter a value in the edit box to change the number of seconds.</p> <p>Minimum value: One second.</p>

UI Element	Description
Application to monitor	<p>The application whose system counters you want to monitor. You can define the application in any of the following ways:</p> <ul style="list-style-type: none"> ➤ Enter the name of the application's executable file (without file extension) in the edit box. ➤ Click the down arrow in the edit box for a list of applications previously run in QuickTest, currently running applications, and applications currently specified in the Windows-based Applications tab of the Record and Run Settings dialog box. ➤ Click the browse button  and browse to the application's executable file. ➤ Make sure that your application is currently running. Then click the pointing hand  and point to the application on your desktop. <p>Note: Sometimes a process is used only as a launcher that creates another process that provides the application functionality. Make sure to select the executable file that actually provides the application functionality.</p>
System Counter	<p>The system counter you want to track for the selected application. Click inside a cell and then click the down arrow. Select the counter from the list. Click the expand button  when displayed to show more counters.</p> <p>You can monitor the process counters, which are accessible through the performance console (select Start > Run > and then enter Perfmon). For information these process counters, see the performance console's Help.</p>
Limit	<p>The upper limit of the counter selected in the System Counter column. If the selected counter exceeds this value during the run session, the run fails.</p> <p>The Limit value is optional. If you do not supply a value, the counter is tracked and the results are displayed in the Run Results Viewer.</p>
	Removes the system counter definition from your test.
Description	The description of the counter selected in the System Counter column, as provided by the performance console application.

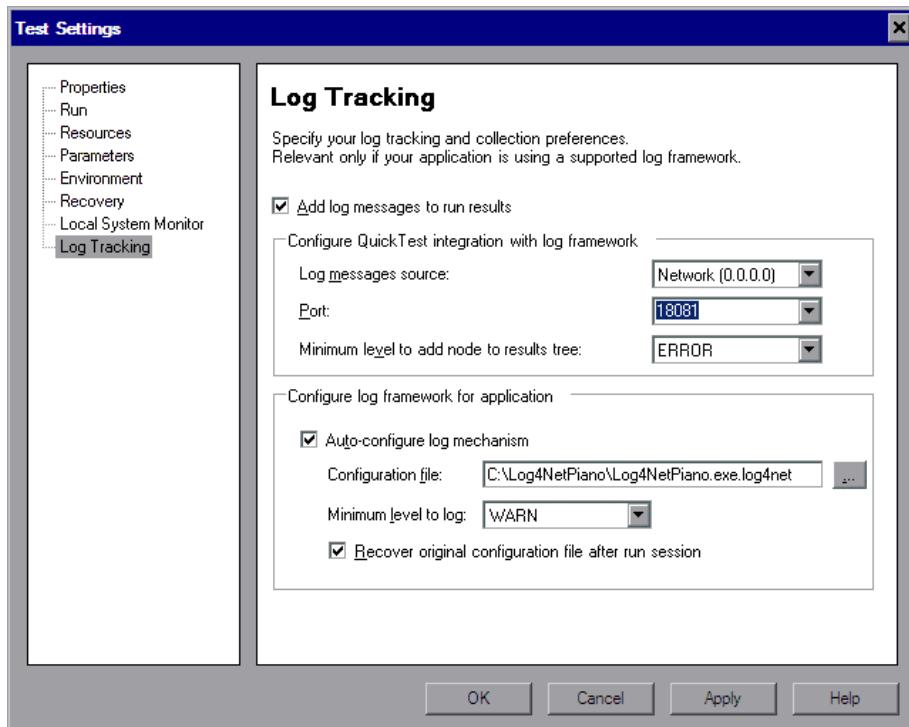
Log Tracking Pane (Test Settings Dialog Box)

This pane enables you to configure your log tracking and collection preferences.

This pane is divided into two sections:

- The settings in the top half are QuickTest-specific, enabling it to receive log messages during a run session.
- The settings in the bottom half are specific to the log configuration file used by your application.

When configured, these settings are used during a run session if your Windows-based application uses a Java or .NET log framework that includes a UDP appender. The log messages generated by your application are displayed in the run results. You can use this information to detect unexpected behavior in your application.



To access	Select File > Settings > Log Tracking .
Important information	<p>Prerequisites:</p> <ul style="list-style-type: none"> ➤ Make sure that the firewall does not block the UDP port. ➤ The log framework must use an XML-based configuration file. (This enables you to configure it to send log messages to QuickTest.) ➤ QuickTest uses timestamps to associate log messages with the relevant test step. Therefore, if your logging application is located on a remote computer, make sure that the system clocks on your application's computer (where QuickTest is installed) and the remote computer are synchronized. ➤ Some applications may need to be restarted after modifying the log configuration file. <p>Limitations:</p> <ul style="list-style-type: none"> ➤ These settings cannot be modified during a run session. ➤ If external events affect your application while it is being tested, messages about these events may also be sent to the run results.
Relevant tasks	"How to Manually Configure Log Tracking Settings" on page 1462
See also	<ul style="list-style-type: none"> ➤ "Log Tracking" on page 1461 ➤ "Log Tracking Pane (Run Results Viewer)" on page 1132

User interface elements are described below:

UI Elements	Description
Add log messages to run results	<p>Enables QuickTest to receive log messages from the log framework used by your application and to include these messages in the run results.</p> <p>In the Run Results viewer, these messages are displayed in the Log Tracking Results pane. You can click a log message to locate the step in the run results tree at which the event that preceded/triggered the message occurred (according to its timestamp). For details, see "Log Tracking Pane (Run Results Viewer)" on page 1132.</p>

UI Elements	Description
Log message source	<ul style="list-style-type: none"> ➤ Network (0.0.0.0). (Default) Enables QuickTest to receive log messages from the log framework wherever it is located—either on a remote computer on the network or on the local computer. ➤ Local (127.0.0.1). Enables QuickTest to receive log messages only from the log framework on the local computer.
Port	<p>The port that QuickTest listens to on the computer on which the log framework runs. You can select any UDP port that is not in use.</p>
Minimum level to add node to results tree	<p>The minimum log message level for which a node is added to the run results.</p> <p>QuickTest inserts a node in the run results tree directly following each step that triggers a log message that matches or exceeds the value you select in this option. You can view details for these message in the Result Details pane.</p> <p>You can also view the generated messages in the Results.xml file in the <Result Name>\Report folder.</p> <p>Possible values in order of severity:</p> <ul style="list-style-type: none"> ➤ TRACE (maps to micDone) ➤ DEBUG (maps to micDone) ➤ INFO (maps to micPass) ➤ WARN (maps to micWarning) ➤ ERROR (maps to micFail) ➤ FATAL (maps to micFail) <p>Default: ERROR</p>

UI Elements	Description
	<p>Auto-configure log mechanism</p> <p>Instructs QuickTest to configure the log framework (via its log configuration file) at the beginning of every run session according to the settings in this pane.</p> <p>If you clear this check box, QuickTest receives log messages only if you modify the log configuration file accordingly. For details, see "How to Manually Configure Log Tracking Settings" on page 1462.</p> <p>When using the Auto-configure log mechanism option, make sure that:</p> <ul style="list-style-type: none"> ▶ All logging prerequisites are met, for example, you may need to set registry values that are specific to your application. ▶ The log framework's configuration file is writable and is in a location that QuickTest can access. ▶ Your application's log framework is configured to use the same file you specified in the Configuration file edit box. ▶ Your application's log framework can monitor changes in the configuration file. If not, then you must always start your application after the run session begins so that QuickTest can modify the configuration file. For details on starting your application, see the SystemUtil.Run topic in the <i>HP QuickTest Professional Object Model Reference</i>. You can also use any standard VB Script command that starts an application. <p>Note: If your application's log framework monitors your configuration file infrequently (for example, once per minute), start your application immediately after the run session begins. This enables QuickTest to modify the configuration file in time to receive the log messages generated during the run session.</p>

UI Elements	Description
Configuration file	<p>The root path of the configuration file used by the log mechanism. The configuration file can be stored in any accessible location.</p> <p>Possible configuration file types:</p> <ul style="list-style-type: none"> ➤ *.XML (Java and .NET) ➤ *.CONFIG (.NET) ➤ *.LOG4NET (.NET) <p>Note: QuickTest can update only one file per run session, so if there are multiple configuration files (for example, if the test is validating more than one application - each with its own configuration file), QuickTest modifies only the file specified in this text box. If you need to configure multiple files, do so manually. QuickTest can receive log messages from multiple applications, but can auto-configure only one of them.</p>
Minimum level to log	<p>The minimum log message level that QuickTest receives from the log framework and sends to the run results. These log messages are displayed in the Log Tracking Pane (Run Results Viewer) and can also be viewed in the LogMessage.xml file in the <Result Name>\Report folder.</p> <p>Possible values in order of severity:</p> <ul style="list-style-type: none"> ➤ TRACE (maps to micDone) ➤ DEBUG (maps to micDone) ➤ INFO (maps to micPass) ➤ WARN (maps to micWarning) ➤ ERROR (maps to micFail) ➤ FATAL (maps to micFail) <p>Default: WARN</p>
Recover original configuration file after run session	<p>Restores the configuration file that existed prior to the beginning of the run session (instead of keeping the configuration file that QuickTest modified at the beginning of the run session).</p>

47

Setting Testing Options During the Run Session

This chapter includes:

Concepts

- Setting Testing Options During the Run Session - Overview on page 1504

Tasks

- How to Set Testing Options During a Run Session on page 1506

Concepts

Setting Testing Options During the Run Session - Overview

You can use the **Setting** object to control how QuickTest run tests by setting and retrieving testing options during a run session.

QuickTest testing options affect how you work with tests. For example, you can set the maximum time that QuickTest allows when loading a Web page, before determining that the URL address cannot be found.

You can set and retrieve the values of testing options during a run session using the **Setting** object in the Expert View. For more information on working in the Expert View, see Chapter 27, "Working in the Expert View and Function Library Windows."

By retrieving and setting testing options using the **Setting** object, you can control how QuickTest runs a test.

You can also set many testing options using the Options dialog box (global testing options) and the Test Settings dialog box (test-specific settings). For more information, see Chapter 45, "Global Testing Options" and Chapter 46, "Individual Test Settings."

This chapter describes some of the QuickTest testing options that can be used with the **Setting** object from within a test script. For detailed information on all the available methods and properties for the **Setting** object, see the **Utility** section of the *HP QuickTest Professional Object Model Reference*.

Note: You can also control QuickTest options as well as most other QuickTest operations using automation scripts. For more information, see "QuickTest Automation Scripts" on page 1589 or the *QuickTest Professional Automation Object Model Reference* (**Help > QuickTest Professional Help > HP QuickTest Professional Advanced References > HP QuickTest Professional Automation Object Model**).

Tasks

How to Set Testing Options During a Run Session

The following steps describe how you can set and retrieve the values of testing options during a run session using the **Setting** object in the Expert view.

- "Set the value of a testing option from within the test script" on page 1506
- "Retrieve Testing Options" on page 1507
- "Control the Test Run" on page 1508
- "Add and Remove Run-Time Settings" on page 1508

Set the value of a testing option from within the test script

To set the option, use the following syntax:

Setting (testing_option) = new_value

Some options are global and others affect only the current test. After you use a Setting object to set a testing option, the setting remains in effect until it is changed again or until the end of your current QuickTest session. You can also use the Setting object to change a setting for a specific part of a specific test. For more information see "Control the Test Run" on page 1508.

Some of the testing options that you can set using the Setting object are also available in the Options dialog box (global options) or the Test Settings dialog box (test specific settings). When you use the Setting object to set these options, the change is reflected in the relevant dialog box. Other test settings can be accessed using only one method, either the relevant dialog box or the Setting object. For more details, see "Setting Testing Options During the Run Session - Overview" on page 1504.

Example:

If you run the following statement with the Web Add-in loaded:

```
Setting("AutomaticLinkRun")=1
```

QuickTest disables automatically created checkpoints in the test. The setting remains in effect during your current QuickTest session until it is changed again, either with another Setting statement, or by clearing the **Ignore automatic checkpoints while running tests or components** check box in the Web Advanced pane (Select **Tools > Options > Web > Advanced** node).

If you run the following statement:

```
Setting("WebTimeOut")=50000
```

QuickTest automatically changes the amount of time it waits for a Web page to load before running a test step to 50000 milliseconds. The setting remains in effect during your current QuickTest session until it is changed again, either with another Setting statement, or by setting the **Browser Navigation Timeout** option in the Web pane of the Test Settings dialog box.

Note: Although the changes you make using the Setting object are reflected in the Options and Test Settings dialog boxes, these changes are not saved when you close QuickTest, unless you make other changes in the same dialog box manually and click **Apply** or **OK** (which saves all current settings in that dialog box).

Retrieve Testing Options

You can use the Setting object to retrieve the current value of a testing option.

To store the value in a variable, use the syntax:

```
new_var = Setting ( testing_option )
```

To display the value in a message box, use the syntax:

MsgBox (Setting (*testing_option*))

For example:

```
LinkCheckSet = Setting("AutomaticLinkRun")
```

assigns the current value of the AutomaticLinkRun setting to the user-defined variable LinkCheckSet.

Control the Test Run

You can use the retrieve and set capabilities of the Setting object together to control a run session without changing global settings. For example, if you want to change the DefaultTimeOut testing option to 5 seconds for objects on one Web page only, insert the following statement after the Web page opens in your test script:

```
'Keep the original value of the DefaultTimeOut testing option  
old_delay = Setting ("DefaultTimeOut")
```

```
'Set temporary value for the DefaultTimeOut testing option  
Setting("DefaultTimeOut")= 5000
```

To return the DefaultTimeOut testing option to its original value at the end of the Web page, insert the following statement immediately before linking to the next page in the script:

```
'Change the DefaultTimeOut testing option back to its original value.  
Setting("DefaultTimeOut")=old_delay
```

Add and Remove Run-Time Settings

You can add, modify, and remove custom run-time settings. These settings are applicable during the run session only.

To add a new run-time setting, use the syntax:

Setting.Add "*testing_option*", "*value*"

For example, you could create a setting that indicates the name of the current tester and displays the name in a message box.

```
Setting.Add "Tester Name", "Mark Train"  
MsgBox Setting("Tester Name")
```

Tip: When using a `Setting.Add` statement, an error occurs if you try to add an existing setting option. To avoid this error you should use a `Setting.Exists` statement first. For more details about all the `Setting` methods, see the *HP QuickTest Professional Object Model Reference*.

To modify a run-time setting that has already been initialized, use the same syntax you use for setting any standard setting option:

Setting (*testing_option*) = *new_value*

For example:

```
Setting("Tester Name")="Alice Wonderlin"
```

To delete a custom run-time setting, use the following syntax:

Setting.Remove (*testing_option*)

For example:

```
Setting.Remove ("Tester Name")
```

Tip: When using a `Setting.Remove` statement, an error occurs if you try to remove a setting option that does not exist. To avoid this error you should use a `Setting.Exists` statement first. For more details about all the `Setting` methods, see the *HP QuickTest Professional Object Model Reference*.

Part X

Working with Advanced Testing Features

48

Virtual Objects

This chapter includes:

Concepts

- Virtual Objects - Overview on page 1514
- How Virtual Objects are Defined and Recognized on page 1515

Tasks

- How to Define Virtual Objects for Unsupported Objects in Your Test on page 1516

Reference

- Virtual Object Manager Dialog Box on page 1518
- Virtual Object Wizard on page 1519

Concepts

Virtual Objects - Overview

Your application may contain objects that behave like standard objects but are not recognized by QuickTest. You can define these objects as virtual objects and map them to standard classes, such as a button or a check box. QuickTest emulates the user's action on the virtual object during the run session. In the run results, the virtual object is displayed as though it is a standard class object.

For example, suppose you want to test a Web page containing a bitmap that the user clicks. The bitmap contains several different hyperlink areas, and each area opens a different destination page. When you create the test, the Web site matches the coordinates of the click on the bitmap and opens the destination page.

To enable QuickTest to click at the required coordinates during a run session, you can define a virtual object for an area of the bitmap, which includes those coordinates, and map it to the button class. When you run the test, QuickTest clicks the bitmap in the area defined as a virtual object so that the Web site opens the correct destination page.

Virtual object collections are groups of virtual objects that are stored in the Virtual Object Manager under a descriptive name. For more information, see "Virtual Object Manager Dialog Box" on page 1518.

The virtual object collections displayed in the Virtual Object Manager are stored on your computer and not with the tests that contain virtual object steps. This means that if you use a virtual object in a test step, the object will be recognized during the run session only if it is run on a computer containing the appropriate virtual object definition. To copy your virtual object collection definitions to another computer, copy the contents of your <QuickTest installation folder>\dat\VoTemplate folder (or individual .vot collection files within this folder) to the same folder on the destination computer.

Note: QuickTest does not support virtual objects for analog or low-level recording. For more information on low-level recording, see "Creating Tests" on page 1790.



How Virtual Objects are Defined and Recognized

QuickTest identifies a virtual object according to its boundaries. Marking an object's boundaries specifies its size and position on a Web page or application window. When you assign a test object as the parent of your virtual object, you specify that the coordinates of the virtual object boundaries are relative to that parent object. When you record a test, QuickTest recognizes the virtual object within the parent object and adds it as a test object in the object repository so that QuickTest can identify the object during the run session. QuickTest also recognizes the virtual object as a test object when you add it manually to the object repository.

To perform an operation in the Active Screen on a marked virtual object, you must first record it, so that its properties are saved in the test object description in the object repository. If you perform an operation in the Active Screen on a virtual object that has not yet been recorded, QuickTest treats it as a standard object.

You can use virtual objects only when recording and running a test. You cannot insert any type of checkpoint on a virtual object, or use the Object Spy to view its properties.

You can enable and disable recognition of virtual objects during recording, in the General pane of the Options dialog box.

During a run session, make sure that the application window is the same size and in the same location as it was during recording, otherwise the coordinates of the virtual object relative to its parent object may be different, and this may affect the success of the run session.

Tasks

How to Define Virtual Objects for Unsupported Objects in Your Test

This task describes how to define virtual objects for your test, for objects that are not normally recognized by QuickTest.

This task includes the following steps:

- "Display the object containing the area you want to define as a virtual object." on page 1516
- "Use the Virtual Object wizard to define your virtual object." on page 1517

1 Display the object containing the area you want to define as a virtual object.

With QuickTest open (but not in record mode), open your application and display the object containing the area you want to define as a virtual object.

You can define virtual objects only for objects on which QuickTest records **Click** or **DblClick** methods. Otherwise, the virtual object is ignored. For example, if you define a virtual object over the WinList object, the Select operation is recorded, and the virtual object is ignored. QuickTest does not support virtual objects for analog or low-level recording. For more information on low-level recording, see "Frequently Asked Questions" on page 1789.

2 Use the Virtual Object wizard to define your virtual object.

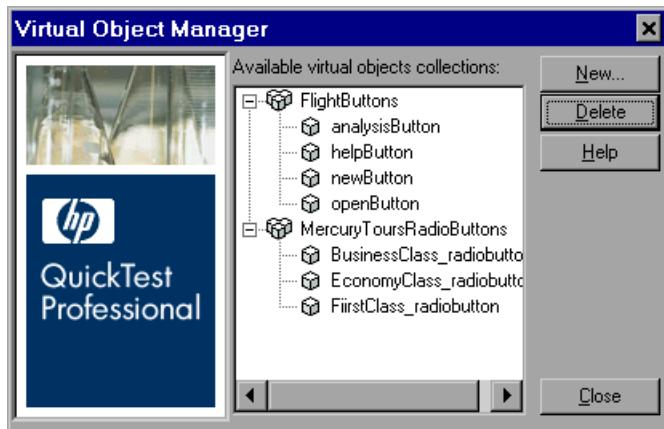
Open the Virtual Object wizard (Tools > Virtual Objects > New Virtual Object). The Virtual Object wizard includes the following pages:

- ▶ Map to a Standard Class Page (Virtual Object Wizard), described on page 1520
- ▶ Mark Virtual Object Page (Virtual Object Wizard), described on page 1520
- ▶ Object Configuration Page (Virtual Object Wizard), described on page 1521
- ▶ Save Virtual Object Page (Virtual Object Wizard), described on page 1522

Reference

Virtual Object Manager Dialog Box

This dialog box enables you to view and manage the virtual object collections defined on your computer. From the Virtual Object Manager, you can define and delete virtual objects and collections.



To access	Select Tools > Virtual Objects > Virtual Object Manager .
Important information	"How Virtual Objects are Defined and Recognized" on page 1515
Relevant tasks	"How to Define Virtual Objects for Unsupported Objects in Your Test" on page 1516

User interface elements are described below:

UI Elements	Description
Available virtual object collections	Displays the virtual object collections defined on your computer and the virtual objects contained in each collection. Click the + and - signs next to a collection to view or hide the virtual objects defined in that collection.
New	Opens the Virtual Object Wizard, which guides you through the process of defining a new virtual object for a new or existing collection.
Delete	Deletes the selected virtual object or virtual object collection.

Virtual Object Wizard

This wizard enables you to define a virtual object by:

- Mapping it to a standard class
- Marking its boundaries
- Assigning it a parent object
- Specifying its name and grouping objects into collections

To access	Use one of the following: ➤ Select Tools > Virtual Objects > New Virtual Object . ➤ Select Tools > Virtual Objects > Virtual Object Manager . From the Virtual Object Manager, click New .
Important information	"How Virtual Objects are Defined and Recognized" on page 1515
Relevant tasks	"How to Define Virtual Objects for Unsupported Objects in Your Test" on page 1516
Wizard map	This wizard contains: Welcome > Map to a Standard Class (page 1520) > Mark Virtual Object (page 1520) > Object Configuration (page 1521) > Save Virtual Object (page 1522)



Map to a Standard Class Page (Virtual Object Wizard)

This wizard page enables you to configure a standard class for the virtual object.

Wizard map	Virtual Object Wizard contains: Welcome > Map to a Standard Class > Mark Virtual Object > Object Configuration > Save Virtual Object
-------------------	--

User interface elements are described below:

UI Elements	Description
Class	Specify a standard object class from the list. <ul style="list-style-type: none"> ▶ For the list class, specify the number of rows in the virtual object. ▶ For the table class, select the number of rows and columns.



Mark Virtual Object Page (Virtual Object Wizard)

This wizard page enables you to configure the size and location of the virtual object.

Important information	<ul style="list-style-type: none"> ▶ Make sure that the virtual object does not overlap any other virtual object, as this may prevent QuickTest from identifying the virtual object during a run session. ▶ To record and run tests properly, you must ensure that the application window is the same size and in the same position as it was when you defined the virtual object.
Wizard map	Virtual Object Wizard contains: Welcome > Map to a Standard Class > Mark Virtual Object > Object Configuration > Save Virtual Object



Object Configuration Page (Virtual Object Wizard)

This wizard page enables you to configure an object as a parent of the virtual object.

Wizard map	Virtual Object Wizard contains: Welcome > Map to a Standard Class > Mark Virtual Object > Object Configuration > Save Virtual Object
-------------------	--

User interface elements are described below:

UI Elements	Description
Select the parent of the virtual object	Enables you to select an object in the tree as the parent object. The coordinates of the virtual object outline are relative to the parent object.
Selected parent	Indicates the name of the object selected as the parent object (read-only).
Identify object using	<ul style="list-style-type: none"> ▶ Entire parent hierarchy. Select this radio button to identify the virtual object in one occurrence only. QuickTest identifies the virtual object only if it has the exact parent hierarchy. For example, if the virtual object is defined using <code>Browser("A").Page("B").Image("C")</code>, QuickTest does not recognize it if the hierarchy changes to <code>Browser("X").Page("B").Image("C")</code>. ▶ Parent only. Select this radio button to identify all occurrences of the virtual object. QuickTest identifies the virtual object using its direct parent only, regardless of the entire parent hierarchy. For example, if the virtual object was defined using <code>Browser("A").Page("B").Image("C")</code>, QuickTest will recognize the virtual object even if the hierarchy changes to <code>Browser("X").Page("Y").Image("C")</code>.



Save Virtual Object Page (Virtual Object Wizard)

This wizard page enables you to configure a name and a collection for the virtual object. It also enables you to begin defining another virtual object.

Wizard map	Virtual Object Wizard contains: Welcome > Map to a Standard Class > Mark Virtual Object > Object Configuration > Save Virtual Object
-------------------	--

Recovery Scenarios

This chapter includes:

Concepts

- ▶ Recovery Scenarios Overview on page 1524

Tasks

- ▶ How to Create and Manage Recovery Scenarios on page 1528
- ▶ How to Manage Recovery Scenario Associations on page 1530

Reference

- ▶ Add Recovery Scenario Dialog Box on page 1532
- ▶ Recovery Scenario Manager Dialog Box on page 1534
- ▶ Recovery Scenario Properties Dialog Box on page 1537
- ▶ Recovery Scenario Wizard on page 1539

Troubleshooting and Limitations - Recovery Scenarios on page 1569

Concepts

Recovery Scenarios Overview

Unexpected events, errors, and application crashes during a run session can disrupt your run session and distort results. This is a problem particularly when tests run unattended—the test pauses until you perform the operation needed to recover. To handle situations such as these, QuickTest enables you to create recovery scenarios and associate them with specific tests. Recovery scenarios activate specific recovery operations when trigger events occur.

The Recovery Scenario Manager provides a wizard that guides you through the process of defining a recovery scenario, which includes a definition of an unexpected event and the operations necessary to recover the run session. For example, you can instruct QuickTest to detect a **Printer out of paper** message and recover the run session by clicking the **OK** button to close the message and continue the test.

A recovery scenario consists of the following:

- **Trigger Event.** The event that interrupts your run session. For example, a window that pops up on the screen, or a QuickTest run error.
- **Recovery Operations.** The operations to perform to enable QuickTest to continue running the test after the trigger event interrupts the run session. For example, clicking an **OK** button in a pop-up window, or restarting Microsoft Windows.
- **Post-Recovery Test Run Option.** The instructions on how QuickTest should proceed after the recovery operations have been performed, and from which point in the test QuickTest should continue, if at all. You may want to restart a test from the beginning, or skip a step entirely and continue with the next step in the test.

After you create recovery scenarios, you associate them with selected tests so that QuickTest will perform the appropriate scenarios during the run sessions if a trigger event occurs. You can prioritize the scenarios and set the order in which QuickTest applies the scenarios during the run session. You can also choose to disable specific scenarios, or all scenarios, that are associated with a test. You can also define which recovery scenarios will be used as the default scenarios for all new tests.

Note: You can associate, remove, enable, disable, prioritize, and view the properties of the recovery scenarios associated with your test in the Resources pane. For details, see "Resources Pane User Interface" on page 1391.

This section also includes:

- "When to Use Recovery Scenarios" on page 1525
- "Programmatically Controlling the Recovery Mechanism" on page 1527



When to Use Recovery Scenarios

Recovery scenarios are intended for use **only** with events that you cannot predict in advance, or for events that you cannot otherwise synchronize with a specific step in your test.

By default, recovery scenario operations are activated only after a step returns an error. This can potentially occur several steps after the step that originally caused the error. The alternative, checking for trigger events after every step, may slow performance. For this reason, it is best to handle predictable errors directly in your test.

If you can predict that a certain event may happen at a specific point in your test, it is highly recommended to handle that event directly within your test by adding steps such as If statements or optional steps, rather than depending on a recovery scenario.

Handling an event directly within your test enables you to handle errors more specifically than recovery scenarios, which by nature are designed to handle a more generic set of unpredictable events. It also enables you to control the timing of the corrective operation with minimal resource usage and maximum performance.

Examples

- If you know that an Overwrite File message box may open when a **Save** button is clicked during a run session, you can handle this event with an If statement that clicks **OK** if the message box opens or by adding an optional step that clicks **OK** in the message box.
- You could define a recovery scenario to handle printer errors. Then if a printer error occurs during a run session, the recovery scenario could instruct QuickTest to click the default button in the Printer Error message box.

You would use a recovery scenario in this example because you cannot handle this type of error directly in your test. This is because you cannot know at what point the network will return the printer error. Even if you try to handle this event by adding an If statement in your test immediately after a step that sends a file to the printer, your test may progress several steps before the network returns the actual printer error.



Programmatically Controlling the Recovery Mechanism

You can use the Recovery object to control the recovery mechanism programmatically during the run session. For example, you can enable or disable the entire recovery mechanism or specific recovery scenarios for certain parts of a run session, retrieve status information about specific recovery scenarios, and explicitly activate the recovery mechanism at a certain point in the run session.

By default, QuickTest checks for recovery triggers when an error is returned during the run session. You can use the Recovery object's Activate method to force QuickTest to check for triggers after a specific step in the run session. For example, suppose you know that an object property checkpoint will fail if certain processes are open when the checkpoint is performed. You want to be sure that the pass or fail of the checkpoint is not affected by these open processes, which may indicate a different problem with your application.

However, a failed checkpoint does not result in a run error. So by default, the recovery mechanism would not be activated by the object state. You can define a recovery scenario that looks for and closes specified open processes when an object's properties have a certain state. This state shows the object's property values as they would be if the problematic processes were open. You can instruct QuickTest to activate the recovery mechanism if the checkpoint fails so that QuickTest will check for and close any problematic open processes and then try to perform the checkpoint again. This ensures that when the checkpoint is performed the second time it is not affected by the open processes.

For details on the Recovery object and its methods, see the *HP QuickTest Professional Object Model Reference*.

Tasks

How to Create and Manage Recovery Scenarios

This task describes how to perform different recovery scenario creation and management operations in the Recovery Scenario Manager dialog box. For a user interface description, see "Recovery Scenario Manager Dialog Box" on page 1534.

This task includes the following:

- "Define a recovery scenario file in which to store the recovery scenarios" on page 1528
- "Create a new recovery scenario operation using the Recovery Scenario Wizard" on page 1529
- "Manage existing recovery scenarios" on page 1529

Define a recovery scenario file in which to store the recovery scenarios

By default, the Recovery Scenario Manager dialog box opens with a new recovery file. You can use this new file, or you can open an existing recovery file, in one of the following ways:

- Click the arrow next to the **Open** button to select a recently-used recovery file from the list.
- Open an existing recovery file using the Open Recovery Scenario Dialog Box. For details, see "Open <Resource> Dialog Box" on page 406.

Create a new recovery scenario operation using the Recovery Scenario Wizard

- 1 In the Recovery Scenario Manager Dialog Box (described on page 1534), click the **New Scenario** button  . The Recovery Scenario Wizard opens.
- 2 Follow the on-screen instructions. The wizard includes the following pages (pages that are in parentheses open according to the option selected in the previous page):
 - "Select Trigger Event Page" on page 1541
 - ("Specify Pop-up Window Conditions Page" on page 1544)
 - ("Select Object Page" on page 1546)
 - ("Set Object Properties and Values Page" on page 1548)
 - ("Select Test Run Error Page" on page 1550)
 - ("Select Processes Page" on page 1552)
 - "Recovery Operations Page" on page 1554
 - "Recovery Operation Page" on page 1556
 - ("Recovery Operation - Click Button or Press Key Page" on page 1558)
 - ("Recovery Operation - Close Processes Page" on page 1560)
 - ("Recovery Operation - Function Call Screen" on page 1562)
 - "Post-Recovery Test Run Options Page" on page 1564
 - "Name and Description Page" on page 1566
 - "Completing the Recovery Scenario Wizard Page" on page 1568

Manage existing recovery scenarios

- View properties for your recovery scenarios in the Recovery Scenario Properties Dialog Box (described on page 1537).
- Edit, save, and remove existing recovery scenarios in the Recovery Scenario Manager by using the toolbar buttons.

How to Manage Recovery Scenario Associations

This task describes how to perform different recovery scenario management and association operations using the Test Settings dialog box. For a user interface description, see "Recovery Pane (Test Settings Dialog Box)" on page 1490.

This task includes the following:

- "Prerequisites" on page 1530
- "View read-only recovery scenario properties" on page 1530
- "Prioritize recovery scenarios" on page 1531
- "Remove recovery scenarios from your test" on page 1531
- "Enable/disable specific recovery scenarios" on page 1531
- "Set default recovery scenario settings for all new tests" on page 1531

Prerequisites

Display the **Recovery** pane in the Test Settings dialog box.

View read-only recovery scenario properties

- 1 In the **Scenarios** box, select the recovery scenario whose properties you want to view.
- 2 Do one of the following:



- Click the **Properties** button.
- Double-click a scenario in the **Scenarios** box.

The Recovery Scenario Properties dialog box opens, displaying read-only information regarding the settings for the selected scenario. For details, see "Recovery Scenario Properties Dialog Box" on page 1537.

Prioritize recovery scenarios



- 1 In the **Scenarios** box, select the scenario whose priority you want to change.
- 2 Click the **Up** or **Down** button. The selected scenario's priority changes according to your selection.

Remove recovery scenarios from your test



- 1 In the **Scenarios** box, select the scenario you want to remove.
- 2 Click the **Remove** button. The selected scenario is no longer associated with the test.

Enable/disable specific recovery scenarios

In the **Scenarios** box, perform one of the following:

- Select the check box to the left of one or more individual scenarios to enable them.
- Clear the check box to the left of one or more individual scenarios to disable them.

Set default recovery scenario settings for all new tests

Click the **Set as Default** button in the Recovery pane of the Test Settings dialog box to set the current list of recovery scenarios to be the default scenarios for all new tests. Any future changes you make to the current recovery scenario list only affect the current test, and do not change the default list that you defined.

Reference

Add Recovery Scenario Dialog Box

This dialog box enables you to associate existing scenarios with your test. This instructs QuickTest to perform the recovery scenarios during the run session.



To access	Select File > Settings > Recovery node, and click the Add button  .
Important information	If the recovery scenario you need is not available, you can create a new recovery scenario using the Recovery Scenario Manager (Resources > Recovery Scenario Manager). You can then associate it with your test using the Add Recovery Scenario dialog box.

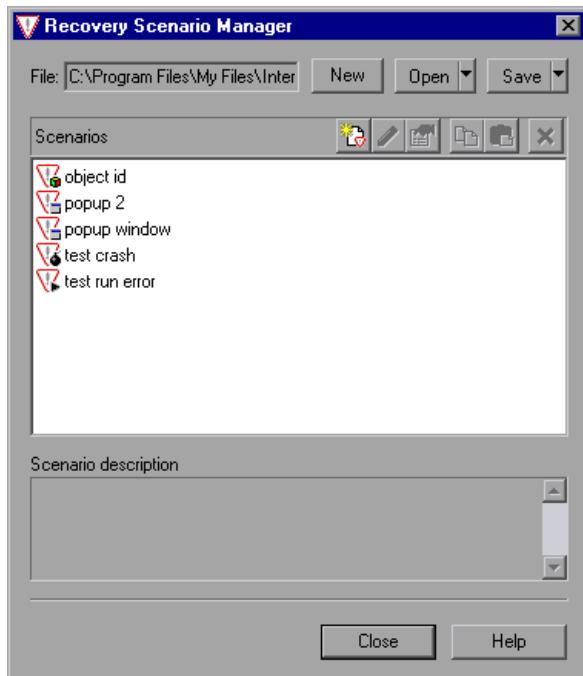
Relevant tasks	"How to Create and Manage Recovery Scenarios" on page 1528
See also	<ul style="list-style-type: none">➤ "Recovery Scenario Manager Dialog Box" on page 1534➤ "Recovery Scenario Wizard" on page 1539➤ "Recovery Scenario Properties Dialog Box" on page 1537

User interface elements are described below:

UI Elements	Description
Recovery file	The recovery file that contains the recovery scenario that you want to associate with the test. You can click the browse button to navigate to the recovery file you want to select.
Scenarios	The recovery scenarios in the recovery file selected in the Recovery file box.
Add Scenario	Associates the selected recovery scenario with the current test.

Recovery Scenario Manager Dialog Box

This dialog box enables you to create and edit recovery files, and create and manage the recovery scenarios stored in those files.



To access	Select Resources > Recovery Scenario Manager .
Relevant tasks	"How to Create and Manage Recovery Scenarios" on page 1528
See also	<ul style="list-style-type: none"> ➤ "Recovery Scenario Wizard" on page 1539 ➤ "Recovery Scenario Properties Dialog Box" on page 1537

This dialog box contains the following key elements:

- "General User Interface Elements" on page 1535
- "Scenarios Box Icons" on page 1536

General User Interface Elements

UI Elements	Description
File	The name of the selected recovery scenario file.
New	Creates a new recovery file.
Open	Opens an existing recovery file. You can also click the arrow to select a recovery file from the list of recently-used recovery files.
Save	Saves the current recovery file.
Version Control	Enables you to manage version control for your recovery scenarios. (Options are available only if QuickTest is connected to a version control-enabled Quality Center project.) For details, see "Managing Versions of Assets in Quality Center Overview" on page 1696 and "Viewing and Comparing Versions of QuickTest Assets" on page 1669.
Scenarios	The list of associated recovery scenario files. For details, see "Scenarios Box Icons" on page 1536.
	New Scenario. Opens the Recovery Scenario Wizard, in which you define a new recovery scenario. For details, see "Recovery Scenario Wizard" on page 1539.
	Edit. Opens the Recovery Scenario Wizard for the selected recovery scenario, in which you can modify the recovery scenario settings. For details, see "Recovery Scenario Wizard" on page 1539.
	Properties. Displays summary properties for the selected recovery scenario in read-only format. For details, see "Recovery Scenario Properties Dialog Box" on page 1537.
	Copy. Copies a recovery scenario from the open recovery file to the Clipboard. This enables you to paste a recovery scenario into another recovery file.

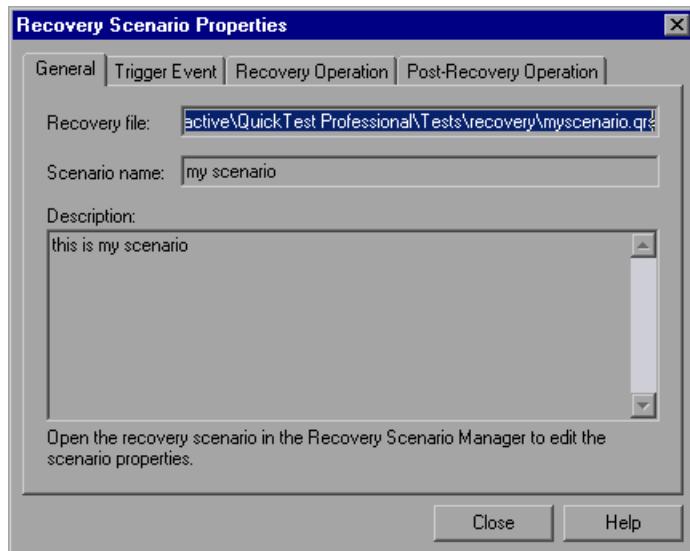
UI Elements	Description
	<p>Paste. Pastes a recovery scenario from the Clipboard into the open recovery file.</p> <p>Note: If a scenario with the same name already exists in the recovery scenario file, you can choose whether you want to replace it with the new scenario you have just copied.</p>
	<p>Delete. Deletes a recovery scenario.</p> <p>Note: If a deleted recovery scenario is associated with a test, QuickTest ignores it during the run session.</p>
Scenario description	The description for the currently selected recovery scenario file.

Scenarios Box Icons

Icon	Description
	Indicates that the recovery scenario is triggered when a window pops up in an open application during the run session.
	Indicates that the recovery scenario is triggered when the property values of an object in an application match specified values.
	Indicates that the recovery scenario is triggered when a step in the test does not run successfully.
	Indicates that the recovery scenario is triggered when an open application fails during the run session.

Recovery Scenario Properties Dialog Box

This dialog box enables you to view read-only properties for any defined recovery scenario.



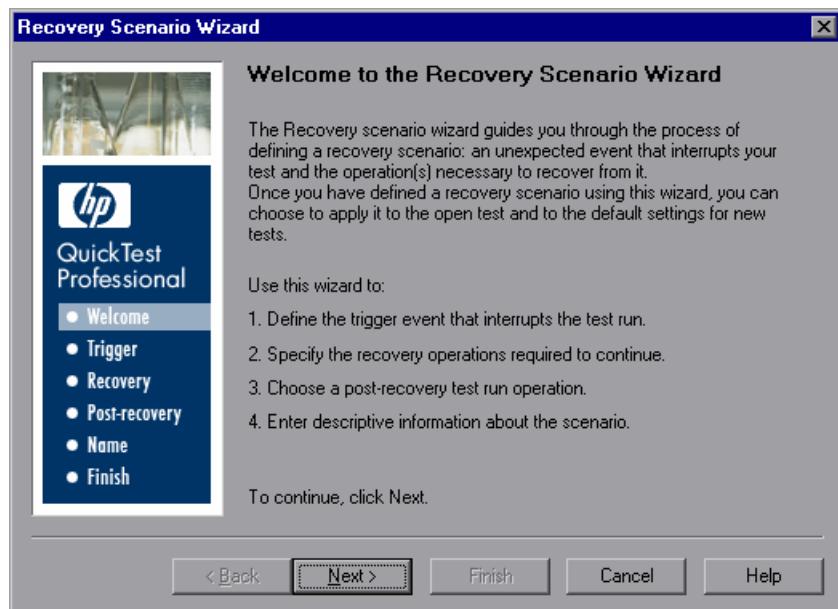
To access	<ul style="list-style-type: none"> ➤ In the Recover Scenario Manager dialog box, select a scenario and click the Properties button . ➤ In the Recovery pane of the Test Settings dialog box, select a scenario and click the Properties button .
Important information	Properties are displayed in read-only format. To modify any of the properties, use the Edit button in the Recovery Scenario Manager dialog box.
Relevant tasks	"How to Create and Manage Recovery Scenarios" on page 1528
See also	<ul style="list-style-type: none"> ➤ "Recovery Scenario Manager Dialog Box" on page 1534 ➤ "Recovery Scenario Wizard" on page 1539

The Recovery Scenario Properties dialog box displays read-only information about the selected scenario in the following tabs:

UI Elements	Description
General	The name and description defined for the recovery scenario, plus the name and path of the recovery file in which the scenario is saved.
Trigger Event	The settings for the trigger event defined for the recovery scenario.
Recovery Operation	The recovery operations defined for the recovery scenario.
Post-Recovery Operation	The post-recovery operation defined for the recovery scenario.

Recovery Scenario Wizard

This wizard enables you to create a recovery scenario to associate with your test.



To access	In the Recovery Scenario Manager Dialog Box (described on page 1534), click the New Scenario button  .
Relevant tasks	"How to Create and Manage Recovery Scenarios" on page 1528

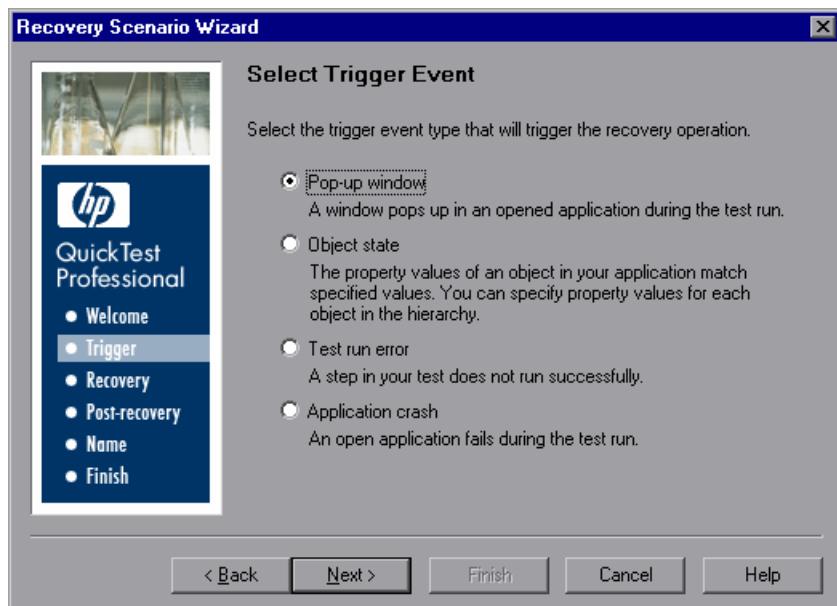
Wizard map	<p>This wizard contains:</p> <p>Welcome > Select Trigger Event Page (page 1541) > (Specify Pop-up Window Conditions Page (page 1544)) > (Select Object Page (page 1546)) > (Set Object Properties and Values Page (page 1548)) > (Select Test Run Error Page (page 1550)) > (Select Processes Page (page 1552)) > Recovery Operations Page (page 1554) > Recovery Operation Page (page 1556) > (Recovery Operation - Click Button or Press Key Page (page 1558)) > (Recovery Operation - Close Processes Page (page 1560)) > (Recovery Operation - Function Call Screen (page 1562)) > Post-Recovery Test Run Options Page (page 1564) > Name and Description Page (page 1566) > Completing the Recovery Scenario Wizard Page (page 1568)</p> <p>Note: Pages that are in parentheses open according to the option selected in the previous page. Therefore, not all pages are displayed for every option.</p>
See also	"Recovery Scenarios Overview" on page 1524

User interface elements are described below (unlabeled elements are shown in angle brackets):

UI Elements	Description
<welcome area>	General information on the different options in the Recovery Scenario Wizard, and an overview of the stages involved in defining a recovery scenario.

Select Trigger Event Page

This page enables you to define the event type that triggers the recovery scenario, and the way in which QuickTest recognizes the event.



Wizard map	<p>The Recovery Scenario Wizard contains:</p> <p>Welcome > Select Trigger Event Page > (Specify Pop-up Window Conditions Page) > (Select Object Page) > (Set Object Properties and Values Page) > (Select Test Run Error Page) > (Select Processes Page) > Recovery Operations Page > Recovery Operation Page > (Recovery Operation - Click Button or Press Key Page) > (Recovery Operation - Close Processes Page) > (Recovery Operation - Function Call Screen) > Post-Recovery Test Run Options Page > Name and Description Page > Completing the Recovery Scenario Wizard Page</p> <p>Note: Pages that are in parentheses open according to the option selected in the previous page. Therefore, not all pages are displayed for every option.</p>
-------------------	--

Important information	<ul style="list-style-type: none"> ➤ The set of recovery operations is performed for each occurrence of the trigger event criteria. <p>Example: If you define a specific object state, and two objects match this state, the set of recovery operations is performed two times, once for each object that matches the specified state.</p> <ul style="list-style-type: none"> ➤ The recovery mechanism does not handle triggers that occur in the last step of a test. If you need to recover from an unexpected event or error that may occur in the last step of a test, you can do this by adding an extra step to the end of your test.
------------------------------	--

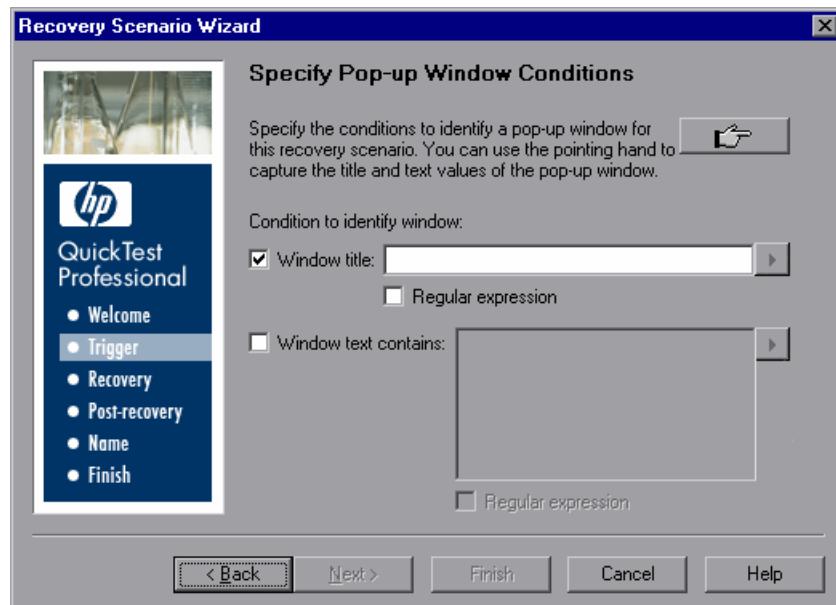
User interface elements are described below:

UI Elements	Description
Pop-up window	<p>QuickTest detects a pop-up window and identifies it according to the window title and textual content.</p> <p>Example: A message box may open during a run session, indicating that the printer is out of paper. QuickTest can detect this window and activate a defined recovery scenario to continue the run session.</p>
Object state	<p>QuickTest detects a specific test object state and identifies it according to its property values and the property values of all its ancestors. Note that an object is identified only by its property values, and not by its class.</p> <p>Example: A specific button in a dialog box may be disabled when a specific process is open. QuickTest can detect the object property state of the button that occurs when this problematic process is open and activate a defined recovery scenario to close the process and continue the run session.</p>

UI Elements	Description
Test run error	<p>QuickTest detects a run error and identifies it by a failed return value from a method.</p> <p>Example: QuickTest may not be able to identify a menu item specified in the method argument, due to the fact that the menu item is not available at a specific point during the run session. QuickTest can detect this run error and activate a defined recovery scenario to continue the run session.</p>
Application crash	<p>QuickTest detects an application crash and identifies it according to a predefined list of applications.</p> <p>Example: A secondary application may crash when a certain step is performed in the run session. You want to be sure that the run session does not fail because of this crash, which may indicate a different problem with your application. QuickTest can detect this application crash and activate a defined recovery scenario to continue the run session.</p>

Specify Pop-up Window Conditions Page

This page enables you to specify how the pop-up window should be identified.



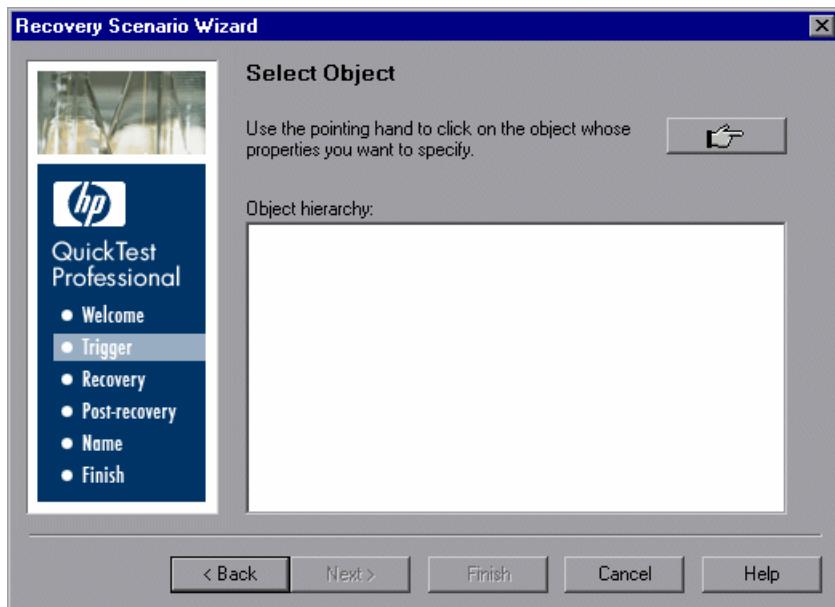
Wizard map	<p>The Recovery Scenario Wizard contains:</p> <p>Welcome > Select Trigger Event Page > (Specify Pop-up Window Conditions Page) > (Select Object Page) > (Set Object Properties and Values Page) > (Select Test Run Error Page) > (Select Processes Page) > Recovery Operations Page > Recovery Operation Page > (Recovery Operation - Click Button or Press Key Page) > (Recovery Operation - Close Processes Page) > (Recovery Operation - Function Call Screen) > Post-Recovery Test Run Options Page > Name and Description Page > Completing the Recovery Scenario Wizard Page</p> <p>Note: Pages that are in parentheses open according to the option selected in the previous page. Therefore, not all pages are displayed for every option.</p>
-------------------	--

User interface elements are described below:

UI Elements	Description
	Pointing hand. Instructs QuickTest to identify only pop-up windows that match the object property values of the window you select. For information on using the pointing hand, see "Tips for Using the Pointing Hand" on page 157.
Window title	Instructs QuickTest to identify any pop-up window that contains the relevant title.
Window text	Instructs QuickTest to identify any pop-up window that contains the relevant text.
Regular expression	Enables you to use regular expressions to identify the pop-up window. For information on regular expressions, see "Regular Expressions Overview" on page 863. Note: You can click the right arrow to display a list of regular expression characters that you can select, and to open the Regular Expression Evaluator, which enables you to test your regular expression. For details, see "Smart Regular Expression List" on page 884 and "Regular Expression Evaluator" on page 882.

Select Object Page

This page enables you to select the object whose properties you want to specify.



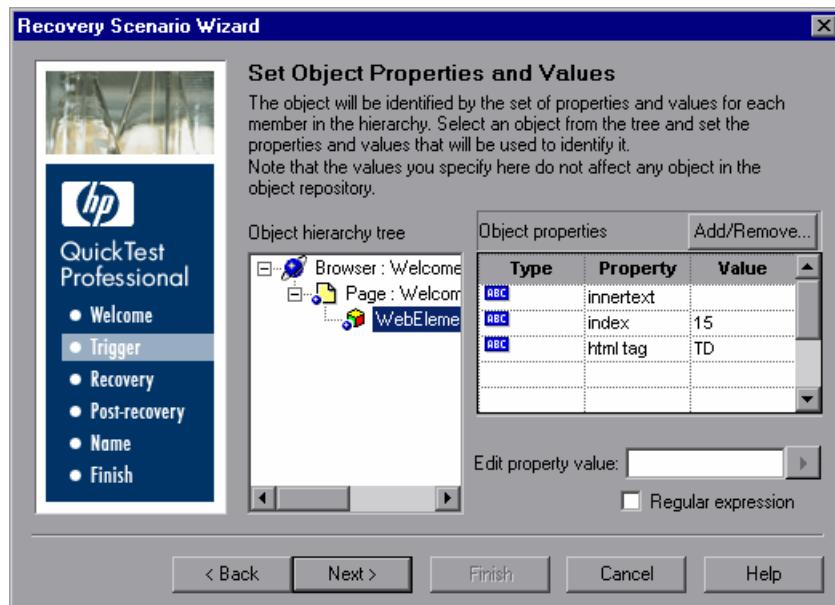
Wizard map	<p>The Recovery Scenario Wizard contains:</p> <p>Welcome > Select Trigger Event Page > (Specify Pop-up Window Conditions Page) > (Select Object Page) > (Set Object Properties and Values Page) > (Select Test Run Error Page) > (Select Processes Page) > Recovery Operations Page > Recovery Operation Page > (Recovery Operation - Click Button or Press Key Page) > (Recovery Operation - Close Processes Page) > (Recovery Operation - Function Call Screen) > Post-Recovery Test Run Options Page > Name and Description Page > Completing the Recovery Scenario Wizard Page</p> <p>Note: Pages that are in parentheses open according to the option selected in the previous page. Therefore, not all pages are displayed for every option.</p>
-------------------	--

User interface elements are described below:

UI Elements	Description
	<p>Pointing hand. Enables you to click the object whose properties you want to specify. For information on using the pointing hand, see "Tips for Using the Pointing Hand" on page 157.</p> <p>If the location you click is associated with more than one object, the Object Selection dialog box opens. Select the object whose properties you want to specify and click OK. The selected object and its parents are displayed in the Select Object screen. For details, see "Object Selection Dialog Box" on page 155.</p> <p>Note: The hierarchical object selection tree also enables you to select an object that QuickTest would not ordinarily learn (a non-parent object), such as a Web table.</p>
Object hierarchy	The path to the selected object.

Set Object Properties and Values Page

This page enables you to specify the properties and values for the selected object to use in the recovery scenario.



Wizard map

The Recovery Scenario Wizard contains:

Welcome > Select Trigger Event Page > (Specify Pop-up Window Conditions Page) > (Select Object Page) > (**Set Object Properties and Values Page**) > (Select Test Run Error Page) > (Select Processes Page) > Recovery Operations Page > Recovery Operation Page > (Recovery Operation - Click Button or Press Key Page) > (Recovery Operation - Close Processes Page) > (Recovery Operation - Function Call Screen) > Post-Recovery Test Run Options Page > Name and Description Page > Completing the Recovery Scenario Wizard Page

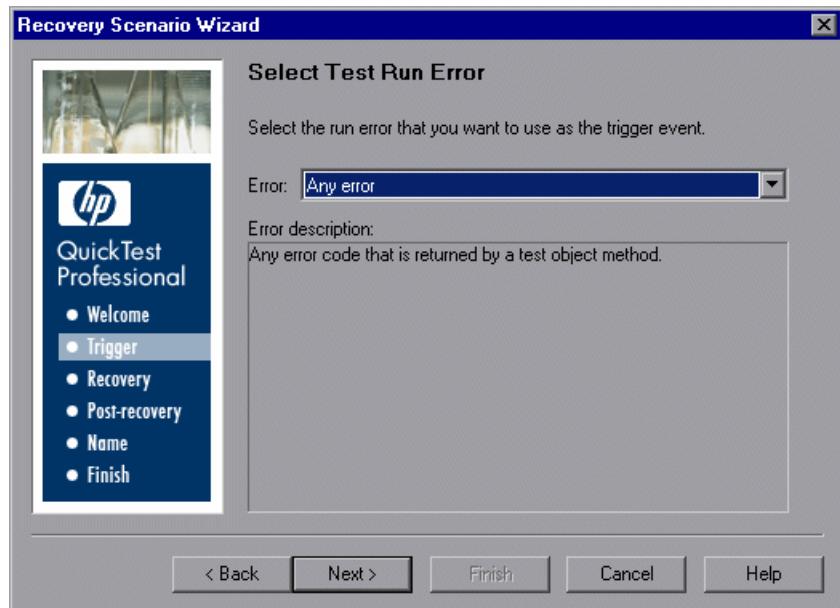
Note: Pages that are in parentheses open according to the option selected in the previous page. Therefore, not all pages are displayed for every option.

User interface elements are described below:

UI Elements	Description
Object hierarchy tree	The path to the selected object.
Object properties	The list of available properties for the selected object.
	Enables you to add or remove object properties from the list of property values to check. Note: An object is identified only by its property values, and not by its class.
Edit property value	Enables you to modify the property values used to identify the object.
Regular expression	Enables you to use regular expressions in the property value. For information on regular expressions, see "Regular Expressions Overview" on page 863. Note: You can click the right arrow to display a list of regular expression characters that you can select, and to open the Regular Expression Evaluator, which enables you to test your regular expression. For details, see "Smart Regular Expression List" on page 884 and "Regular Expression Evaluator" on page 882.

Select Test Run Error Page

This page enables you to select the type of error to use as a trigger event.



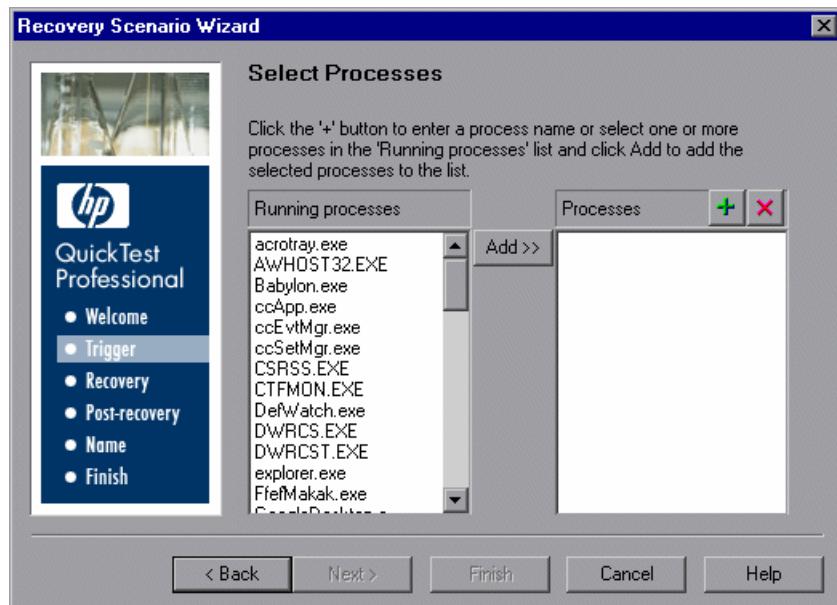
Wizard map	<p>The Recovery Scenario Wizard contains:</p> <p>Welcome > Select Trigger Event Page > (Specify Pop-up Window Conditions Page) > (Select Object Page) > (Set Object Properties and Values Page) > (Select Test Run Error Page) > (Select Processes Page) > Recovery Operations Page > Recovery Operation Page > (Recovery Operation - Click Button or Press Key Page) > (Recovery Operation - Close Processes Page) > (Recovery Operation - Function Call Screen) > Post-Recovery Test Run Options Page > Name and Description Page > Completing the Recovery Scenario Wizard Page</p> <p>Note: Pages that are in parentheses open according to the option selected in the previous page. Therefore, not all pages are displayed for every option.</p>
-------------------	--

User interface elements are described below:

UI Elements	Description
Error	<p>The list of error types to use as trigger events. The following option are available:</p> <ul style="list-style-type: none"> ▶ Any error. Any error code that is returned by a test object method. ▶ Item in list or menu is not unique. Occurs when more than one item in the list, menu, or tree has the name specified in the method argument. ▶ Item in list or menu not found. Occurs when QuickTest cannot identify the list, menu, or tree item specified in the method argument. This may be due to the fact that the item is not currently available or that its name has changed. ▶ More than one object responds to the physical description. Occurs when more than one object in your application has the same property values as those specified in the test object description for the object specified in the step. ▶ Object is disabled. Occurs when QuickTest cannot perform the step because the object specified in the step is currently disabled. ▶ Object not found. Occurs when no object within the specified parent object matches the test object description for the object. ▶ Object not visible. Occurs when QuickTest cannot perform the step because the object specified in the step is not currently visible on the screen.
Error description	The description of the selected error type.

Select Processes Page

This page enables you to select processes that will trigger a recovery scenario if they crash.



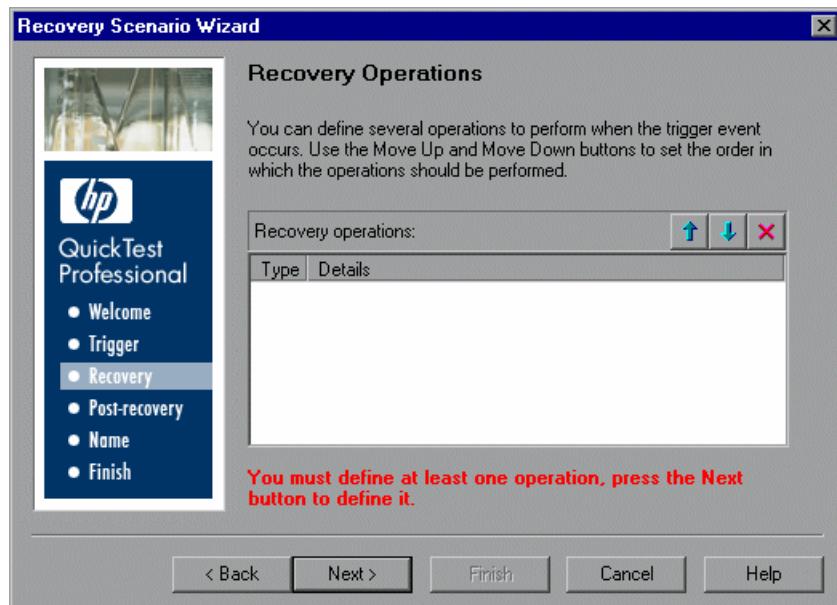
Wizard map	<p>The Recovery Scenario Wizard contains:</p> <p>Welcome > Select Trigger Event Page > (Specify Pop-up Window Conditions Page) > (Select Object Page) > (Set Object Properties and Values Page) > (Select Test Run Error Page) > (Select Processes Page) > Recovery Operations Page > Recovery Operation Page > (Recovery Operation - Click Button or Press Key Page) > (Recovery Operation - Close Processes Page) > (Recovery Operation - Function Call Screen) > Post-Recovery Test Run Options Page > Name and Description Page > Completing the Recovery Scenario Wizard Page</p> <p>Note: Pages that are in parentheses open according to the option selected in the previous page. Therefore, not all pages are displayed for every option.</p>
-------------------	--

User interface elements are described below:

UI Elements	Description
Running processes	<p>The list of all application processes that are currently running.</p> <p>To add a process from the Running processes list, double-click a process in the Running processes list or select it and click the Add button. You can select multiple processes using standard Windows multiple selection techniques (CTRL and SHIFT keys).</p>
Processes	<p>The list of the application processes that will trigger the recovery scenario if they crash.</p> <p>Notes:</p> <ul style="list-style-type: none"> ➤ You can add application processes to the Processes list by typing them in the Processes list or by selecting them from the Running processes list. ➤ You can modify the name of a process by selecting it in the Processes list and clicking the process name to edit it.
	<p>Add new process. Enables you to add a process directly to the Processes list by entering the name of any process you want to add to the list.</p>
	<p>Remove process. Enables you to remove a process from the Processes list.</p>

Recovery Operations Page

This page enables you to manage the collection of recovery operations in the recovery scenario.



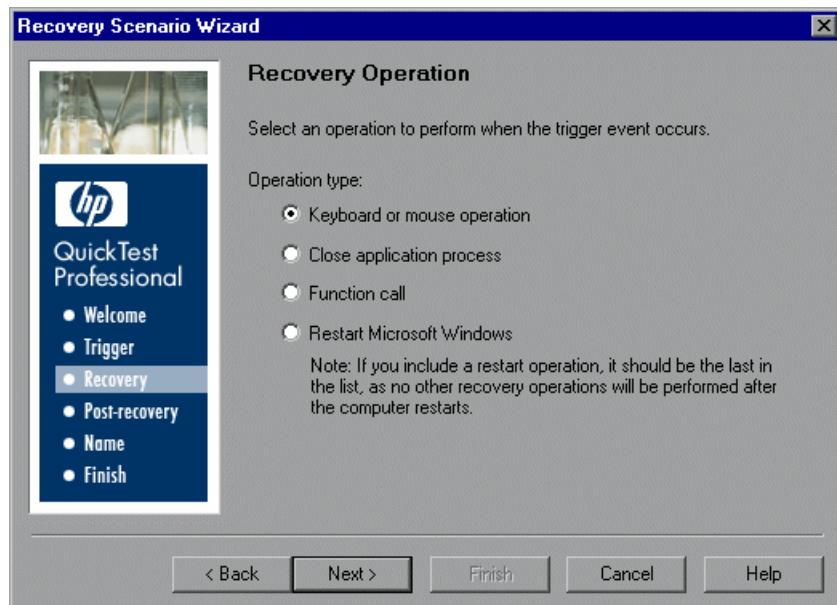
Wizard map	<p>The Recovery Scenario Wizard contains:</p> <p>Welcome > Select Trigger Event Page > (Specify Pop-up Window Conditions Page) > (Select Object Page) > (Set Object Properties and Values Page) > (Select Test Run Error Page) > (Select Processes Page) > Recovery Operations Page > Recovery Operation Page > (Recovery Operation - Click Button or Press Key Page) > (Recovery Operation - Close Processes Page) > (Recovery Operation - Function Call Screen) > Post-Recovery Test Run Options Page > Name and Description Page > Completing the Recovery Scenario Wizard Page</p> <p>Note: Pages that are in parentheses open according to the option selected in the previous page. Therefore, not all pages are displayed for every option.</p>
-------------------	--

User interface elements are described below:

UI Elements	Description
Recovery operations	<p>The list of selected recovery operations.</p> <p>To add a recovery scenario to the list, click Next to continue to the Recovery Operation Page (described on page 1556).</p> <p>Notes:</p> <ul style="list-style-type: none"> ▶ If you define two or more recovery operations, you can select a recovery operation and use the Move Up or Move Down buttons to change the order in which QuickTest performs the recovery operations. ▶ You can also select a recovery operation and click the Remove button to delete a recovery operation from the recovery scenario. ▶ If you define a Restart Microsoft Windows recovery operation, it is always inserted as the last recovery operation, and you cannot change its position in the list.
Add another recovery operation	<p>Displayed only after you have defined at least one recovery operation. The following options are available:</p> <ul style="list-style-type: none"> ▶ Select the check box and click Next to define another recovery operation. ▶ Clear the check box and click Next to continue to the Post-Recovery Test Run Options Page (described on page 1564).

Recovery Operation Page

This page enables you to specify the operations QuickTest performs after it detects the trigger event.



Wizard map

The Recovery Scenario Wizard contains:

Welcome > Select Trigger Event Page > (Specify Pop-up Window Conditions Page) > (Select Object Page) > (Set Object Properties and Values Page) > (Select Test Run Error Page) > (Select Processes Page) > Recovery Operations Page > **Recovery Operation Page** > (Recovery Operation - Click Button or Press Key Page) > (Recovery Operation - Close Processes Page) > (Recovery Operation - Function Call Screen) > Post-Recovery Test Run Options Page > Name and Description Page > Completing the Recovery Scenario Wizard Page

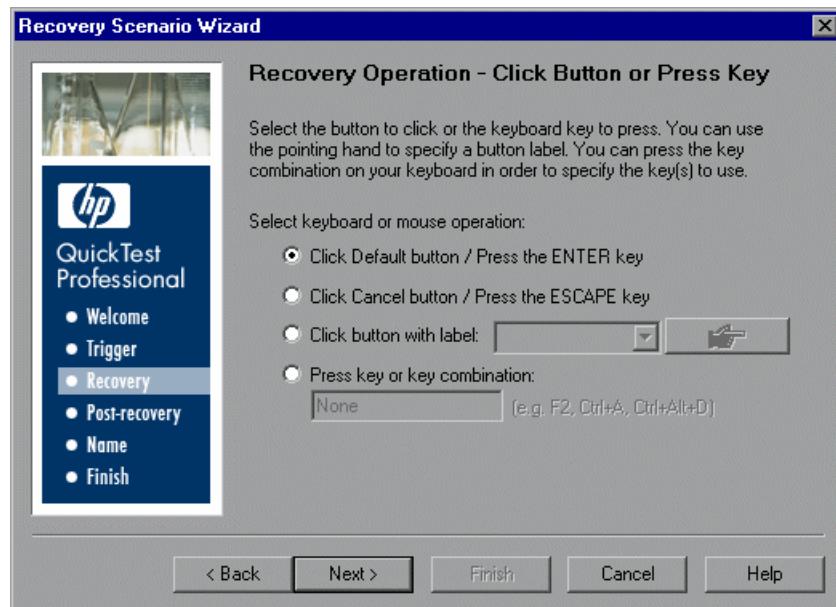
Note: Pages that are in parentheses open according to the option selected in the previous page. Therefore, not all pages are displayed for every option.

User interface elements are described below:

UI Elements	Description
Keyboard or mouse operation	QuickTest simulates a click on a button in a window or a press of a keyboard key.
Close application process	QuickTest closes specified processes.
Function call	QuickTest calls a VBScript function.
Restart Microsoft Windows	QuickTest restarts Microsoft Windows. Note: If you use the this recovery operation, you must ensure that any test associated with this recovery scenario is saved before you run it. You must also configure the computer on which the test is run to automatically log in on restart.

Recovery Operation - Click Button or Press Key Page

This page enables you to specify the keyboard or mouse operation that you want QuickTest to perform when it detects the trigger event:



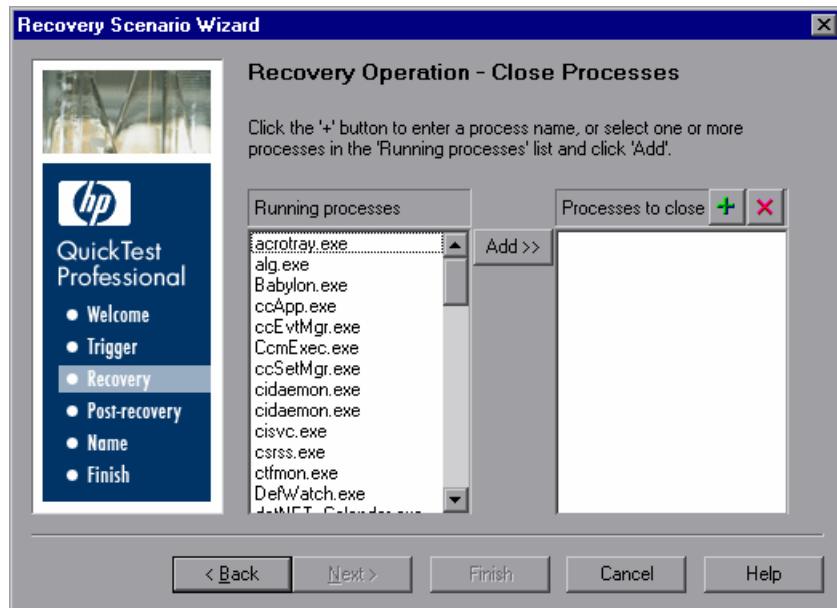
Wizard map	<p>The Recovery Scenario Wizard contains:</p> <p>Welcome > Select Trigger Event Page > (Specify Pop-up Window Conditions Page) > (Select Object Page) > (Set Object Properties and Values Page) > (Select Test Run Error Page) > (Select Processes Page) > Recovery Operations Page > Recovery Operation Page > (Recovery Operation - Click Button or Press Key Page) > (Recovery Operation - Close Processes Page) > (Recovery Operation - Function Call Screen) > Post-Recovery Test Run Options Page > Name and Description Page > Completing the Recovery Scenario Wizard Page</p> <p>Note: Pages that are in parentheses open according to the option selected in the previous page. Therefore, not all pages are displayed for every option.</p>
-------------------	--

User interface elements are described below:

UI Elements	Description
Click Default button / Press the ENTER key	Instructs QuickTest to click the default button or press the ENTER key in the displayed window when the trigger occurs.
Click Cancel button / Press the ESCAPE key	Instructs QuickTest to click the Cancel button or press the ESCAPE key in the displayed window when the trigger occurs.
Click button with label	<p>Instructs QuickTest to click the button with the specified label in the displayed window when the trigger occurs. If you select this option, click the pointing hand and then click anywhere in the trigger window. For information on using the pointing hand, see "Tips for Using the Pointing Hand" on page 157.</p> <p>All button labels in the selected window are displayed in the list box. Select the required button from the list.</p>
Press key or key combination	Instructs QuickTest to press the specified keyboard key or key combination in the displayed window when the trigger occurs. If you select this option, click in the edit box and then press the key or key combination on your keyboard that you want to specify.

Recovery Operation - Close Processes Page

This page enables you to select processes to close in the recovery operation.



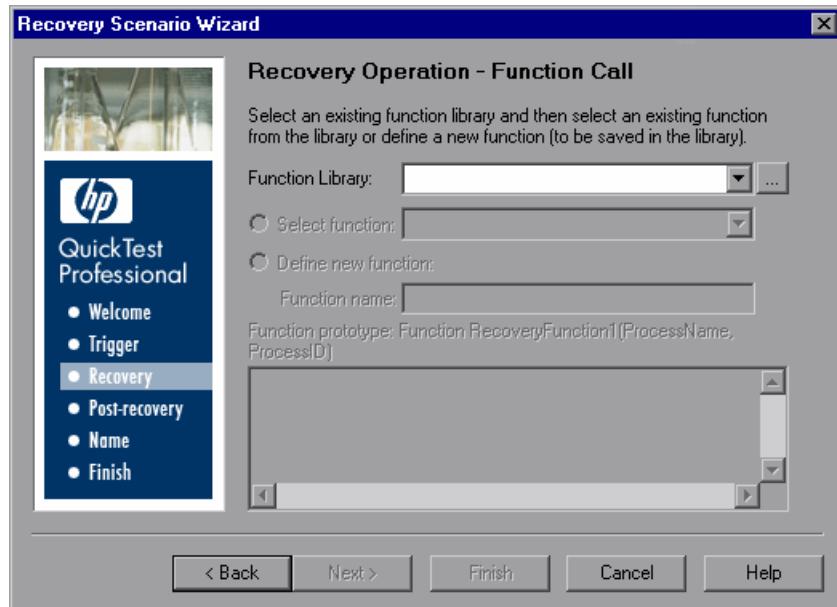
Wizard map	<p>The Recovery Scenario Wizard contains:</p> <p>Welcome > Select Trigger Event Page > (Specify Pop-up Window Conditions Page) > (Select Object Page) > (Set Object Properties and Values Page) > (Select Test Run Error Page) > (Select Processes Page) > Recovery Operations Page > Recovery Operation Page > (Recovery Operation - Click Button or Press Key Page) > (Recovery Operation - Close Processes Screen) > (Recovery Operation - Function Call Screen) > Post-Recovery Test Run Options Page > Name and Description Page > Completing the Recovery Scenario Wizard Page</p> <p>Note: Pages that are in parentheses open according to the option selected in the previous page. Therefore, not all pages are displayed for every option.</p>
-------------------	--

User interface elements are described below:

UI Elements	Description
Running processes	<p>The list of all application processes that are currently running.</p> <p>To add a process from the Running processes list, double-click a process in the Running processes list or select it and click the Add button. You can select multiple processes using standard Windows multiple selection techniques (CTRL and SHIFT keys).</p>
Processes to close	<p>The list of the application processes that will be closed when the trigger is activated.</p> <p>Note: You can modify the name of a process by selecting it in the Processes to close list and clicking the process name to edit it.</p>
	<p>Add new process. Enables you to add a process directly to the Processes list by entering the name of any process you want to add to the list.</p>
	<p>Remove process. Enables you to remove a process from the Processes list.</p>

Recovery Operation - Function Call Screen

This page enables you to select a function to call in the recovery operation.



Wizard map	<p>The Recovery Scenario Wizard contains:</p> <p>Welcome > Select Trigger Event Page > (Specify Pop-up Window Conditions Page) > (Select Object Page) > (Set Object Properties and Values Page) > (Select Test Run Error Page) > (Select Processes Page) > Recovery Operations Page > Recovery Operation Page > (Recovery Operation - Click Button or Press Key Page) > (Recovery Operation - Close Processes Page) > (Recovery Operation - Function Call Page) > Post-Recovery Test Run Options Page > Name and Description Page > Completing the Recovery Scenario Wizard Page</p> <p>Note: Pages that are in parentheses open according to the option selected in the previous page. Therefore, not all pages are displayed for every option.</p>
-------------------	--

Important information	<ul style="list-style-type: none"> ➤ QuickTest automatically associates the function library you select with your test. Therefore, you do not need to associate the function library with your test in the Resources pane of the Test Settings dialog box. ➤ If more than one scenario uses a function with the same name from different function libraries, the recovery process may fail. In this case, information regarding the recovery failure is displayed during the run session.
------------------------------	---

User Interface Elements

UI Elements	Description
Function Library	Displays the selected function library, and enables you to select a recently specified function library in the box. Alternatively, click the Browse button to navigate to an existing function library.
Select function	Choose an existing function from the function library you selected. Only functions that match the prototype syntax for the trigger type selected in the "Select Trigger Event Page" on page 1541 are displayed.
Define new function	Create a new function by specifying a unique name for it, and defining the function in the Function Name box according to the displayed function prototype. The new function is added to the function library you selected.

Function Prototypes by Trigger Type

Following is the prototype for each trigger type:

Test run error trigger

```
OnRunStep
(
[in] Object as Object: The object of the current step.
[in] Method as String: The method of the current step.
[in] Arguments as Array: The actual method's arguments.
[in] Result as Integer: The actual method's result.
)
```

Pop-up window and Object state triggers

OnObject

(

[in] Object as Object: The detected object.

)

Application crash trigger

OnProcess

(

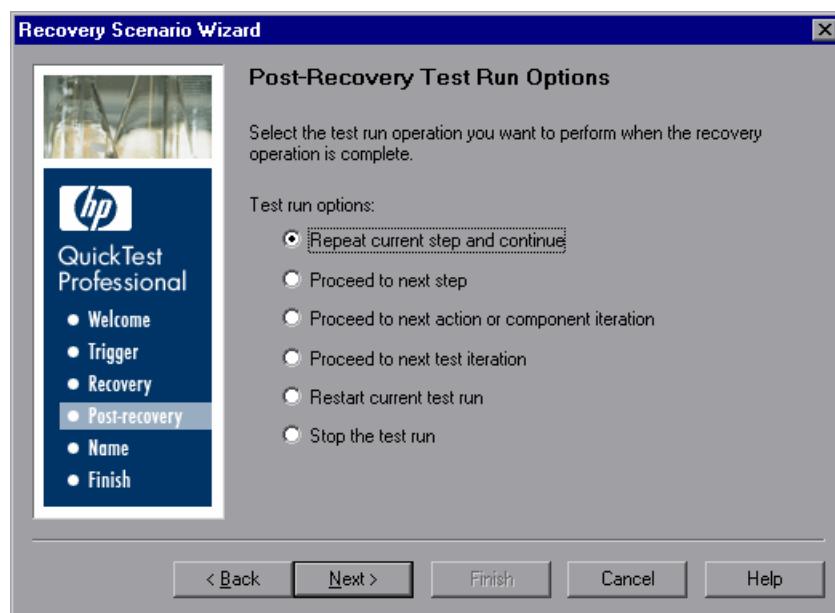
[in] ProcessName as String: The detected process's Name.

[in] ProcessId as Integer: The detected process' ID.

)

Post-Recovery Test Run Options Page

This page enables you to define post-recovery test run options, which specify how to continue the run session after QuickTest has identified the event and performed all of the specified recovery operations.



Wizard map	<p>The Recovery Scenario Wizard contains:</p> <p>Welcome > Select Trigger Event Page > (Specify Pop-up Window Conditions Page) > (Select Object Page) > (Set Object Properties and Values Page) > (Select Test Run Error Page) > (Select Processes Page) > Recovery Operations Page > Recovery Operation Page > (Recovery Operation - Click Button or Press Key Page) > (Recovery Operation - Close Processes Page) > (Recovery Operation - Function Call Screen) > Post-Recovery Test Run Options Page > Name and Description Page > Completing the Recovery Scenario Wizard Page</p> <p>Note: Pages that are in parentheses open according to the option selected in the previous page. Therefore, not all pages are displayed for every option.</p>
Important information	<p>If you chose Restart Microsoft Windows as a recovery operation, you can choose from only the last two test run options.</p>

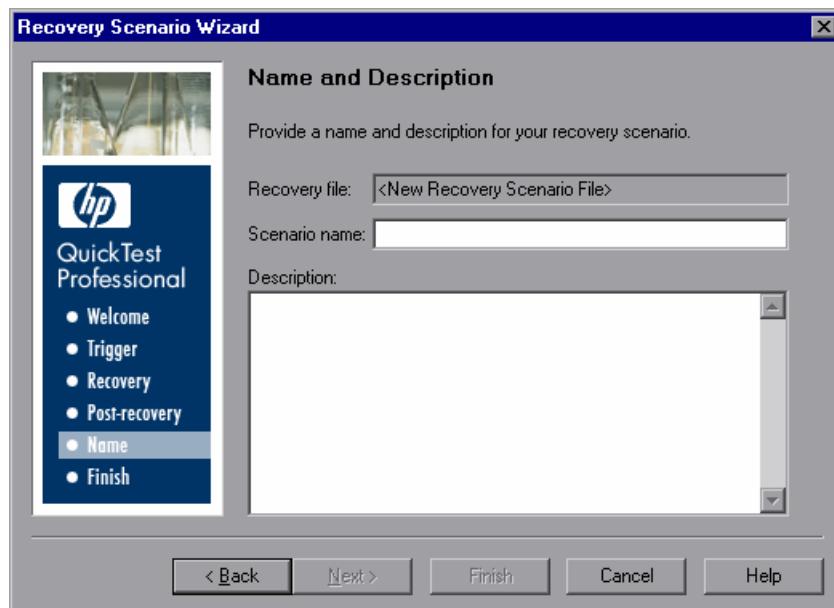
User interface elements are described below:

UI Elements	Description
Repeat current step and continue	<p>The current step is the step that QuickTest was running when the recovery scenario was triggered. If you are using the On error activation option for recovery scenarios, the step that returns the error is often one or more steps later than the step that caused the trigger event to occur. Therefore, in most cases, repeating the current step does not cause the trigger event to occur again.</p>
Proceed to next step	<p>Skips the step that QuickTest was running when the recovery scenario was triggered. Keep in mind that skipping a step that performs operations on your application may cause subsequent steps to fail.</p>

UI Elements	Description
Proceed to next action or component iteration	Stops performing steps in the current action or component iteration and begins the next iteration from the beginning (or from the next action or component if no additional iterations of the current action or component are required).
Proceed to next test iteration	Stops performing steps in the current action and begins the next QuickTest test iteration from the beginning (or stops running the test if no additional iterations of the test are required).
Restart current test run	Stops performing steps and re-runs the test from the beginning.
Stop the test run	Stops running the test.

Name and Description Page

This page enables you to specify a name by which to identify your recovery scenario. You can also add descriptive information regarding the scenario.



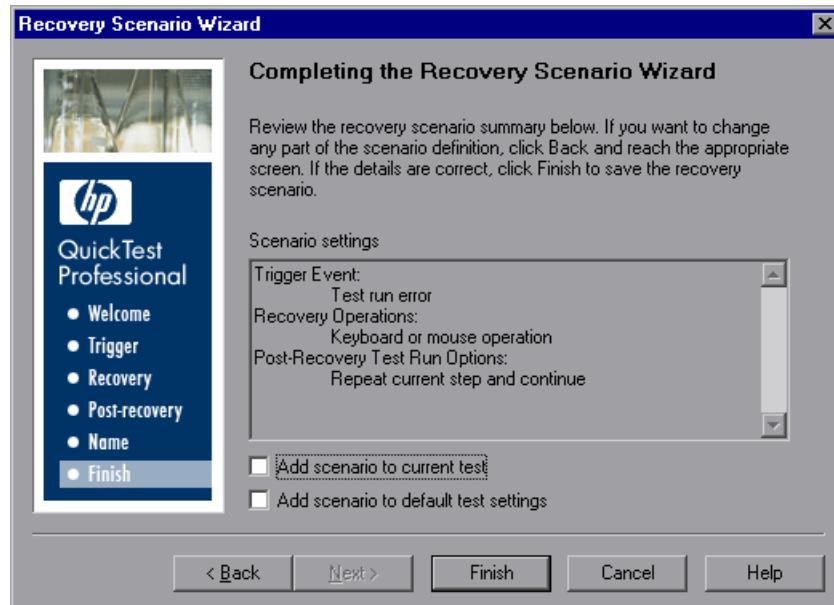
Wizard map	<p>The Recovery Scenario Wizard contains:</p> <p>Welcome > Select Trigger Event Page > (Specify Pop-up Window Conditions Page) > (Select Object Page) > (Set Object Properties and Values Page) > (Select Test Run Error Page) > (Select Processes Page) > Recovery Operations Page > Recovery Operation Page > (Recovery Operation - Click Button or Press Key Page) > (Recovery Operation - Close Processes Page) > (Recovery Operation - Function Call Screen) > Post-Recovery Test Run Options Page > Name and Description Page > Completing the Recovery Scenario Wizard Page</p> <p>Note: Pages that are in parentheses open according to the option selected in the previous page. Therefore, not all pages are displayed for every option.</p>
-------------------	--

User interface elements are described below:

UI Elements	Description
Recovery file	The name of the recovery file.
Scenario name	The name of the scenario operation.
Description	The textual description of the scenario operation.

Completing the Recovery Scenario Wizard Page

This page enables you to review a summary of the scenario settings you defined. You can also specify whether to automatically associate the recovery scenario with the current test and/or to add it to the default settings for all new tests.



Wizard map

The Recovery Scenario Wizard contains:

Welcome > Select Trigger Event Page > (Specify Pop-up Window Conditions Page) > (Select Object Page) > (Set Object Properties and Values Page) > (Select Test Run Error Page) > (Select Processes Page) > Recovery Operations Page > Recovery Operation Page > (Recovery Operation - Click Button or Press Key Page) > (Recovery Operation - Close Processes Page) > (Recovery Operation - Function Call Screen) > Post-Recovery Test Run Options Page > Name and Description Page > **Completing the Recovery Scenario Wizard Screen**

Note: Pages that are in parentheses open according to the option selected in the previous page. Therefore, not all pages are displayed for every option.

User interface elements are described below:

UI Elements	Description
Scenario settings	The settings of the defined recovery operation.
Add scenario to current test	You can select the Add scenario to current test check box to associate this recovery scenario with the current test. When you click Finish , QuickTest adds the recovery scenario to the Scenarios list in the Recovery pane of the Test Settings dialog box.
Add scenario to default test settings	You can select the Add scenario to default test settings check box to make this recovery scenario a default scenario for all new tests. The next time you create a test, this scenario will be listed in the Scenarios list in the Recovery pane of the Test Settings dialog box. Note: You can remove scenarios from the default scenarios list. For details, see "Recovery Pane (Test Settings Dialog Box)" on page 1490.

Troubleshooting and Limitations - Recovery Scenarios

This section describes troubleshooting and limitations for recovery scenarios.

- If you specify multiple function libraries from different locations with the same name in the same recovery scenario, only the first function library is used.

Workaround: Rename the function libraries so that each function library has a unique name.

50

QuickTest Script Editor

This chapter includes:

Concepts

- ▶ QuickTest Script Editor Overview on page 1572

Reference

- ▶ QuickTest Script Editor Window on page 1575
- ▶ Flow Pane (Script Editor) on page 1582
- ▶ Resources Pane (Script Editor) on page 1584
- ▶ Customize Dialog Box (Script Editor) on page 1587

Concepts

QuickTest Script Editor Overview

The QuickTest Script Editor is a standalone tool that enables you to open and edit multiple test scripts and function libraries simultaneously regardless of whether they are stored in the file system or in Quality Center.

The following table illustrates when to use the QuickTest Script Editor and when to use QuickTest:

Use the Script Editor to:	Use QuickTest to:
<ul style="list-style-type: none">➤ simultaneously view and modify as many test scripts and function libraries as you want➤ drag and drop code and comments from one test or function library to another➤ create and modify function libraries➤ open and view the latest version of version control-enabled tests and function libraries stored in Quality Center (read-only) <p>Note: You cannot select a baseline when opening a version control-enabled function library.</p>	<ul style="list-style-type: none">➤ create or modify a test➤ create and modify one or more function libraries➤ associate or remove function library associations➤ change non-code-related information, such as existing test names, test settings, parameterization, or data table values➤ work with components stored in Quality Center➤ modify version control-enabled files stored in Quality Center

Considerations for Working with the QuickTest Script Editor

➤ **Opening a file in the QuickTest Script Editor locks it.** When you open a document in the QuickTest Script Editor, it is locked, and no other users can modify it until you close it.

If you open a function library from the file system that is opened by another user, you are notified if changes are made by the other user, and given the option to accept or reject the changes made.

► **Tests open in read-only mode if:**

- The file you select is currently open by another user. In this case, you are notified that the file is already open, and by whom.
- The file was last saved in QuickTest 9.5 or earlier.
- The file is open in QuickTest, and you try to open the same file in the QuickTest Script Editor on the same computer, or vice versa.

► **Supported function library file types:**

- **.qfl**
- **.vbs**
- **.txt**

Tip: You can open and modify a function library even if it has an unsupported extension by opening it using the file mask ***.*** (All files) in the Open Function Library dialog box.

- **Only the script can be modified.** When working with tests in the QuickTest Script Editor, you cannot create new tests, or save existing tests with a new name. You can modify only the test script. This means that you cannot change information such as test settings, parameterization, data table values, and so on.
- **No syntax or spell check.** When you modify a test script or function library, make sure that you make all changes in the test script using the correct syntax, format, and spelling because the QuickTest Script Editor does not do this for you.
- **Earlier versions open in read-only mode.** Tests and function libraries that were last saved in QuickTest 9.5 or earlier open in read-only mode in the QuickTest Script Editor.

- If a test or function library is stored in the file system or in Quality Center 9.2, you can update it to the current version by opening and saving it in QuickTest. After you save the test, you will not be able to open it in earlier versions of QuickTest or the Script Editor.
- If a file is stored in Quality Center 10.00 or HP ALM, the administrator must update it to the current version using the QuickTest Professional Asset Upgrade Tool for Quality Center, which upgrades all tests in the project simultaneously. After a test is upgraded, you cannot open it in an earlier version of QuickTest or the Script Editor. If the test is saved in a version-control-enabled project, the test opens only in read-only mode. To modify it, you must open it in QuickTest.
- **The Resources and Dependencies Model (described on page 1651) is not supported for the QuickTest Script Editor.** This means that if a test contains a call to an action or function stored in Quality Center, and that action or function cannot be found, the QuickTest Script Editor does not try to locate the missing file. For example if a test contains a call to an action in Test B, and Test B was moved to a different folder in a Quality Center project or was renamed, the QuickTest Script Editor will not be able to locate the action.

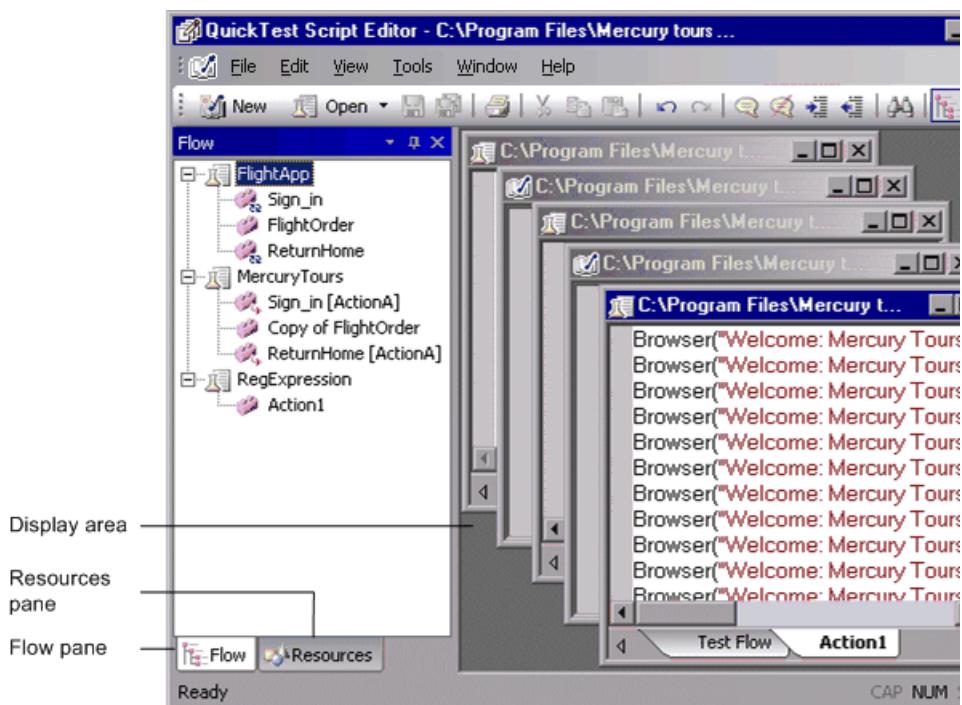
Reference

QuickTest Script Editor Window

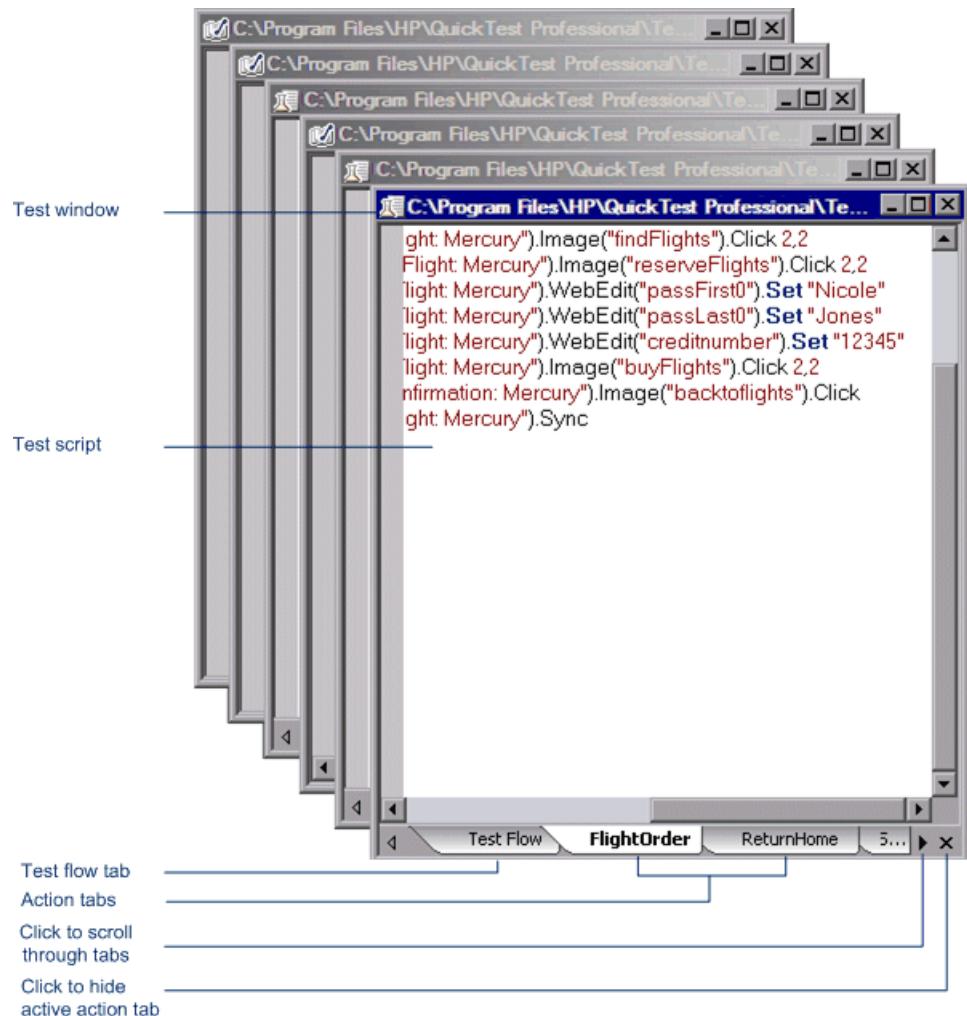
This window enables you to view and edit multiple test scripts and function libraries. The QuickTest Script Editor is comprised of:

- Display Area
- Flow Pane (Script Editor) (described on page 1582)
- Resources Pane (Script Editor) (described on page 1584)

When you open a test, the test and all its actions are displayed in the tree in the Flow pane, and the local actions and function libraries are displayed in the tree in the Resources pane.



The display area contains a separate window for each open test or function library, and each test window contains a tab for each local action open in the test.



To access	Start > Programs > HP QuickTest Professional > Tools > QuickTest Script Editor
Important information	<ul style="list-style-type: none"> ➤ The QuickTest Script Editor automatically adds a UTF-16 identifier to the start of each function library file that you save (either new or existing). ➤ Tests created in earlier versions of QuickTest open in read-only mode. For details, see "Considerations for Working with the QuickTest Script Editor" on page 1572. ➤ When you open a: <ul style="list-style-type: none"> ➤ Test. The test and all its actions are displayed in the tree in the Flow pane, and the local actions and function libraries are displayed in the tree in the Resources pane. ➤ Function library. The function library is displayed in the Opened Function Libraries node in the tree in the Resources pane.

User interface elements are described below:

UI Elements (A-Z)	Description
Display area	<p>Displays a window for each of the open tests and function libraries.</p> <p>For tests: Tabs at the bottom of a window enable you to navigate between the test flow and each of the actions.</p> <p> Hide Action. Closes the active tab in the display area. Note: The action is not closed, only hidden. Therefore you are not prompted to save any changes.</p> <p> Scroll Left / Scroll Right. Enables you to scroll between tabs to display any actions that are not currently visible.</p>
Flow Pane	Displays the flow of the action calls for each of the open tests.

UI Elements (A-Z)	Description
Resources Pane	Displays the open tests, its local actions and any function libraries associated with each test, as well as a list of all currently open function libraries.
<Status Bar>	<p>Left side: Indicates the status of the currently selected window, for example, Ready.</p> <p>Right side: Contains the following elements:</p> <ul style="list-style-type: none"> ➤  ALM/QC Connection. Double-click to open the HP ALM Connection dialog box, enabling you to connect to a Quality Center project. ➤ CAP. Indicates whether Caps Lock on the keyboard is on or off. ➤ NUM. Indicates whether Num Lock on the keyboard is on or off. ➤ SCRL. Indicates whether Scroll Lock on the keyboard is on or off.

Menu command and toolbar elements are described below:

UI Element	Command	Shortcut Key	Function
	File > New Function Library	CTRL+N	Creates a new function library.
	File > Open Test	CTRL+O	<p>Opens a dialog box enabling you to browse to and open a test.</p> <p>Tip: You can also drag files from the file system (Windows Explorer) to the Script Editor window.</p>

UI Element	Command	Shortcut Key	Function
	File > Open Function Library	CTRL+SHIFT+O	<p>Opens a dialog box enabling you to browse to and open a function library file.</p> <p>Tip: You can also drag files from the file system (Windows Explorer) to the Script Editor window.</p>
	File > Save	CTRL+S	Saves the active document.
	File > Save All		Saves all open documents.
	File > Print	CTRL+P	Prints the active document.
	File > <Recent Files>		Lists the recently viewed files.
	File > Exit		Closes the QuickTest Script Editor session.
	Edit > Cut	CTRL+X	Removes the selection from your document.
	Edit > Copy	CTRL+C	Copies the selection from your document.
	Edit > Paste	CTRL+V	Pastes the selection to your document.
	Edit > Delete	DELETE	Deletes the selection from your document.
	Edit > Undo	CTRL+Z	Reverses the last command or deletes the last entry you typed.
	Edit > Redo	CTRL+Y	Reverses the most recent operation of the Undo command.
	Edit > Comment Block	CTRL+M	Comments out the current row, or selected rows in the active document.

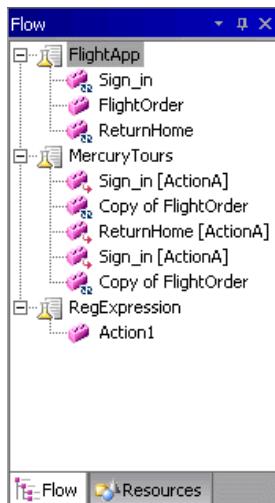
UI Element	Command	Shortcut Key	Function
	Edit > Uncomment Block	CTRL+SHIFT+M	Removes the comment formatting from the current or selected rows in the active document.
	Edit > Indent	TAB	Indents the step in the active document according to the tab spacing defined in the Editor Options dialog box.
	Edit > Outdent	BACKSPACE	Outdents the step in the active document according to the tab spacing defined in the Editor Options dialog box.
	Edit > Find	CTRL+F	Searches for a specified string.
	Edit > Replace	CTRL+H	Searches and replaces a specified string.
	View > Flow View		Displays or hides the Flow pane.
	View > Resources View		Displays or hides the Resources pane.
	View > Toolbars		Enables you to: ► Show or hide the Standard toolbar. ► Customize the display options. For details, see "Customize Dialog Box (Script Editor)" on page 1587.
	View > Status Bar		Displays or hides the status bar.

UI Element	Command	Shortcut Key	Function
	View > Themes		Enables you to select a theme to apply to your QuickTest window.
	Tools > Editor Options		Opens the Editor Options dialog box, enabling you to customize how tests and function libraries are displayed. For details, see the Editor Options dialog box.
	Tools > Quality Center Connection		Opens the Quality Center Connection dialog box, enabling you to connect to a Quality Center project.
	Window > Cascade		Displays the open documents cascaded.
	Window > Tile		<p>Displays the open documents horizontally one above the other.</p> <p>Tip: To display the open documents vertically, choose Window > Windows, select two or more windows in the Windows dialog box, and click Tile Vertically.</p>
	Window > Arrange Icons		Moves minimized windows to the bottom left corner of the display area.

UI Element	Command	Shortcut Key	Function
	Help > QuickTest Script Editor Help		Opens the Script Editor section of the <i>HP QuickTest Professional User Guide</i> Help.
	Help > About QuickTest Script Editor		Displays information about the installed version of QuickTest Script Editor and provides a link to the HP site.

Flow Pane (Script Editor)

This pane displays the test flow (action call flow) for each currently open test.



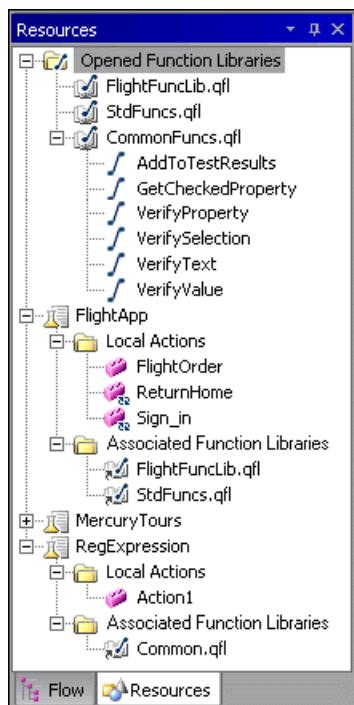
To access	View > Flow View
Important information	Each open test is displayed as a node in a tree, and each node contains the hierarchy of all the actions that were called in the test, including calls to local, reusable, and external actions.

User interface elements are described below:

UI Elements	Description
	<p>A test</p> <p>Shortcut menu:</p> <ul style="list-style-type: none"> ➤ Show. Brings the test to the front making it the active document. <p>Tip: Double-clicking a test also brings it to the front making it the active document.</p> <ul style="list-style-type: none"> ➤ Close. Closes the window and removes the test from the Flow pane. (If there are any unsaved changes, you are prompted to save them.) ➤ Properties. Opens the Test Properties dialog box, displaying the name and path of the test.
	<p>A call to a local action</p>
	<p>A call to an external action</p>
	<p>A call to a reusable action</p>
	<p>A call to an action that cannot be found (missing action)</p>
	<p>A looped action call, meaning a call to an action that was already called earlier in the test flow hierarchy</p>
	<p>Shortcut menu for all action types:</p> <ul style="list-style-type: none"> ➤ Show. Brings the action to the front making it the active document. For an external action, the test containing the called action is added to the tree in the Flow pane and Resources pane, and the selected action is displayed in a new test window in the display area. <p>Tip: Double-clicking an action does the same as above.</p> <ul style="list-style-type: none"> ➤ Go to Action Call. Brings the action to the front making it the active document and highlights the action call script line. ➤ Properties. Opens the Action Call Properties dialog box, displaying the name and path of the action, and indicating whether it is an external action. If the test was defined with a relative path in QuickTest, then the path is displayed as .\<name of action>.

Resources Pane (Script Editor)

This pane displays all the currently open tests and their resources (actions and associated function libraries).



To access	View > Resources View
Important information	Each test is displayed as a node in the tree, and each node contains the actions and function libraries associated with the test. All currently open function libraries, and their functions, are also displayed in a separate node at the top of the pane.

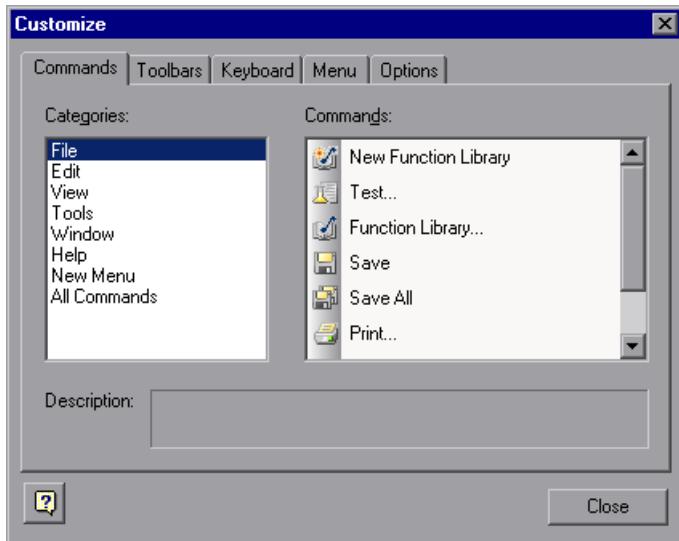
User interface elements are described below:

UI Elements	Description
	An open function library
	A link to a function library that is associated with a test
	<p>Shortcut menu for function libraries:</p> <ul style="list-style-type: none"> ➤ Show. Brings the function library to the front making it the active document. Tip: Double-clicking a function library also brings it to the front making it the active document. ➤ Close. Closes the window and removes it from the Flow pane. If there are any unsaved changes, you are prompted to save them. (Available only if the function library is currently open in the QuickTest Script Editor.) ➤ Properties. Opens the Function Library Properties dialog box, displaying its name and path.
	A public function defined in a function library
	A private function defined in a function library
	<p>Shortcut menu for all function types:</p> <ul style="list-style-type: none"> ➤ Go to Function Definition. Brings the function library containing the function to the front making it the active document, and highlights the first line in the function. Tip: Double-clicking a function does the same as above.
	<p>A test</p> <p>Shortcut menu:</p> <ul style="list-style-type: none"> ➤ Show. Brings the test to the front making it the active document. Tip: Double-clicking a test also brings it to the front making it the active document. ➤ Close. Closes the window and removes it from the Flow pane. ➤ Properties. Opens the Test Properties dialog box, displaying the name and path of the test.

UI Elements	Description
	A local action
	A reusable action
	<p>Shortcut menu for all action types:</p> <ul style="list-style-type: none">➤ Show. Brings the action to the front making it the active document. Tip: Double-clicking an action also brings it to the front making it the active document.➤ Properties. Opens the Action Properties dialog box, displaying the name and path of the action, and indicating whether it is a reusable action. Note: If the action was defined with a relative path in QuickTest, then the path is displayed as .<name of action or function library>.

Customize Dialog Box (Script Editor)

This dialog box enables you to customize Script Editor toolbars, menus, and other display options in a similar way to many other Windows applications. For example, you can choose whether to display line numbers, or change the font and color used to display the scripts.



To access	Right-click in the toolbar or menu bar and select Customize .
-----------	--

Tabs are described below:

Tab	Description
Commands	<p>Enables you to:</p> <ul style="list-style-type: none"> ➤ Add and move buttons and commands in the Script Editor toolbars and menus. ➤ Remove buttons and commands from the displayed toolbars and menus.
Toolbars	<p>Enables you to:</p> <ul style="list-style-type: none"> ➤ Select which of the available toolbars to display in the Script Editor window. ➤ Choose whether to display text labels for the toolbar buttons. ➤ Reset the toolbar display to the default.
Keyboard	<p>Enables you to:</p> <ul style="list-style-type: none"> ➤ Assign new keyboard shortcuts for toolbar and menu commands. ➤ Modify and remove existing shortcuts. ➤ Reset all of the keyboard shortcuts to the default.
Menu	<p>Enables you to:</p> <ul style="list-style-type: none"> ➤ Select which of the available menus to display in the Script Editor window. ➤ Select which of the commands to display in the context menus. ➤ Choose how the menus are animated, and whether they are displayed with a shadow. ➤ Reset the displayed menus to the default.
Options	<p>Enables you to:</p> <ul style="list-style-type: none"> ➤ Select whether to show tooltips for toolbar buttons. ➤ Select whether to show shortcut keys in the tooltips. ➤ Select whether to display toolbar buttons as large or small icons.

51

QuickTest Automation Scripts

This chapter includes:

Concepts

- QuickTest Automation Object Model Overview on page 1590
- When to Use QuickTest Automation Scripts on page 1592
- Application Object on page 1593
- QuickTest Automation Object Model Reference on page 1594
- Generated Automation Scripts on page 1594

Tasks

- How to Create a QuickTest Automation Script on page 1596
- How to Run Automation Scripts on a Remote Computer on page 1600

Concepts

QuickTest Automation Object Model Overview

You can use the QuickTest Professional automation object model to write scripts that automate your QuickTest operations. The QuickTest automation object model provides objects, methods, and properties that enable you to control QuickTest from another application.

Using the objects, methods, and properties exposed by the QuickTest automation object model, you can write scripts that configure QuickTest options and run tests instead of performing these operations manually using the QuickTest interface.

Automation scripts are especially useful for performing the same tasks multiple times or on multiple tests, or for quickly configuring QuickTest according to your needs for a particular environment or application.

What is Automation?

Automation is a Microsoft technology that makes it possible to access software objects inside one application from other applications. These objects can be created and manipulated using a scripting or programming language such as VBScript or VC++. Automation enables you to control the functionality of an application programmatically.

What is the QuickTest Automation Object Model?

An **object model** is a structural representation of software objects (classes) that comprise the implementation of a system or application. An object model defines a set of classes and interfaces, together with their properties, methods and events, and their relationships.

The **QuickTest automation object model** is a set of objects, methods, and properties that enable you to control essentially all of the configuration and run functionality provided via the QuickTest interface. Although a one-on-one comparison cannot always be made, most dialog boxes in QuickTest have a corresponding automation object, most options in dialog boxes can be set and/or retrieved using the corresponding object property, and most menu commands and other operations have corresponding automation methods.

You can use the objects, methods, and properties exposed by the QuickTest automation object model, along with standard programming elements such as loops and conditional statements to design your script.

Automation scripts are especially useful for performing the same tasks multiple times or on multiple tests, or for quickly configuring QuickTest according to your needs for a particular environment or application.

Example:

You can create and run an automation script from Microsoft Visual Basic that loads the required add-ins for a test, starts QuickTest in visible mode, opens the test, configures settings that correspond to those in the Options, Test Settings, and Record and Run Settings dialog boxes, runs the test, and then saves the test.

You can then add a simple loop to your script so that your single script can perform the operations described above for multiple tests.

You can also create an initialization script that opens QuickTest with specific configuration settings. You can then instruct all of your testers to open QuickTest using this automation script to ensure that all of your testers are always working with the same configuration.



When to Use QuickTest Automation Scripts

Creating a useful QuickTest automation script requires planning, design time, and testing. You must always weigh the initial investment with the time and human-resource savings you gain from automating potentially long or tedious tasks.

Any QuickTest operation that you must perform many times in a row or must perform on a regular basis is a good candidate for a QuickTest automation script.

The following are just a few examples of useful QuickTest automation scripts:

- **Initialization scripts.** You can write a script that automatically starts QuickTest and configures the options and the settings required for testing a specific environment.
- **Maintaining your tests.** You can write a script that iterates over your collection of tests to accomplish a certain goal. For example:
 - **Updating values.** You can write a script that opens each test with the proper add-ins, runs it in update run mode against an updated application, and saves it when you want to update the values in all of your tests to match the updated values in your application.
 - **Applying new options to existing tests.** When you upgrade to a new version of QuickTest, you may find that the new version offers certain options that you want to apply to your existing tests. You can write a script that opens each existing test, sets values for the new options, then saves and closes it.
 - **Modifying Actions and Action Parameters.** You can retrieve the entire contents of an action script, and add a required step, such as a call to a new action. You can also retrieve the set of action parameters for an action and add, remove, or modify the values of action parameters.
- **Calling QuickTest from other applications.** You can design your own applications with options or controls that run QuickTest automation scripts. For example, you could create a Web form or simple Windows interface from which a product manager could schedule QuickTest runs, even if the manager is not familiar with QuickTest.

Application Object

Like most automation object models, the root object of the QuickTest automation object model is the **Application** object.

The **Application** object represents the application level of QuickTest. You can use this object to return other elements of QuickTest such as the:

- **Test** object (which represents a test document)
- **Options** object (which represents the Options dialog box)
- **Addins** collection (which represents a set of add-ins from the Add-in Manager dialog box)

You can also use the **Application** object to perform operations like loading add-ins, starting QuickTest, opening and saving tests, and closing QuickTest.

Each object returned by the **Application** object can return other objects, perform operations related to the object and retrieve and/or set properties associated with that object.

Every automation script begins with the creation of the QuickTest **Application** object. Creating this object does not start QuickTest. It simply provides an object from which you can access all other objects, methods and properties of the QuickTest automation object model.

Note: You can also optionally specify a remote QuickTest computer on which to create the object (the computer on which to run the script). For more information, see **Running Automation Programs on a Remote Computer** in the **Introduction** section of the *QuickTest Automation Object Model Reference* in the *QuickTest Professional Help*.

QuickTest Automation Object Model Reference

The QuickTest Automation Object Model Reference is a Help file that provides detailed descriptions, syntax information, and examples for the objects, methods, and properties in the QuickTest automation object model.

You can open the *HP QuickTest Professional Automation Object Model Reference* from:

- QuickTest program folder (**Start > Programs > HP QuickTest Professional > Documentation > QuickTest Automation Reference**)
- Main QuickTest Help (**Help > HP QuickTest Professional Help > HP QuickTest Professional Advanced References > HP QuickTest Professional Automation Object Model**)

Generated Automation Scripts

The Properties pane of the Test Settings dialog box, the General pane of the Options dialog box, and the Object Identification dialog box each contain a **Generate Script** button. Clicking this button generates an automation script file (.vbs) containing the current settings from the corresponding dialog box.

You can run the generated script as is to open QuickTest with the exact dialog box configuration of the QuickTest application that generated the script, or you can copy and paste selected lines from the generated files into your own automation script.

For example, the generated script for the Options dialog box may look something like this:

```
Dim App 'As Application
Set App = CreateObject("QuickTest.Application")
App.Launch
App.Visible = True
App.Options.DisableVORecognition = False
App.Options.AutoGenerateWith = False
App.Options.WithGenerationLevel = 2
App.Options.TimeToActivateWinAfterPoint = 500
...
...
App.Options.WindowsApps.NonUniqueListItemRecordMode = "ByName"
App.Options.WindowsApps.RecordOwnerDrawnButtonAs = "PushButtons"
App.Folders.RemoveAll
```

For more information on the **Generate Script** button and for information on the options available in the Options, Object Identification, and Test Settings dialog boxes, see Chapter 7, "Configuring Object Identification", Chapter 45, "Global Testing Options", and Chapter 46, "Individual Test Settings."

Tasks

How to Create a QuickTest Automation Script

This task describes when and how to create automation scripts, and includes the following steps:

- "Prerequisites" on page 1596
- "Create the Application object" on page 1597
- "Reference the type library - optional" on page 1598
- "Write your automation script" on page 1599
- "Run your automation script" on page 1600

1 Prerequisites

- **Decide whether to use QuickTest Automation Scripts**

Creating a useful QuickTest automation script requires planning, design time, and testing. You must always weigh the initial investment with the time and human-resource savings you gain from automating potentially long or tedious tasks.

Any QuickTest operation that you must perform many times in a row or must perform on a regular basis is a good candidate for a QuickTest automation script.

- **Choose a language and development environment for designing and running Automation scripts**

You can write your QuickTest automation scripts in any language and development environment that supports automation. For example, you can use: VBScript, JavaScript, Visual Basic, Visual C++, or Visual Studio .NET.

For each language, there are a number of development environments available for designing and running your automation scripts.

2 Create the Application object

The procedure for creating the **Application** object differs slightly from language to language. Below are some examples for creating the QuickTest **Application** object and starting QuickTest in visible mode, using different programming languages. For conceptual details on the **Application** object, see "Application Object" on page 1593.

Visual Basic

The example below can be used only after setting a reference to the type library. If you are not working in a development environment that allows referencing type libraries create the **Application** object as described for VBScript below.

```
Dim qtApp As QuickTest.Application ' Declare the application object
Set qtApp = New QuickTest.Application ' Create the application object
qtApp.Launch ' Start QuickTest
qtApp.Visible = True ' Make it visible
```

VBScript

```
Dim qtApp
Set qtApp = CreateObject("QuickTest.Application") ' Create the application object
qtApp.Launch 'Start QuickTest
qtApp.Visible = True ' Make it visible
```

JavaScript

```
var qtApp = new ActiveXObject("QuickTest.Application"); // Create the application
object
qtApp.Launch(); // Start QuickTest
qtApp.Visible = true // Make it visible
```

Visual C++

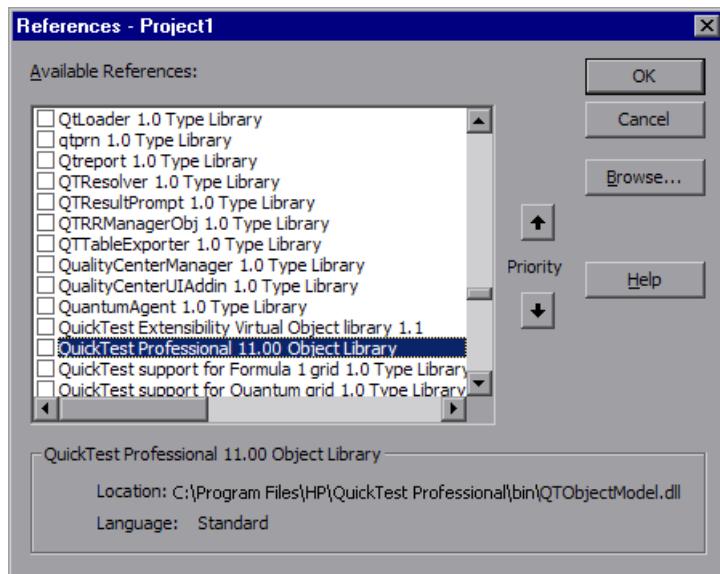
```
#import "QTObjectModel.dll" // Import the type library
QuickTest::_ApplicationPtr spApp; // Declare the application pointer
spApp.CreateInstance("QuickTest.Application"); // Create the application object
spApp->Launch(); // Launch the application
spApp->Visible = VARIANT_TRUE; // Make it visible
```

3 Reference the type library - optional

Some development environments support referencing a type library. A **type library** is a binary file containing the description of the objects, interfaces, and other definitions of an object model.

If you choose a development environment that supports referencing a type library, you can take advantage of features like Microsoft IntelliSense, automatic statement completion, and status bar help tips while writing your script. The QuickTest automation object model supplies a type library file named **QTObjectModel.dll**. This file is stored in <QuickTest installation folder>\bin.

If you choose an environment that supports it, be sure to reference the QuickTest type library before you begin writing or running your automation script. For example, if you are working in Microsoft Visual Basic, select **Project > References** to open the References dialog box for your project. Then select **QuickTest Professional <Version> Object Library** (where <Version> is the current installed version of the QuickTest automation type library).



4 Write your automation script

The structure for your script depends on the goals of the script. You may perform a few operations before you start QuickTest such as retrieving the associated add-ins for a test, loading add-ins, and instructing QuickTest to open in visible mode.

After you perform these preparatory steps, if QuickTest is not already open on the computer, you can open QuickTest using the **Application.Launch** method. Most operations in your automation script are performed after the Launch method.

For information on the operations you can perform in an automation program, see the online *HP QuickTest Professional Object Model Reference*. For more information on this Help file, see "QuickTest Automation Object Model Reference" on page 1594.

Tip: You can generate automation scripts from QuickTest that contain the settings for the Test Settings dialog box, the Options dialog box, and the Object Identification dialog box as they are set on your computer. You can then run each generated script as is to instruct QuickTest to open on other computers with the exact dialog box configuration defined in the generated script, or you can copy and paste selected lines from the generated files into your own automation script. For details, see "Generated Automation Scripts" on page 1594.

When you finish performing the necessary operations, or you want to perform operations that require closing and restarting QuickTest (such as changing the set of loaded add-ins), use the **Application.Quit** method.

5 Run your automation script

There are several applications available for running automation scripts. You can also run automation scripts from the command line using Microsoft's Windows Script Host.

For example, you could use the following command line to run your automation script:

```
WScript.exe /E:VBSCRIPT myScript.vbs
```

For details on running automation scripts on a remote computer, see "How to Run Automation Scripts on a Remote Computer" on page 1600.

How to Run Automation Scripts on a Remote Computer

By default, when you create an **Application** object in your automation script, it is created on your local computer (using your local copy of QuickTest Professional). You can also run automation scripts on a remote QuickTest computer. To do so, you must:

1 Set DCOM Configuration Properties on the Remote Computer

QuickTest automation enables QuickTest to act as a COM automation server. Therefore, to run a QuickTest automation script on a remote computer, you must ensure that the DCOM configuration properties for that computer give you the proper permissions to launch and configure the QuickTest COM server.

The procedure below describes the steps you need to perform on the remote computer to enable your automation script to run on that computer. Note that the DCOM Configuration Property the appearance and names of the dialog boxes and options mentioned here may vary depending on the computer's operating system.

To enable automation scripts to access a QuickTest computer remotely:

- a** On the computer where you want to run the automation script, select **Start > Run**. The Run dialog box opens.
- b** Enter **dcomcnfg** and click **OK**. The Distributed COM Configuration Properties dialog box or the Component Services window opens (depending on your operating system) and displays the list of COM applications available on the computer.
- c** Select **QuickTest Professional Automation** from the list and open the Properties dialog box for the application. (Click the **Properties** button or right-click and select **Properties**, depending on your operating system.)
- d** In the QuickTest Professional Automation Properties dialog box, click the **Security** tab.
- e** In the launch permissions section, select the custom option and click **Edit**.
- f** Use the **Add** and **Remove** options to select the network users or groups for which you want to allow or deny permission to launch QuickTest Professional via an automation script. When you are finished, click **OK** to save your settings.
- g** Repeat steps e and f for the configuration permissions section to select the users or groups who can modify QuickTest Professional configuration options via an automation script.
- h** In the QuickTest Professional Automation Properties dialog box, click the **Identity** tab and select the **interactive user** option.
- i** Click **OK** to save the QuickTest Professional Automation Properties settings.
- j** Click **OK** to close the Distributed COM Configuration Properties dialog box or the Component Services window.

2 Create an Application Object on the Remote Computer

After you set the necessary DCOM Configuration settings for a remote computer, you can specify that computer in your automation script.

In VBScript, you do this by specifying the computer name as the optional location argument of the CreateObject function. The computer name should be the same as the computer name portion of a share name. For example, to run an automation script on a computer called MyServer, you could write:

```
Dim qtApp  
Set qtApp = CreateObject("QuickTest.Application", "MyServer")
```

For information on the syntax for specifying the remote computer in another language you are using, see the documentation included with your development environment or the general documentation for the programming language.

Part XI

Working with Quality Center

Quality Center Integration

Note: Unless otherwise specified, references to **Quality Center** in this guide apply to all currently supported versions of Quality Center and HP ALM. Note that some features and options may not be supported in the specific edition of Quality Center or HP ALM that you are using.

For a list of the supported versions of Quality Center or HP ALM, see the *HP QuickTest Professional Product Availability Matrix*, available from the Documentation Library Home page or the root folder of the QuickTest Professional DVD.

For details on Quality Center or HP ALM editions, see the *HP Quality Center User Guide* or the *HP Application Lifecycle Management User Guide*.

This chapter includes:

Concepts

- Quality Center Integration Overview on page 1607
- Template Tests on page 1610
- Quality Center Test Run Preferences on page 1611
- QuickTest Remote Agent Preferences on page 1613
- Data Awareness in HP ALM on page 1613

Tasks

- How to Work with Tests in Quality Center on page 1618

- How to Data Drive a Test in HP ALM on page 1621
- How to Enable Quality Center to Run Tests on a QuickTest Computer on page 1627
- How to Use the Quality Center Connectivity Add-in on page 1628
- How to Create a Template Test on page 1629
- How to Create a Test in Quality Center Using a Template Test on page 1631
- How to View or Modify Remote Agent Settings on page 1633

Reference

- HP ALM Data Awareness - Task Breakdown on page 1634
- HP ALM Connection Dialog Box on page 1635
- Remote Agent Settings Dialog Box on page 1642

Troubleshooting and Limitations - Quality Center Integration
on page 1648

Concepts

Quality Center Integration Overview

QuickTest integrates with Quality Center, the HP centralized quality solution. Quality Center helps you maintain a project of all kinds of tests (such as QuickTest tests, business process tests, manual tests, tests created using other HP products, and so on) that cover all aspects of your application's functionality. Each test in your project is designed to fulfill a specified testing requirement of your application. To meet the goals of a project, you organize the tests in your project into unique groups.

Quality Center provides an intuitive and efficient method for scheduling and running tests, collecting results, analyzing the results, and managing test versions. It also features a system for tracking defects, enabling you to monitor defects closely from initial detection until resolution.

At its most basic level, integrating QuickTest with Quality Center enables you to store and access QuickTest tests and resource files in a Quality Center project, when QuickTest is connected to Quality Center.

This section also includes: "What is a Quality Center Project?" on page 1607

What is a Quality Center Project?

A Quality Center project is a database for collecting and storing data relevant to a testing process. For QuickTest to access a Quality Center project, you must connect to the local or remote Web server where Quality Center is installed.

When QuickTest is connected to Quality Center, you can:

- **Create tests and save them in your Quality Center project.**
- **View your QuickTest test scripts.** This can help you decide if you want to run a test as part of a test set. Note that the Test Flow in Quality Center and the Test Flow pane in QuickTest display only the actions that will be run when the currently selected test runs. This means that if a nested action is commented out, for example, that action is not displayed in Quality Center or in the QuickTest Test Flow pane. However, the action is still available from the Action toolbar in QuickTest, and you can uncomment it in the QuickTest Expert View when needed.
- **Run your tests and view the results in Quality Center.**
- **Associate tests with external files stored in the Test Resources module of a Quality Center project.**
- **Associate external files for all tests or for a single test.** For example, suppose you set the shared object repository mode as the default mode for new tests. You can instruct QuickTest to use a specific object repository stored in the Test Resources module in Quality Center.

For details on specifying external files for all tests, see Chapter 45, "Global Testing Options." For details on specifying external files for a single test, see Chapter 46, "Individual Test Settings."

- **Take advantage of all of the features provided with the Resources and Dependencies model.** For details, see "Resources and Dependencies Model" on page 1651.
- **Use the QCUtil object to access and use the full functionality of the Quality Center OTA (Open Test Architecture).** This enables you to automate integration operations during a run session, such as reporting a defect directly to a Quality Center database. For details, see the **Utility** section of the *HP QuickTest Professional Object Model Reference* and the Quality Center Open Test Architecture documentation.
- **Use the TDOTA object in your QuickTest automation scripts to access the Quality Center OTA.** For details, see the *QuickTest Professional Automation Object Model Reference* (**Help > QuickTest Professional Help > HP QuickTest Professional Advanced References > HP QuickTest Professional Automation Object Model**).

- Report defects to a Quality Center project either automatically as they occur, or manually directly from the QuickTest Run Results Viewer. For information on manually or automatically reporting defects to a Quality Center project, see "How to Manually Submit Defects to Quality Center" on page 1098 and "How to Automatically Submit Defects to a Quality Center Project" on page 1100.
-

Tip: You can save standalone, portable copies of tests stored in a Quality Center project when needed. For example, you may need to open or run a test while travelling because you do not have access to Quality Center. For details, see "Portable Copies of Tests" on page 395.

For details on working with Quality Center, see the HP ALM or Quality Center user guide.

Required Access Permissions

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

- Full read and write permissions to the Quality Center cache folder (located on the Quality Center client side)
 - Full read and write permissions to the QuickTest Add-in for ALM/QC installation folder
-

Tip: For information about the various QuickTest add-ins, see the *HP QuickTest Professional Add-ins Guide*.

Template Tests

Template tests serve as the basis for all QuickTest tests created in Quality Center. A **template test** is a QuickTest test that contains default test settings. For example, a template test might specify the QuickTest add-ins, associated function libraries, and recovery scenarios that are associated with a test. You can modify these test settings in the Test Settings dialog box (**File > Settings**) in QuickTest.

In addition to default test settings, a template test can also contain any comments or steps you want to include with all new QuickTest tests created in Quality Center. For example, you may want to add a comment notifying users which add-ins are associated with the template test, or you may want to add a step that opens a specific Web page or application at the beginning of every test. Any steps or comments you add to a template test are included in all new tests created in Quality Center that are based on that template test.

All template tests are saved in your Quality Center project (except for the default template test, which is located on the Quality Center client) and do not need to be copied to each user's local computer. This enables users to customize their local default template tests, if needed, and still have access to globally maintained template tests.

When tests based on a specific template test are run from Quality Center, QuickTest automatically loads the associated add-ins and applies the required settings, as defined in the test.

Default Template Test

- A default template test is installed on each Quality Center client when the QuickTest Professional Add-in for Quality Center is installed in the **<QuickTest Add-in for Quality Center folder>\bin\Templates** folder on your computer (for example: C:\Program Files\HP\QuickTest Add-in for ALM/QC\bin\Templates\Template11).
- When a Quality Center user creates a new QuickTest test in Quality Center, the default template test for the installed QuickTest version is automatically associated with the test unless the user selects another template test, as described in "How to Create a Template Test" on page 1629.

- The default template test is installed locally, therefore any changes you make to the template test are applied only to tests created on your computer (using the Quality Center client).

Working with New Template Tests

When you create new template tests, they are stored in your Quality Center project, making them available to all Quality Center users as the basis for new QuickTest tests created in that Quality Center project.

You create a template test by first creating a blank test in QuickTest with the required test settings. Then, in the **Test Plan** module of your Quality Center project, you browse to your QuickTest test and mark it as a **Template Test**. For details, see "How to Create a Template Test" on page 1629.

You can create multiple template tests, each for a specific testing purpose.

Example of Usage

You may want to create one template test for QuickTest tests that test Web applications with ActiveX controls, and another for QuickTest tests that test standard Windows applications. You would associate the ActiveX and Web Add-ins with the first template test. For the second template test, you would not associate any QuickTest add-ins at all, but you might specify the Windows application that you want to test. You could also make other modifications to the test settings for each of the template tests, as needed.



Quality Center Test Run Preferences

You can run QuickTest tests that are stored in a Quality Center database via:

- QuickTest
- Quality Center client that is installed on your computer
- A remote Quality Center client

You cannot run QuickTest tests from Quality Center if the QuickTest computer is logged off or locked.

When Quality Center runs your QuickTest test, it uses the associated add-ins list to load the proper add-ins for your test on the QuickTest computer. For details, see "Add-in Associations in Your Test" on page 1458 and "Template Tests" on page 1610.

You can instruct QuickTest to report a defect for each failed step when Quality Center test runs on your QuickTest computer. You can also submit defects to Quality Center manually from the QuickTest Run Results Viewer. For details, see "How to Manually Submit Defects to Quality Center" on page 1098.

Before you instruct a remote Quality Center client to run QuickTest tests on your computer, you must give Quality Center permission to use your QuickTest application (described in "How to Enable Quality Center to Run Tests on a QuickTest Computer" on page 1627).

You can also view or modify the QuickTest Remote Agent settings (described below).

Hidden mode

By default, QuickTest opens and runs in hidden mode when Quality Center activates it to run a test in a test set. This helps to improve performance. You can change this setting in the QuickTest Remote Agent. You can also instruct the Remote Agent to display a tooltip window indicating that QuickTest is running a Quality Center test in hidden mode. For details, see "QuickTest Remote Agent Preferences" on page 1613. While QuickTest is running in hidden mode, you can open QuickTest to view the steps being run in the Expert View by clicking the  button in the status bar.



QuickTest Remote Agent Preferences

When you run a QuickTest test from Quality Center, the QuickTest Remote Agent opens on the QuickTest computer where the test runs. The QuickTest Remote Agent determines how QuickTest behaves when a test is run by a remote application such as Quality Center.

You can open the Remote Agent Settings dialog box at any time to view or modify the settings that your QuickTest application uses when Quality Center runs a test on your computer. For details, see "Remote Agent Settings Dialog Box" on page 1642.



Data Awareness in HP ALM

Note: The data awareness feature is available when integrating QuickTest with HP ALM.

In HP ALM, you can upload and store your Microsoft Excel (.xls) test data resource files in the Test Resources module, enabling you to use the same data for multiple tests, or different data for each test run.

In HP ALM, you run **configurations** instead of standard tests. When working with data-driven QuickTest tests in HP ALM, each **configuration** is a QuickTest test that is set to run with a selected data resource file and optional data filter settings. The filter settings enable you to filter data by specified row numbers, by parameter text values, or both.

Note: You define and manage configurations only in HP ALM.

Example

Suppose you define three configurations in a single test to test an application that provides different solutions for Gold Card, Silver Card, and Bronze Card users. You could use the same data resource for each configuration by filtering the rows or text of the data resource to match the relevant user type. Similarly, if your data is stored in various .xls files, you might want to run the same test using a different data source each time. You would do this by associating a different data source with each configuration.

By managing your data as a resource in HP ALM, you can see at a glance which data resource files are associated with a particular test or configuration, and which tests or configurations are using a specific data resource. For more details, see "Advantages of Data Awareness in HP ALM" on page 1616.

How Does Data Awareness in HP ALM Work?

When you create a test in HP ALM (or save a test to an HP ALM project from QuickTest for the first time), a default **configuration** with the same name as the test is created simultaneously. You can view this configuration in the Configuration tab of the Test Plan module.

In the HP ALM Test Resources module, you create one or more data resources and upload one Microsoft Excel (.xls) file for each.

After you upload a data resource file to your HP ALM project, you can define the test-level configuration settings for a particular test. You do this in the Parameters tab of the Test Plan module by selecting the data resource file and mapping its column names to the data table parameters in your test. You can associate one data resource with the test-level configuration. By mapping the column names to data table parameters, you enable QuickTest to identify and use the correct parameter value during a run session.

Note:

- Specifying a test-level data resource in the Parameters tab of the HP ALM Test Plan module overrides the **Data Table** settings in the Resources pane of the QuickTest Test Settings dialog box.
 - The data table parameters in your QuickTest test must be defined in the **Global** sheet.
-

In the Configurations tab of the Test Plan module, you can create as many additional configurations as needed. By default, every configuration uses the test-level configuration settings unless you override the data resource.

If you associate a configuration with the data resource defined in the Parameters tab, you do not need to map the data table parameters to column names. You can, however, modify the filter settings by applying text filters to any parameter, and/or selecting the rows on which the configuration can run.

If you associate a different data resource with a configuration, or you select to override the test data resource and then select the same data resource file that you selected in the Parameters tab (which is the same as selecting a different data resource), you need to map the data table parameters to the column names in the .xls file. You can also apply filter settings, as described previously.

Note: If the data table parameter names in the **Global** sheet and the column names in the associated data resource are identical, you do not need to perform any mappings.

Next, in the Test Lab module, you can run any or all of the configurations defined for a test (or associated with a requirement) by adding them to a test set. When you run a configuration containing steps that use data table parameters, the parameter values are retrieved from the data resource files according to the settings defined for that configuration.

For a task describing how to work with configurations, see "How to Data Drive a Test in HP ALM" on page 1621.

For a chart listing modules and tabs in which you perform tasks in HP ALM, see "HP ALM Data Awareness - Task Breakdown" on page 1634.

This section also includes:

- "Advantages of Data Awareness in HP ALM" on page 1616
- "Considerations for Data Awareness in HP ALM" on page 1617

Advantages of Data Awareness in HP ALM

Test Reuse. You can use the same test to test various scenarios, each time with a different set of data. You can do this associating a different data resource with each configuration, by associating a single data resource with different configurations and then filtering each configuration, or by using a combination of both.

Data Reuse. You can associate the same data resource with multiple tests or configurations.

Ease of Management. All of your resources are stored in one central location.

Visibility. You can see which tests are using a particular data table, and you can see which data table each test is using.

Requirements Coverage. You can cover all of your testing requirements by linking them to specific configurations. This enables you, for example, to use a single test to cover multiple requirements by associating different configurations in the same test with each requirement.

Resources and Dependencies Model. By storing your data in the Test Resources module, you can take advantage of additional HP ALM integration features, such as:

- Version control and baselines
- Asset Comparison Tool and Asset Viewer
- Sharing data resources across HP ALM projects



Considerations for Data Awareness in HP ALM

Consider the following when managing your data resources in HP ALM:

- Only Microsoft Excel .xls files are supported as data resources.
- If the data table parameter names in your test's **Global** sheet are identical to the column names in the associated data resource file, you do not need to perform any mapping.
- If you modify a data table parameter or a column name in the associated data resource after they are mapped to each other, you must update the mappings accordingly. For details, see "How to Data Drive a Test in HP ALM" on page 1621.

Tasks

How to Work with Tests in Quality Center

This task includes the following steps:

- "Prerequisites for Windows Vista, Windows 7, and Windows Server 2008, and Windows Server 2008 R2 Users" on page 1618
- "Connect to a Quality Center project" on page 1619
- "Create or open a test" on page 1620
- "Create a template test - Optional" on page 1620
- "Create a test using a template test - Optional" on page 1620
- "Save your test in the Quality Center project" on page 1620
- "Set QuickTest Remote Agent preferences" on page 1620
- "Run your test in the Quality Center project" on page 1620
- "Manage versions of your project using version control - Optional" on page 1620
- "Disconnect from the Quality Center project" on page 1620

Prerequisites for Windows Vista, Windows 7, and Windows Server 2008, and Windows Server 2008 R2 Users

The security settings in the following operating systems may prevent you from connecting to a Quality Center project:

- Windows Vista
- Windows 7
- Windows Server 2008
- Windows Server 2008 R2

This can occur when the UAC (User Account Control) option is set to ON, and you have never connected to a Quality Center project.

To connect to Quality Center for the first time, you must disable the UAC option. After you successfully connect to Quality Center, you can turn the UAC option on again. Thereafter, you should be able to connect to Quality Center, as needed.

To enable QuickTest to connect to a Quality Center project:

For Microsoft Windows Vista and Windows Server 2008:

- 1** Log in as an administrator.
- 2** From the Control Panel, select **User Accounts > Change Security Settings**, and clear the **Use User Account Control (UAC) to help protect your computer** check box.
- 3** Connect to Quality Center, as described in "HP ALM Connection Dialog Box" on page 1635.
- 4** Select the **Use User Account Control (UAC) to help protect your computer** check box and click **OK** to turn the UAC option on again.

For Microsoft Windows 7 and Windows Server 2008 R2:

- 1** Log in as an administrator.
- 2** From the Control Panel, select **User Accounts > User Accounts > Change User Account Settings**.
- 3** In the User Account Control Settings window, move the slider to **Never notify**.
- 4** Connect to Quality Center, as described in "HP ALM Connection Dialog Box" on page 1635.
- 5** In the User Account Control Settings window, restore the slider to its previous position to turn the UAC option on again.

Connect to a Quality Center project

For details, see "HP ALM Connection Dialog Box" on page 1635.

Create or open a test

For details, see "How to Perform File Operations on Test Files" on page 398.

Create a template test - Optional

Perform this step to create a QuickTest template test that has pre-defined test settings. This template test can then be used when creating new QuickTest tests in Quality Center. For details, see "How to Create a Template Test" on page 1629.

Create a test using a template test - Optional

Perform this step to create a QuickTest test that has the pre-defined test settings defined in a template test. For details, see "How to Create a Test in Quality Center Using a Template Test" on page 1631.

Save your test in the Quality Center project

For details, see "How to Perform File Operations on Test Files" on page 398.

Set QuickTest Remote Agent preferences

Use the Remote Agent to view or modify the settings that your QuickTest application uses when Quality Center runs a test on your computer. For details, see "Remote Agent Settings Dialog Box" on page 1642.

Run your test in the Quality Center project

For details, see "QuickTest Run Sessions" on page 1063 and "Run Dialog Box: Results Location Tab (For Tests Stored in Quality Center)" on page 1075.

Manage versions of your project using version control - Optional

For details, see "Managing Versions of Assets in Quality Center Overview" on page 1696.

Disconnect from the Quality Center project

For details, see "HP ALM Connection Dialog Box" on page 1635.

How to Data Drive a Test in HP ALM

This task describes how to data drive a QuickTest test in HP ALM.

This task provides a general overview of the steps involved in data driving a test in HP ALM. After you are familiar with the steps, you can perform many of them in the order you choose. Some steps may not be necessary in all cases.

This task includes the following steps:

- "Prerequisites" on page 1621
- "Create a data resource file in your HP ALM project" on page 1622
- "Specify a default data table resource for all new test configurations" on page 1622
- "Define your test configurations" on page 1624
- "Link your configurations to requirements to create requirements coverage (Optional)" on page 1626
- "Run your test configurations" on page 1626

1 Prerequisites

- Connect to HP ALM, as described in "HP ALM Connection Dialog Box" on page 1635.
- Make sure that:
 - You have a test that uses data table parameters from the **Global** sheet. For details, see "How to Define Values for Your Step Arguments" on page 491.
 - Your test is saved in your HP ALM project. For details on creating tests and saving them to HP ALM, see "Methodologies for Creating Tests" on page 384 and "Save Test Dialog Box" on page 412.

2 Create a data resource file in your HP ALM project

In the Test Resources module:

- a Expand the Resources tree and select the required node.
- b Select **Resources > New Resource** to add a resource under that node.
- c In the New Resource dialog box:
 - In the **Type** list, select **Data table**.
 - In the **Name** box, enter a name for the data resource, for example, the name of the Microsoft Excel (**.xls**) file you plan to use.
 - Fill in the remaining fields (optional) and click **OK** to close the dialog box.
- d In the Resource Viewer tab, click **Upload Resource**. Then browse to and upload the relevant **.xls** file.

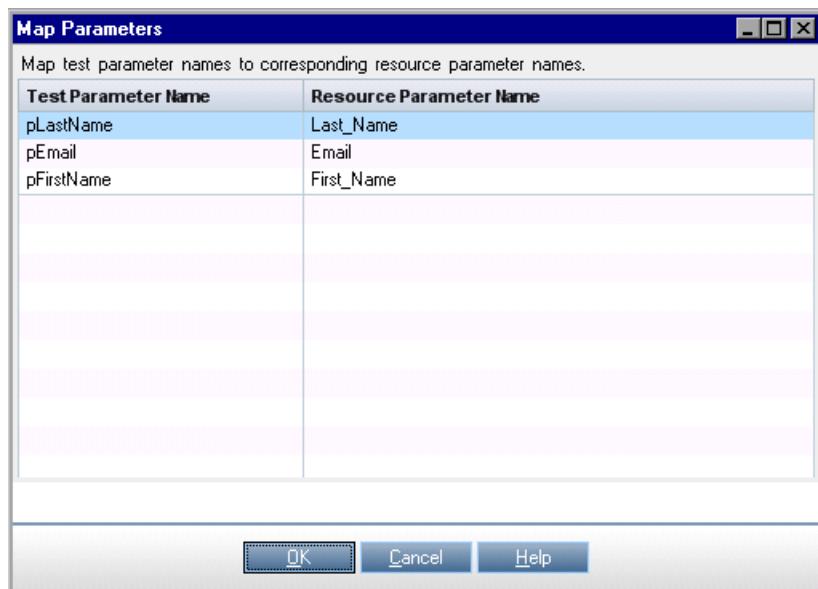
Tip: You can convert an internal data table from an open QuickTest test to an uploadable data resource file by right-clicking the data table, selecting **File > Export**, saving the data table to the file system as an **.xls** file, and then uploading it as described above.

3 Specify a default data table resource for all new test configurations

- a In the Parameters tab of the Test Plan module, select the data table resource you want to use as the default for all test configurations.
If you do not specify a data table resource, the data table specified in the Resources pane of the QuickTest Test Settings dialog box is used instead.

Note: In this tab, only the Parameter Name column is relevant for QuickTest tests.

- b** Click the **Map Parameters**  button. In the Map Parameters dialog box, map the data table parameters (column headings) to the test parameters by entering the matching data table parameter names in the **Resource Parameter Name** column, as shown in the example below.



All new configurations will use these default mappings unless you specify otherwise in the Data tab of the Configurations tab. For details, see step 4, **Define your test configurations**, below.

4 Define your test configurations

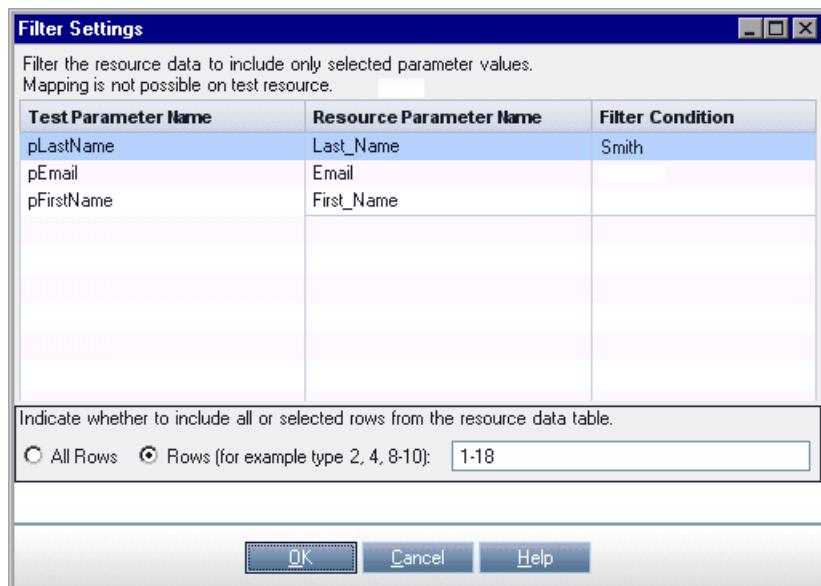
Define test configurations for various run sessions. For each configuration, you specify whether to use the default resource file that you specified in step 3 above, or whether to use a different data resource file.

- a** In the HP ALM Test Plan module, browse to and select the test to which you want to associate your data table resource. For details, see the *HP Application Lifecycle Management User Guide*.
- b** Click the **Configurations** tab. A default configuration is displayed in the grid. This configuration was created when your test was added to the HP ALM project. For details on configurations, see "Data Awareness in HP ALM" on page 1613.
- c** In the bottom pane of the Configurations tab, click the **Data** tab.
- d** In the Data tab:
 - Select the **Override test data resource** check box to select a different data resource file from the Test Resources module, or leave the check box blank to use the default resource file you selected in the Parameters tab in step 3 above.
 - In the **Data Resource** box, browse to and select the relevant data resource file to associate with this configuration. (Relevant only if you selected the **Override test data resource** check box.)
 - Click the **Data Resource Settings** button and, in the Filter Settings dialog box:
 - Map the data table parameters from your test to the column headers in the data table file. (Relevant only if you selected a different data resource file in the previous step.)
 - Apply filter conditions (text strings), as needed. You can apply one filter condition to each parameter.
 - Specify the rows on which to run iterations. For example, if you run a configuration named Gold, and users of this type are listed in rows 2-114, then specify these rows only.

Note: If you apply filter conditions and specify rows, AND logic is used, meaning that the parameter value must equal the filter text value and the parameter value must be located in one of the specified rows.

Example:

The **pLastName** data table parameter is mapped to **Last_Name** and is filtered to include only parameter text values that equal Smith. The configuration is set to run only on rows 1-18 in the **.xls** file you uploaded as the data resource.



For details on this dialog box, see the *HP Application Lifecycle Management User Guide*.

5 Link your configurations to requirements to create requirements coverage (Optional)

If you want to make sure that your requirements are fully covered, link them to configurations. This enables you to select configurations to run based on requirement coverage when you plan your run session.

- a In the Test Plan module, click the **Req Coverage** tab.
- b Click the **Select Req** button. The Requirement Tree tab is displayed in the right pane.
- c From the Requirement Tree tab, select a requirement to add to the Req Coverage grid. When you add the requirement, the Add Advanced Coverage dialog box opens.
- d Select the test configurations that cover this requirement.

6 Run your test configurations

You run configurations from HP ALM.

- a In QuickTest, make sure that in the **Tools> Options > Run** tab, **Allow other HP products to run tests and components** is selected.
- b In the HP ALM Test Lab module, select or create a test set. For details, see the *HP Application Lifecycle Management User Guide*.
- c In the right pane, select the **Execution Grid** tab.
- d Click the **Select Tests** button to display the **Test Plan Tree** and **Requirements Tree** tabs in the right pane.

- e Do one of the following to select the configurations to run:
 - From the Test Plan Tree tab, select a test to add to the Execution Grid. When you add the test, all of its configurations are added to the Execution Grid. (The test itself is not added to the Execution Grid because HP ALM runs configurations—not tests.)
 - Below the Test Plan Tree tab, expand the **Test Configurations** pane, and add the specific configurations that you want to run to the Execution Grid.
 - From the Requirements Tree tab, select a requirement to add to the Execution Grid. When you add the requirement, all of its linked configurations are added to the Execution Grid.
 - Below the Requirements Tree tab, expand the coverage pane, and select a test to add to the Execution Grid. When you add the test, all of its configurations are added to the Execution Grid. (The test itself is not added to the Execution Grid because HP ALM runs configurations—not tests.).
- f Click the **Run** button to run the selected configurations.
- g After the run session, click **Launch Report** button in the Last Run Report tab to view the results.

How to Enable Quality Center to Run Tests on a QuickTest Computer

To enable Quality Center to run QuickTest tests:

In QuickTest, select the **Allow other HP products to run tests and components** option in the Run pane of the Options dialog box (**Tools > Options > Run** node). For details, see "Run Pane (Options Dialog Box)" on page 1447.

For security reasons, remote access to your QuickTest application is not enabled by default. This option enables Quality Center (or other remote access clients) to open and run QuickTest tests.

To enable full access to QuickTest tests from Quality Center:

Make sure that the QuickTest Professional Add-in for ALM/QC is installed on your Quality Center client computer. This enables you to view the test and view the run results in the Run Results viewer. For details on this add-in, go to the QuickTest Professional Add-in screen (accessible from the main Quality Center screen).

For limitations for specific operating systems, see "Troubleshooting and Limitations - Quality Center Integration" on page 1648.

How to Use the Quality Center Connectivity Add-in

Install the Quality Center Connectivity Add-in

You integrate QuickTest with Quality Center using the Quality Center Connectivity Add-in. This add-in is installed on your QuickTest computer automatically when you connect QuickTest to Quality Center using the Quality Center Connection dialog box.

You can also install it manually from the Quality Center Add-ins page by choosing **Help > Add-ins Page > HP Quality Center Connectivity** in Quality Center.

View the version of the Quality Center Connectivity Add-in that is currently installed on your computer



Select **Help > About** and then click the **Product Information** button. For details, see "About QuickTest Professional Dialog Box" on page 117.

How to Create a Template Test

This task describes how to create a template test. For an overview, see "Working with New Template Tests" on page 1611.

1 In QuickTest:

- a Open QuickTest with the required add-ins loaded. For details on loading QuickTest add-ins, see the section on loading QuickTest add-ins in the *HP QuickTest Professional Add-ins Guide*.

Note: Make sure that the add-ins are actually installed on the QuickTest computer on which the test will eventually run. Otherwise, when the test is run, QuickTest will not be able to load the required add-ins and the test may fail.

- b Define the required settings in the Test Settings dialog box (**File > Settings**). For details, see "Test Settings Overview" on page 1456.
- c If you want to include comments or steps in all tests based on this template test, add them.
- d  Click the **Save** button or select **File > Save** to save the test. The Save Test to Quality Center dialog box opens. Save the test to your Quality Center project using a descriptive name that clearly indicates its purpose. For details, see "Save Test Dialog Box" on page 412.

Tip:

- In the Quality Center test plan tree (Test Plan module), you may want to create a special folder for your template tests. This will enable other users to quickly locate the relevant template test when they create new QuickTest tests in Quality Center.
 - When you save the test in QuickTest, you should apply a descriptive name that clearly indicates its purpose. For example, if the template test is to be used to associate the ActiveX and Web Add-ins with a new test, you could call it ActiveX_Web_Addins_Template.
-

2 In Quality Center:

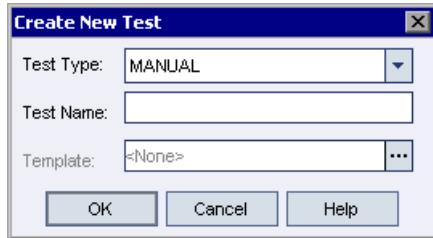
-  **a** Open the project in Quality Center, click the **Test Plan** button on the sidebar to open the Test Plan module, and browse to the test you saved in step d.
- b** Right-click the test and select **Mark as Template Test**. The test is converted to a template test.

How to Create a Test in Quality Center Using a Template Test

This task describes how to create a test in Quality Center using a template test as its basis.



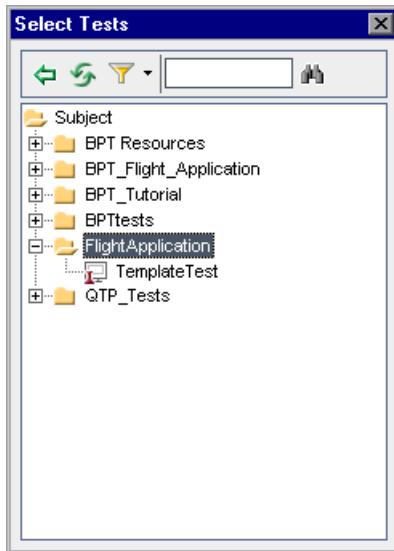
- 1 In Quality Center, click the **Test Plan** button on the sidebar to open the Test Plan module.
- 2 In the test plan tree, select a folder.
- 3 Click the **New Test** button, or select **Test > New Test**. The Create New Test dialog box opens.



Note: The **Template** box is displayed only if the **QuickTest Professional Add-in for Quality Center** is installed on your computer. If the **Template** box is not displayed, you must install the **QuickTest Professional Add-in for Quality Center** from the QuickTest Professional DVD or from the More Quality Center Add-ins page (opened from the Quality Center Options window, or from **Help > Add-ins Page**).

- 4 From the **Test Type** list, select **QUICKTEST_TEST**.
- 5 In the **Test Name** box, type a name for the test start using English (Roman) letters, numbers, and underscores (if needed). For a list of naming conventions, see "Naming Conventions" on page 1779.
- 6 Click the **Template** box browse button. The Select Tests dialog box opens.

- 7 Expand the folder containing your template test.



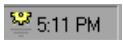
- 8 Select the template test on which to base your new test and click the **Add** button. The Select Tests dialog box closes and the template test you selected is displayed in the **Template** box (in the Create New Test dialog box).
- 9 In the Create New Test dialog box, click **OK**. The new test is created with the test settings defined in the template test.
- 10 The new test is displayed in the Test Plan tree under the subject folder you selected.

Note: If the Required Fields dialog box opens, set the required values and click **OK**. For details, see the HP ALM or Quality Center administrator guide.

- 11 Continue creating the test. For details on creating tests in Quality Center, see the HP ALM or Quality Center user guide.

How to View or Modify Remote Agent Settings

This task describes how to view or modify the settings in the Remote Agent Settings dialog box.



- 1** Select **Start > Programs > HP QuickTest Professional > Tools > Remote Agent**. The Remote Agent opens and the **Remote Agent** icon is displayed in the task bar tray.
- 2** Right-click the **Remote Agent** icon and select **Settings**. The Remote Agent Settings dialog box opens. (See "Remote Agent Settings Dialog Box" on page 1642 for an image.)
- 3** View or modify the settings in the dialog box. For details, see "Remote Agent Settings Dialog Box" on page 1642.
- 4** Click **OK** to save your settings and close the dialog box.
- 5** Right-click the **Remote Agent** icon and select **Exit** to end the Remote Agent session.

Reference

HP ALM Data Awareness - Task Breakdown

The following table lists where to perform specific data awareness tasks in HP ALM.

Module	Tab	Task
Test Plan	Parameters	<p>Define the test-level configuration for the current test:</p> <ul style="list-style-type: none"> ▶ Specify a data resource file. ▶ Map the data table parameters used in your test steps to the column names in your data resource file.
	Configurations	<p>Create additional configurations. (Optional)</p> <p>Data tab:</p> <ul style="list-style-type: none"> ▶ Select a different data resource. (Optional) ▶ Specify the data resource settings: <ul style="list-style-type: none"> ▶ Specify the rows to be used during a run session. ▶ Specify any text parameters values filters. (Optional) ▶ If you select a different data resource (by overriding the data resource defined in the Parameters tab), map the data table parameters used in your test steps to the column names in your data resource file.
	Req Coverage	<p>If you add requirement coverage to a test that contains multiple configurations, specify which configuration(s) to use for coverage.</p> <p>(Can also be done from the Requirements module)</p>

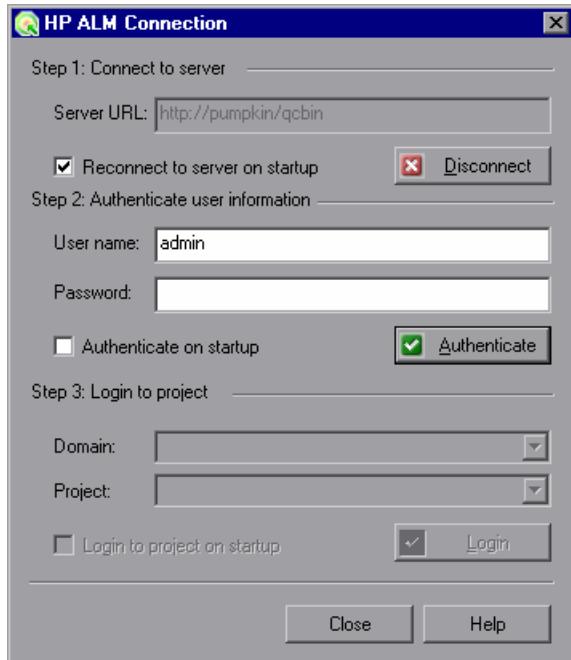
Module	Tab	Task
Test Resources	Resource Viewer	<ul style="list-style-type: none">➤ Create a data resource.➤ Upload a Microsoft Excel (.xls) file to use as the data resource.
Test Lab	Execution Grid	<ul style="list-style-type: none">➤ Add all of the configurations in a test to a test set.➤ Add only specific configurations to a test set.➤ Add all of the configurations contained in a tests that cover specific requirements to a test set.➤ Add only the configurations associated with a specific requirement to a test set.

HP ALM Connection Dialog Box

This dialog box enables you to connect or disconnect QuickTest to or from a project in any supported version of HP ALM or Quality Center.



After you perform step 1, the dialog box expands to include the remaining connection fields.



To access	Use one of the following: ► Select File > ALM/QC Connection . ► Double-click the ALM/QC icon in the status bar.  Ready
------------------	---

Important information	<ul style="list-style-type: none">➤ 1st time connection. The first time you connect your computer to an HP ALM or Quality Center server, you must connect as a user with administrator privileges.➤ Connecting to different HP ALM or Quality Center servers. You can simultaneously connect your Web browser to multiple HP ALM clients and one Quality Center 9.2 or 10.00 client simultaneously. While these clients are open, you can connect QuickTest to the Quality Center 9.2 or 10.00 client that is currently open on your computer. However, if you want to connect QuickTest to an HP ALM client, you must first close the Quality Center 9.2 or 10.00 client.➤ Windows Vista, Windows 7, Windows Server 2008, and Windows Server 2008 R2 Users. See "Guidelines for Windows Vista, Windows 7, Server 2008, and Server 2008 R2 Users" on page 1640.➤ Connect. The connection process has two stages. First, you connect QuickTest to a local or remote HP ALM or Quality Center server. This server handles the connections between QuickTest and the HP ALM or Quality Center project. Next, you log in and choose the project you want QuickTest to access. The project stores tests and run session information for the application you are testing. Projects are password protected, so you must provide a user name and a password.➤ Disconnect. You can disconnect QuickTest from an HP ALM or Quality Center project or from an HP ALM or Quality Center server at any time. If you disconnect QuickTest from an HP ALM or Quality Center server without first disconnecting from a project, the QuickTest connection to that project database is automatically disconnected. If an HP ALM or Quality Center test or shared file (such as a shared object repository or Data Table file) is open when you disconnect from HP ALM or Quality Center, then QuickTest closes it.
------------------------------	---

Relevant tasks	To view the current connection, point to the ALM/QC icon on the status bar. A tooltip displays the server name and project to which QuickTest is connected.
See also	"Quality Center Integration Overview" on page 1607

User interface elements are described below:

UI Elements (A-Z)	Description
Authenticate / Change User	<p>Authenticates your user information against the HP ALM or Quality Center server.</p> <p>Note: After your user information has been authenticated, the edit boxes in the Authenticate user information area are displayed in read-only format. The Authenticate button changes to a Change User button.</p> <p>Tip: You can log in to the same HP ALM or Quality Center server using a different user name by clicking Change User, and then entering a new user name and password and clicking Authenticate again.</p>
Authenticate on Startup	Instructs QuickTest to automatically authenticate your user information against the HP ALM or Quality Center server every time you open QuickTest.
Close	Closes the HP ALM Connection dialog box.
	Note: The dialog box does not close automatically. You must click this button to close the dialog box.
Connect / Disconnect	Connects or disconnects QuickTest to or from the selected HP ALM or Quality Center server. (After you successfully connect to a server, the button changes to Disconnect .)
Domain	<p>The domain that contains the HP ALM or Quality Center project.</p> <p>Note: Only those domains to which you have permission to connect are displayed.</p>

UI Elements (A-Z)	Description
Login / Logout	<ul style="list-style-type: none"> ➤ Login. Logs into the selected domain and project using the current user information. (After you successfully log into a project, the button changes to Logout.) ➤ Logout. Logs out of the selected domain and project. <p>Note: You must click Close to close the dialog box after logging into or out of a project.</p>
Login to project on startup	Instructs QuickTest to automatically log into the selected project every time you open QuickTest.
Password	<p>Your HP ALM or Quality Center password.</p> <p>Note: To enter a password in any CJK (Chinese, Japanese, Korean) language, copy/paste the password into the edit box. (Windows does not support typed CJK characters in password fields.)</p>
Project	<p>The HP ALM or Quality Center project with which you want to work.</p> <p>Note: Only those projects for which you are a defined user are displayed.</p>
Reconnect to Server on startup	Instructs QuickTest to automatically reconnect to the HP ALM or Quality Center server every time you open QuickTest.
Server URL	<p>The URL address of the Web server where HP ALM or Quality Center is installed.</p> <p>You can choose a server that is accessible via a Local Area Network (LAN) or a Wide Area Network (WAN).</p> <p>Note: You can connect to any currently supported version of HP ALM or Quality Center. For a list of supported versions, see the <i>HP QuickTest Professional Product Availability Matrix</i>, available from the Documentation Library Home page or the root folder of the QuickTest Professional DVD.</p>
User name	Your HP ALM or Quality Center user name.

Guidelines for Windows Vista, Windows 7, Server 2008, and Server 2008 R2 Users

The security settings in the following operating systems may prevent you from connecting to an HP ALM or Quality Center project:

- Windows Vista
- Windows 7
- Windows Server 2008
- Windows Server 2008 R2

This can occur when the UAC (User Account Control) option is set to ON, and you have never connected to an HP ALM or Quality Center project.

To connect to HP ALM or Quality Center for the first time, you must disable the UAC option. After you successfully connect to HP ALM or Quality Center, you can turn the UAC option on again. Thereafter, you should be able to connect to HP ALM or Quality Center, as needed.

To enable QuickTest to connect to an HP ALM or Quality Center project:

For Microsoft Windows Vista and Windows Server 2008:

- 1** Log in as an administrator.
- 2** From the Control Panel, select **User Accounts > Change Security Settings**, and clear the **Use User Account Control (UAC) to help protect your computer** check box.
- 3** Connect to HP ALM or Quality Center, as described above.
- 4** Select the **Use User Account Control (UAC) to help protect your computer** check box and click **OK** to turn the UAC option on again.

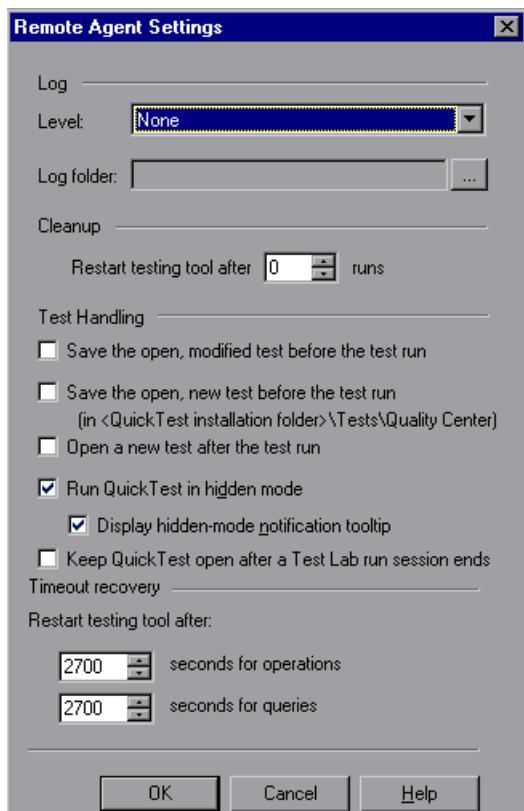
For Microsoft Windows 7 and Windows Server 2008 R2:

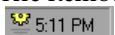
- 1** Log in as an administrator.
- 2** From the Control Panel, select **User Accounts > User Accounts > Change User Account Settings**.
- 3** In the User Account Control Settings window, move the slider to **Never notify**.

- 4** Connect to HP ALM or Quality Center, as described above.
- 5** Return to the User Account Control Settings window, and restore the slider to its previous position to turn the UAC option on again.

Remote Agent Settings Dialog Box

This dialog box enables you to view or modify the settings that QuickTest uses when Quality Center runs a QuickTest test on your computer.



To access	<p>1 Select Start > Programs > HP QuickTest Professional > Tools > Remote Agent. The Remote Agent opens and the Remote Agent icon  is displayed in the task bar tray.</p> <p>2 Right-click the Remote Agent icon and select Settings. The Remote Agent Settings dialog box opens.</p>
Relevant tasks	"How to View or Modify Remote Agent Settings" on page 1633

User interface elements are described below:

UI Elements	Description
Level	<p>The level of detail to include in the log that is created when Quality Center runs a QuickTest test or business process test.</p> <ul style="list-style-type: none"> ➤ None. (Default) No log is created. ➤ Low. The log lists any Quality Center-QuickTest communication errors. ➤ Medium. The log includes Quality Center-QuickTest communication errors and information on other major operations that result in Quality Center-QuickTest communication. ➤ High. The log includes all available information related to Quality Center-QuickTest communications.
Log folder	<p>The folder path for storing the log file. Required if a log type is specified in the Level option.</p>
Restart testing tool after _____ runs	<p>For QuickTest tests, restarts QuickTest after Quality Center completes the specified number of test runs. When QuickTest restarts, it continues with the next test in the test set.</p> <p>You may want to use this option to maximize available memory.</p> <p>If you do not want QuickTest to restart during a test set run, enter 0 (default).</p>
Save the open, modified test before the test run	<p>If an existing (named) test or is open in QuickTest when the Remote Agent begins running a test, this option instructs QuickTest to save any unsaved changes to the open test or.</p> <p>Note: This option is not relevant for function libraries. Therefore, if an existing (named) function library is open in QuickTest when the Remote Agent begins running a test, the function library is not saved.</p>

UI Elements	Description
Save the open, new test before the test run	<p>If a new (untitled) test is open in QuickTest when the Remote Agent begins running a test, the test is saved in:</p> <p><QuickTest installation folder>\Tests\Quality Center with a sequential test name.</p>
Open a new test after the test run	<p>By default, the last test run by the Remote Agent stays open in QuickTest when it finishes running the test. However, while it remains open, it is locked to other users. You can select this option to ensure that the last test that Quality Center runs on your computer is closed, and a blank test is open instead.</p> <p>If you select this option, a new, blank test opens after each test run, even if Quality Center is running multiple tests on your computer as part of a test set.</p>

UI Elements	Description
	<p>Run in hidden mode</p> <p>Specifies whether to run QuickTest in hidden (silent) mode when you run a test set from the Test Lab module in Quality Center. By default, this option is selected.</p> <p>Display hidden-mode notification tooltip: If this check box is selected, the Remote Agent displays a tooltip window when QuickTest runs a Quality Center test in hidden mode. You can click the tooltip to display QuickTest during the test set run. By default, this option is selected.</p> <p>Notes:</p> <ul style="list-style-type: none">➤ Clicking the notification tooltip clears the Run QuickTest in hidden mode check box and QuickTest will run in normal mode. You can run QuickTest in hidden mode again by reselecting Run QuickTest in hidden mode before the next test set run.➤ When running in hidden mode, QuickTest can be optionally redisplayed at the end of each test or at the end of the test set. This behavior is configured in Quality Center Site Administration using the SUPPORT_TESTSET_END parameter. For details, see the section on setting Quality Center configuration parameters in the Quality Center or HP ALM administrator guide.

UI Elements	Description
Keep QuickTest open after a Test Lab run session ends	<p>By default, when HP ALM opens QuickTest on a remote computer during a test set run (or when it runs selected tests or configurations from the Test Lab module), it closes QuickTest at the end of that Test Lab run session. This ensures that the QuickTest license is released at that point and made available for other QuickTest users.</p> <p>Selecting this option causes QuickTest to stay open on your computer (and to continue using the QuickTest license) after a Test Lab run session ends.</p> <p>Note: The behavior described above is relevant only when QuickTest is opened from an HP ALM server that has the SUPPORT_TESTSET_END parameter set to Y. (Y is the default setting.)</p> <p>If QuickTest is opened from an earlier version of Quality Center (or the above-mentioned parameter is set to N), this option is ignored and QuickTest always remains open at the end of the run session.</p> <p>For more details on the SUPPORT_TESTSET_END parameter, see the section on setting Quality Center configuration parameters in the <i>HP Application Lifecycle Management Administrator Guide</i>.</p>

UI Elements	Description
Restart testing tool after	<p>Restarts QuickTest if there is no response after the specified number of seconds for:</p> <ul style="list-style-type: none">➤ Operations. QuickTest operations such as Open or Run.➤ Queries. Standard status queries that remote applications perform to confirm that the application is responding (such as the Quality Center get_status query). <p>The default value for both options is 2700 seconds (45 minutes). However, while QuickTest operations may take a long time between responses, queries usually take only several seconds. Therefore, you may want to set different values for each of these options.</p> <p>Note: If a function library with unsaved changes is open in QuickTest, QuickTest prompts you to save it. If the function library is not saved within 10 seconds, QuickTest restarts and any unsaved changes are lost.</p>

Troubleshooting and Limitations - Quality Center Integration

This section describes troubleshooting and limitations for working with QuickTest tests and Quality Center. For information on issues related to working with QuickTest and business component assets in Quality Center, see the *HP Quality Center Readme* or *HP Application Lifecycle Management Readme*.

- If QuickTest is installed on one of the operating systems listed in the table below, you must perform prerequisites before you run QuickTest tests remotely from Quality Center.

Operating System	What you should do
<ul style="list-style-type: none"> ➤ Windows XP Service Pack 2 ➤ Windows 2003 Server 	<p>You must first change DCOM permissions and open firewall ports. For details, see the <i>HP QuickTest Professional Installation Guide</i>.</p>
<ul style="list-style-type: none"> ➤ Windows Vista ➤ Windows Server 2008 ➤ Windows 7 ➤ Windows Server 2008 R2 	<ol style="list-style-type: none"> 1 Change DCOM permissions and open firewall ports. For details, see the <i>HP QuickTest Professional Installation Guide</i>. 2 Run RmtAgentFix.exe from the <QuickTest installation>\bin folder, or use the Additional Requirements Utility, which you open by selecting Start > Programs > HP QuickTest Professional > Tools > Additional Installation Requirements. This is due to a problem with opening DCOM permissions on Windows Vista, Windows Server 2008, Windows 7 and Windows Server 2008 R2. 3 Disable User Account Control (UAC) in Windows before you first connect to Quality Center. For details, see "Troubleshooting and Limitations - QuickTest Program Management" on page 121.

- Before opening or running a test from Quality Center, you must open the current installation of QuickTest at least once. Otherwise, Quality Center may not be able to open QuickTest.

- Renaming a QuickTest test from Quality Center may cause the test to behave unexpectedly.

Workaround: To rename a test, open it in QuickTest and rename it using the **Save As** option. If the test has already been renamed from Quality Center, use the rename option again to restore the old name, and then use the **Save As** option in QuickTest. Renaming a QuickTest test parameter from QuickTest will cause any run-time parameter values already set for this parameter in Quality Center to be lost.

- The security settings in Windows Vista, Windows Server 2008, Windows 7 and Windows Server 2008 R2 may prevent you from performing a QuickTest Professional-related installation, such as a patch installation, or connecting to a Quality Center project (either directly or from QuickTest Professional). This can occur when the UAC (User Account Control) option is set to ON, and you have not yet connected to a Quality Center project (if relevant).

Workaround: Temporarily turn off the UAC option. For details, see "Troubleshooting and Limitations - QuickTest Program Management" on page 121.

After disabling the UAC option as described above, perform the required installation or connect to Quality Center as usual. When you are finished, you can turn the User Account Control (UAC) option on again. Hereafter, you should be able to connect to Quality Center, as needed.

- Quality Center is not Unicode compliant. Therefore:
 - When working with tests stored in Quality Center, you should not use Unicode values (such as the name of the test, the name of an application area, the default value of a test or action parameter, method argument values, and so forth).
 - Data that is sent to QuickTest from Quality Center (such as values for test or action parameters) is not Unicode compliant.
 - QuickTest results containing Unicode characters may appear corrupted in the Quality Center result grid. You can, however, open and view results containing Unicode characters in the QuickTest Run Results Viewer.

For additional information on QuickTest Professional Unicode issues, see "Troubleshooting and Limitations - Multilingual Applications" on page 1206.

- Renaming a QuickTest test from Quality Center may cause the test to behave unexpectedly.

Workaround: To rename a test, open it in QuickTest and rename it using the **Save As** option. If the test has already been renamed from Quality Center, use the rename option again to restore the old name, and then use the **Save As** option in QuickTest. Renaming a QuickTest test parameter from QuickTest will cause any run-time parameter values already set for this parameter in Quality Center to be lost.

Resources and Dependencies Model

Note: The references to Quality Center in this chapter apply to Quality Center 10.00 and HP ALM. Note that some features and options may not be supported in the Quality Center or HP ALM edition you are using. For information on Quality Center or HP ALM editions, see the *HP Quality Center User Guide* or the *HP Application Lifecycle Management User Guide*.

This chapter includes:

Concepts

- ▶ Resources and Dependencies Model Overview on page 1652
- ▶ Asset Dependencies - Advantages on page 1653

Reference

- ▶ Relative Paths and Quality Center on page 1656
- ▶ Resources and Dependencies Model Terminology on page 1657
- ▶ Quality Center Resources-Related User Interface on page 1659

Troubleshooting and Limitations - Resources and Dependencies
on page 1666

Concepts

Resources and Dependencies Model Overview

QuickTest enables you to use the Resources and Dependencies model to fully integrate your QuickTest tests into Quality Center projects.

Note: Before you read this section, you may want to familiarize yourself with the Resources and Dependencies Model Terminology (described on page 1657).

- **Replaces the use of attachments with linked QuickTest assets, for example, tests and actions can be linked with function libraries and shared object repositories, respectively.** You store your tests in the Test Plan module, and you store your resource files in the Test Resources module. When you associate a resource file to a test, these assets become linked. Linking assets improves runtime performance by decreasing download time. (Using attachments instead of resources increases download time from Quality Center 10.00 and HP ALM.) Linking assets also helps to ensure that the relationships between dependent assets are maintained.
- **Supports versioning for tests and resource files.** You can create versions of these assets in QuickTest or in Quality Center, and you can manage asset versions in Quality Center. For more information, see "Version Control in Quality Center 10.00 or HP ALM" on page 1695.
- **Supports baselines for tests and resource files.** You can view baseline history in QuickTest, and you can view and manage baselines in Quality Center. For more information, see "Baseline History Dialog Box" on page 1712.

- **Enables you to view and compare your QuickTest assets in both Quality Center and QuickTest.** You can use the Asset Comparison Tool to compare versions of individual QuickTest assets and the Asset Viewer for viewing an earlier version of a QuickTest asset. Both of these viewers are available in Quality Center and in QuickTest. For more information, see "Viewing and Comparing Versions of QuickTest Assets" on page 1669.
- **Enables you to import and share assets across Quality Center projects.** For more information, see the Quality Center or HP ALM user guide.

For more information about the advantages of working with this model, see "Asset Dependencies - Advantages" on page 1653.

Asset Dependencies - Advantages

When you associate a **dependent** resource file with a test, the assets become integrally linked, and these links can be viewed in Quality Center (in the Dependencies tab of various modules) and—for external actions—in QuickTest (in the Action Properties dialog box).

Assets stay linked

When you move a test, or rename a test or action, a folder, or a resource, dependent assets are automatically updated to reflect the change. This helps ensure that there are no missing resources during a run session.

Resource files are all stored in one Quality Center module

Resource files are stored in the Test Resources module, enabling you to manage your resources in one central location, and to view at a glance which tests are using each resource file. For more information on the Test Resources module, see the Quality Center or HP ALM user guide.

Using resources stored in the Test Resources module improves runtime performance

Tests open and run faster when the associated resource files are stored in the Test Resources module (instead of being stored as attachments to tests in the Test Plan module).

Version control can also be applied to resource files

When version control is enabled for a project, all of the assets can be checked into the version control database. For more information, see "Version Control in Quality Center 10.00 or HP ALM" on page 1695.

You can create, view, compare, and run baselines

In Quality Center, you can create baselines that capture the developmental stage for each asset, view and compare these read-only baseline "snapshots", and run baselines from a project. In QuickTest, you can view and compare baselines. For more information, see "Baseline History Dialog Box" on page 1712.

You can share assets with other projects and synchronize them

You can copy assets from other projects. This enables you to reuse your existing assets instead of creating new assets whenever you create a new project. For example, you can create a "template" set of assets to use as a basis for new projects.

You can synchronize these assets in both projects when changes are made, or you can customize your assets to suit the unique needs of each development project. For more information, see the sections on importing and synchronizing libraries in the Quality Center or HP ALM user guide.

Deleting assets is easier

When you delete an asset (a reusable action or associated resource file), a warning message informs you if the asset is used by other tests (or more than once in the current test). This message contains a **Details** section that lists the tests that are associated with this asset or contain calls to this action so you can modify the tests, as needed. This helps you manage your tests and action calls so that tests do not inadvertently fail.

You can verify which tests are associated with specific resources and vice versa

In the Quality Center Test Plan module, you can highlight a test and, in the Dependencies tab, see which assets are using the test, and see which assets the test is using. Similarly, in the Quality Center Test Resources module, you can highlight a resource file and see the assets with which it is associated.

For details, see "Dependencies Tab" on page 1660.

You can verify which tests contain calls to an action

You can view a list of the tests that contain calls to a particular action by focusing on the action and opening the Used By tab in the Action Properties dialog box. (Right-click an action in the Test Flow pane and select **Action Properties**.) For more information, see "Used By Tab (Action Properties Dialog Box)" on page 571.

Reference

Relative Paths and Quality Center

Resource files and actions that are associated with tests using a relative path are not considered dependencies. To ensure that your resource files are recognized as dependencies, they must be saved in the Test Resources module in Quality Center, and they must be associated using the full Quality Center path. This enables you to benefit from the features provided by the Resources and Dependencies Model, as described in "Asset Dependencies - Advantages" on page 1653.

Despite this, there may be cases in which you may want to use a relative path. For example, if your application is released in different languages, you may want to use a relative path when associating shared object repositories with your tests, as this enables you to use the same test with localized shared object repositories.

Limitations - Relative Paths and Quality Center

- Run-time performance times are slower.
- Dependency information for these assets is not displayed in:
 - The Using and Used By grids in the Dependencies tab in Quality Center.
See: "Dependencies Tab" on page 1660
 - The Used By tab of the Action Properties dialog box in QuickTest.
See: "Used By Tab (Action Properties Dialog Box)" on page 571
 - The message box that opens when you delete an asset that is associated with other assets.
See: "Asset Dependencies - Advantages" on page 1653
- Quality Center does not verify that these assets are included during the baseline verification process.
See: "Viewing Baseline History" on page 1701

- When opening or running tests from a baseline, any associated external action or resource file that is associated via a relative path but is **not** included in the baseline is considered a missing resource. This may cause a test run to fail. (Note that the baseline version of an associated asset is used if the asset associated via a relative path **is** included in the baseline.) See: "Viewing Baseline History" on page 1701
- When using the Asset Comparison Tool to view a test, you cannot drill down to view assets that are associated via a relative path.
See: "Asset Comparison Tool and Asset Viewer - Overview" on page 1670



Resources and Dependencies Model Terminology

Term	Description
Asset	<p>Any QuickTest testing document or resource file, including:</p> <ul style="list-style-type: none">➤ tests➤ actions➤ function libraries➤ shared object repositories➤ recovery scenarios➤ data table files➤ environment variable files <p>Note: In Quality Center, QuickTest assets are referred to by the more general term entities.</p>

Term	Description
Resource	<p>Any asset stored in the Quality Center Test Resources module that is used by a test. For example, a test may contain calls to functions in associated function libraries, and it may reference test objects stored in shared object repositories that are associated with the test. Resources include:</p> <ul style="list-style-type: none"> ▶ function libraries ▶ shared object repositories ▶ recovery scenarios ▶ data table files ▶ environment variable files <p>Note: In some cases, a resource can be used by another resource. For example, a recovery scenario can use functions in a function library.</p>
Dependency	<p>The linked relationship between a resource or external action and a particular test. Associated resource files and actions are linked to each test that uses these resources or calls these actions.</p> <p>In some cases, a resource can also be linked to another resource. For example, a recovery scenario can call functions in a function library.</p> <p>Assets are considered dependencies if they are associated using absolute paths, and they are stored in the following modules:</p> <ul style="list-style-type: none"> ▶ Test Plan module: tests ▶ Test Resources module: function libraries, shared object repositories, recovery scenarios, data table files, environment variable files <p>Note: Tests stored in the Unattached folder in the Test Plan module are not considered dependencies because they are not associated with any test.</p>
Configuration	<p>A test that is set to run from Quality Center with an optional data resource file and optional data filter settings. A QuickTest test can contain one or more configurations.</p> <p>For details, see "Data Awareness in HP ALM" on page 1613.</p>

Quality Center Resources-Related User Interface

When you create a Quality Center project in your Quality Center server, the QuickTest tests that you create in this project are saved to the Test Plan module. You save your resource files to the Test Resources module. By associating resource files with your tests, they become linked dependencies.

This section provides a general overview of the tabs that are relevant for QuickTest tests. For information on using any of these tabs, see the relevant section in the Quality Center or HP ALM user guide.

This section includes (in alphabetical order):

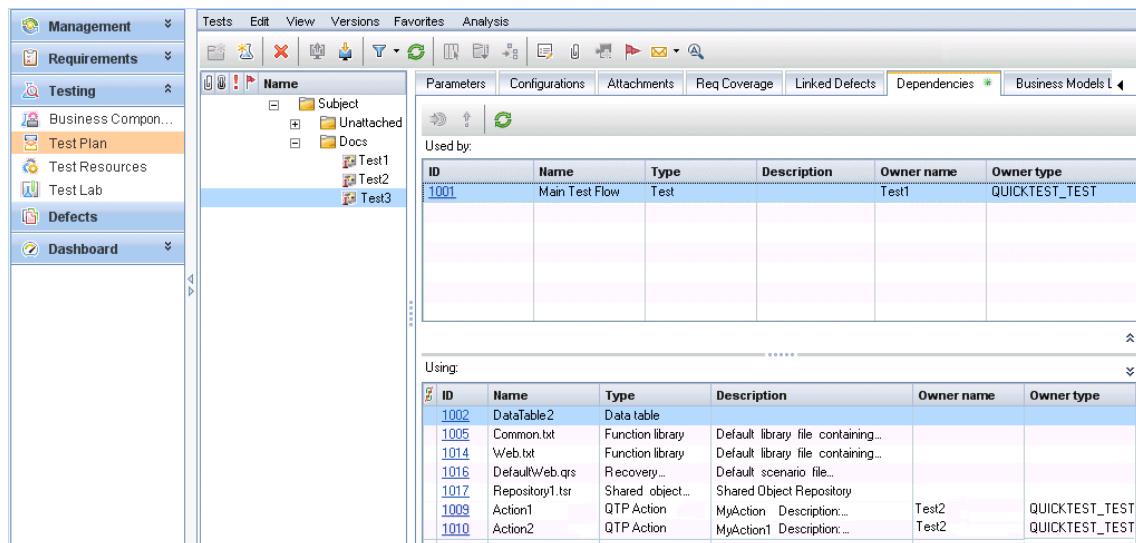
- "Dependencies Tab" on page 1660
- "History Tab" on page 1665
- "Libraries Module" on page 1666

Dependencies Tab

This tab displays the relationship between a selected asset, such as a test, and the assets with which it is associated. You use the Dependencies tab to see at a glance which resources are used by a particular asset, and which asset is using a particular resource. This information is displayed in the Using Grid and Used By Grid in the Dependencies tab in the Test Plan and Test Resources modules.

Usage Example

Suppose you want to modify the objects in a shared object repository. You can navigate to the shared object repository in the Test Resources module to view a list of the tests with which it is associated. This enables you to determine which assets this resource file is used by and helps you to analyze the impact that a proposed change may make to the dependent assets.



The screenshot shows the QTP Test Plan interface. The left sidebar has categories like Management, Requirements, Testing, Test Plan (selected), Test Resources, Test Lab, Defects, and Dashboard. The main area has tabs: Tests, Edit, View, Versions, Favorites, Analysis. Below these are toolbars for creating, deleting, and modifying objects. The 'Dependencies' tab is currently selected. A tree view on the left shows a folder named 'Subject' containing 'Unattached' and 'Docs' subfolders, with three files: 'Test1', 'Test2', and 'Test3'. The 'Used by:' grid below shows one entry for 'Test3'. The 'Using:' grid below shows multiple entries, including 'DataTable2', 'Common.txt', 'Web.txt', 'DefaultWeb.qrs', 'Repository1.tsr', 'Action1', and 'Action2'. The 'Using:' grid has columns: ID, Name, Type, Description, Owner name, and Owner type.

ID	Name	Type	Description	Owner name	Owner type
1001	Main Test Flow	Test		Test1	QUICKTEST_TEST

ID	Name	Type	Description	Owner name	Owner type
1002	DataTable2	Data table	Default library file containing...		
1005	Common.txt	Function library	Default library file containing...		
1014	Web.txt	Function library	Default scenario file...		
1016	DefaultWeb.qrs	Recovery...	Shared Object Repository		
1017	Repository1.tsr	Shared object...			
1009	Action1	QTP Action	MyAction Description:...	Test2	QUICKTEST_TEST
1010	Action2	QTP Action	MyAction1 Description:...	Test2	QUICKTEST_TEST

To access	In Quality Center, the Dependencies tab is available from the following modules: ► Test Plan module ► Test Resources module
Important information	For information on using this tab, see the Quality Center or HP ALM user guide.
Relevant Tasks	In QuickTest, you can view Used By information for actions in the Action Properties dialog box. For more information, see "Used By Tab (Action Properties Dialog Box)" on page 571. QuickTest also displays Used By information when you try to delete a dependent asset, so that you can determine how the change might affect associated assets before you delete the asset.

QuickTest-specific user interface elements are described below:

Used By Grid

This grid lists the assets that are using the asset currently selected in the tree. Suppose you are looking at the Used By grid for a shared object repository. The Used By grid lists all of the tests that are associated with the selected shared object repository. This list indicates the assets that will be affected if you modify or delete the asset selected in the tree.

Column	Description
ID	A unique numeric ID assigned automatically by Quality Center. If the ID is a link, you can click it to jump to that asset in Quality Center. Example: Suppose you are looking at the Used By grid for a specific function library in the Test Resources module. You can click the ID link to jump to the test with which it is associated. (The link takes you to the Test Plan module.)
Name	The name of the asset that is using the asset selected in the tree, for example, the name of a test or action that is using the asset selected in the tree. QuickTest-related owner names include: <ul style="list-style-type: none"> ➤ Main Test Flow. Indicates the test container called by the top-level action in the currently selected test in the Test Plan module. When Main Test Flow is displayed, the Owner Type is Test. ➤ Action<#>. Indicates the internal name of the action that is called by an action in the currently selected test in the Test Plan module. Action<#> refers to the sequential number of the action when it was created. Action<#> is displayed when the Owner Type is QTP Action. Note: Action<#> is displayed even if an action was renamed in the test. ➤ The actual name of the asset if the asset is not an action.
Type	The type of asset that is using the asset selected in the tree, for example Test , QTP Action , or QUICKTEST_TEST .

Column	Description
Description	<p>The description specified in the Details tab for the asset listed in the Name column.</p> <ul style="list-style-type: none"> ▶ If the Type is QTP Action, displays the actual name of the action as shown in QuickTest (for example, if the action was renamed, the user-defined name is displayed) and displays its description, if any. ▶ If the Type is QUICKTEST_TEST or Test, this cell is empty.
Owner Name	<p>The name of the asset that owns the asset listed in the Name column. For example, if the asset listed in the Name column is an action, then the Owner Name is the name of the test in which that action is stored.</p>
Owner Type	<p>The type of asset that owns the asset listed in the Name column. QuickTest-related owner types include:</p> <ul style="list-style-type: none"> ▶ QUICKTEST_TEST. A QuickTest test in the Test Plan module. ▶ QTP Action. An action that is part of a test in the Test Plan module.

Using Grid

This grid lists all of the dependencies that the selected asset is using. For example, suppose you are looking at a test. You can see all of the external actions called by the test, all of the shared object repositories containing test objects used by the test, function libraries containing functions called by the test, and so on.

Column	Description
ID	A unique numeric ID assigned automatically by Quality Center.
Name	<p>The name of the associated asset that the selected asset uses, for example, the name of the shared object repository, data table resource, or function library. QuickTest-related names include:</p> <ul style="list-style-type: none"> ➤ Action<#>. Indicates the internal name of the action that is called by an action in a test in the Test Plan module. Action<#> refers to the sequential number of the action when it was created. Action<#> is displayed when the Related Type is QTP Action. Note: Action<#> is displayed even if an action was renamed in the test. ➤ The actual name of the asset if the asset is not a test.
Type	The type of associated asset that the selected asset uses, for example, QTP Action , Data table , Function library , and Shared object repository .
Description	<p>The description of the associated asset that the selected asset is using, if any.</p> <p>If the Type is QTP Action, displays the name of the action as shown in QuickTest and displays its description, if any.</p>

Column	Description
Owner Name	The name of the asset that owns the asset listed in the Name column. For example, if the asset listed in the Name column is an action, then the Owner Name is the name of the test in which that action is stored.
Owner Type	The type of asset that owns the asset listed in the Name column, for example, QUICKTEST_TEST .

History Tab

This tab enables you to:

- View version information for a selected file.
- View baseline information for a selected file.
- View and compare file versions.
- View the baseline in which a version is stored (if applicable).
- Check out an earlier version of a file if you want to roll back to that version. (When you check the file back into the version control database, that version becomes the current version.)

To access	<p>In Quality Center, the History tab is available from the following modules:</p> <ul style="list-style-type: none"> ➤ Test Plan module ➤ Test Resources module <p>Note: The History tab is located in the pane on the right side of the window. You may need to scroll to the right to display it.</p>
Important information	For information on using this tab, see the Quality Center or HP ALM user guide.

Relevant tasks	In QuickTest, you can also view version history and baseline history by selecting one of the following: ➤ File > ALM/QC Version Control > Version History ➤ File > ALM/QC Version Control > Baseline History
See also	➤ "Version History Dialog Box" on page 1709 ➤ "Viewing Baseline History" on page 1701

Libraries Module

This module enables you to:

- **Create, view, and compare baselines.** For more information, see "Viewing Baseline History" on page 1701 and the sections describing baselines in the Quality Center or HP ALM user guide.
- **Import assets from other Quality Center projects.** This enables you to create a complete copy of the assets that are included in a baseline in another project in any accessible domain. For more information, see the Quality Center or HP ALM user guide.

To access	In the Quality Center sidebar, under Management select Libraries .
Important information	For information on using this tab, see the Quality Center or HP ALM user guide.

Troubleshooting and Limitations - Resources and Dependencies

This section describes troubleshooting and limitations for resources and dependencies.

- When you save a resource to Quality Center (either from QuickTest or using the **Upload** option from the Quality Center Test Resources module), and the resource file has a comma in the file name, the resource appears to be saved successfully, but the file is not actually uploaded to the Quality Center server.

- If you insert a call to an external action that is associated with a data table, and that data table was previously renamed or moved in the Test Resources module of Quality Center 10.00 or HP ALM, QuickTest tries to locate the data table in its original location.

Workaround: Save the test, close it, and reopen it.

Viewing and Comparing Versions of QuickTest Assets

Note: The references to Quality Center in this chapter apply to Quality Center 10.00 and HP ALM. Note that some features and options may not be supported in the Quality Center or HP ALM edition you are using. For information on Quality Center or HP ALM editions, see the *HP Quality Center User Guide* or the *HP Application Lifecycle Management User Guide*.

This chapter includes:

Concepts

- Asset Comparison Tool and Asset Viewer - Overview on page 1670

Tasks

- How to Open the QuickTest Asset Comparison Tool on page 1673
- How to Open the QuickTest Asset Viewer on page 1676
- How to Work with the Asset Comparison Tool and Asset Viewer on page 1679

Reference

- Asset Comparison Tool on page 1681
- Asset Viewer on page 1690

Troubleshooting and Limitations - Asset Comparison Tool on page 1693

Concepts



Asset Comparison Tool and Asset Viewer - Overview

An **asset** is a QuickTest testing document (such as a test) or any resource file that is used by a QuickTest testing document (such as a function library, a shared object repository, a data table, or a recovery scenario). The Asset Comparison Tool and Asset Viewer enable you to view and compare versions of a particular asset.

Using these tools, you can:

View any version of an asset using the Asset Viewer

For details, see "Asset Viewer" on page 1690.

Compare two versions of an asset using the Asset Comparison Tool

For details, see "Asset Comparison Tool" on page 1681.

Drill down to view or compare versions

Drilling down enables you to:

- View or compare versions of an **integral element**. An integral element is a resource file that is a part of the test (not saved as an external resource), such as a local object repository. When you check in a test, these elements are checked in, too, because they are part of the test. Therefore, when you drill down in the asset, you can view or compare the version that existed when the test was checked in, in addition to the currently saved version.
- View or compare versions of **associated external assets**. An associated asset is any external (not integral) resource file used by an asset (such as a function library, a shared object repository, a data table, or a recovery scenario).

Note:

- When you drill down while viewing or comparing an asset, the currently checked in content of the associated assets is displayed, even if you are viewing or comparing an earlier version of the main asset.

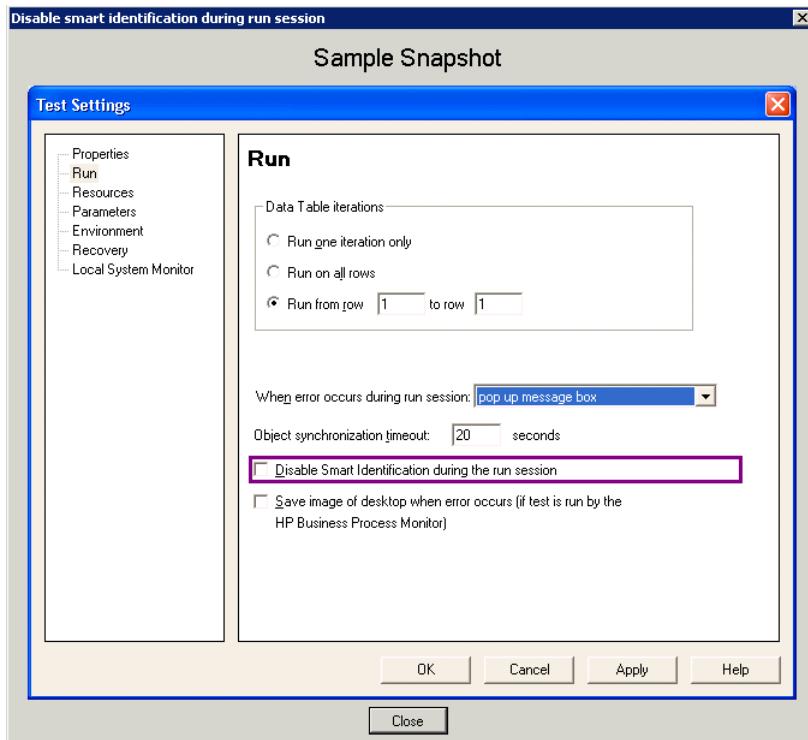
To view or compare an earlier version of the drilled-down associated asset, open the resource file itself and use the Asset Viewer or Asset Comparison Tool.

- To view or compare by drilling down, the resource file must be associated via an absolute or Quality Center path. For details, see "Relative Paths and Quality Center" on page 1656.
-

View a screen capture depicting the QuickTest location of an element

The screen capture displays an example of the relevant dialog box. The option (or area) for the node you right-clicked is highlighted in the screen capture.

For example, suppose you are viewing a comparison of a test, and you notice that the Disable Smart Identification during the run session node is highlighted, indicating that it was modified. If you are not sure where this option is located in QuickTest, you can right-click the node in the comparison tree and select **View Sample Snapshot**. QuickTest then opens a dialog box showing you that this area is located in the Run pane of the Test Settings dialog box. The title bar of the dialog box lists the selected element, and a purple rectangle outlines the option.



For details, see "How to Work with the Asset Comparison Tool and Asset Viewer" on page 1679 and "Shortcut Menu Commands" on page 1685.

Tasks

How to Open the QuickTest Asset Comparison Tool

This task describes how to open the Asset Comparison Tool.

Prerequisites

To open the Asset Comparison Tool from QuickTest or from Quality Center, the asset must be saved in a version control-enabled Quality Center project.

Open the Asset Comparison Tool from any of the following:

The main QuickTest window

- 1 Open the test or function library whose versions you want to compare.
- 2 Select **File > ALM/QC Version Control > Version History or Baseline History**. The Version History or Baseline History dialog box opens.
- 3 Select two versions (using the CTRL key) and click **Compare**. The Asset Comparison Tool opens.

The Object Repository Manager

- 1 Open the Object Repository Manager (**Resources > Object Repository Manager**).
- 2 Browse to and open the shared object repository whose versions you want to compare. For more information, see "Open a shared object repository" on page 259.
- 3 Select **File > ALM/QC Version Control > Version History or Baseline History**. The Version History or Baseline History dialog box opens.
- 4 Select two versions (using the CTRL key) and click **Compare**. The Asset Comparison Tool opens.

The Recovery Scenario Manager

- 1** Open the Recovery Scenario Manager (**Resources > Recovery Scenario Manager**).
- 2** Open the recovery scenario file whose versions you want to compare. For more information, see "Recovery Scenario Manager Dialog Box" on page 1534.
- 3** Click the **Version Control** down arrow and select **Version History** or **Baseline History**.
- 4** Select two versions (using the CTRL key) and click **Compare**. The Asset Comparison Tool opens.

Quality Center

- 1** In Quality Center, connect to the project containing the asset you want to compare.
- 2** Do one of the following:
 - Click the **Test Plan** button in the sidebar to open the Test Plan module.
 - Click the **Test Resources** button in the sidebar to open the **Test Resources** module. This module contains the resource files associated with your test, such as function libraries, shared object repositories, data tables, and recovery scenarios.
- 3** In the tree, select the file whose versions you want to compare.
- 4** Click the **History** tab, and then click the **Versions and Baselines** tab.
- 5** In the **View by** box, select either **Versions** or **Baselines**.
- 6** In the grid, select two versions to compare (using the CTRL key), and then click the **Compare** button.
- 7** In the sidebar of the window that opens, click the **QTP Comparison** button. The Asset Comparison Tool opens.



Tip: You can also compare baselines from the **Management** module. Click the **Management** button in the side bar to open the **Management** module. Select a baseline in the tree and click the **Compare To** button. For more information, see the Quality Center or HP ALM user guide. For more information on baselines, see "Managing Versions of Assets in Quality Center Overview" on page 1696.

The Command Line Interpreter (cmd.exe):

- 1** Open the Command Line Interpreter.
- 2** Enter the command using the following syntax:

"<Asset Comparison Tool executable path>" P1: "<file path 1>" P2: "<file path 2>"
where **P1** = the file system path to the first asset, and **P2** = the file system path to the second asset.

Note: Make sure you insert a blank space after each argument. The options are not case-sensitive and can be entered in any order.

Example:

```
"C:\Program Files\HP\QuickTest Professional\Bin\QTPDiffApplication.exe" P1: "C:\Program Files\HP\QuickTest Professional\Tests\Test1" P2: "C:\Program Files\HP\QuickTest Professional\Tests\Test2"
```

How to Open the QuickTest Asset Viewer

This task describes how to open the Asset Viewer.

Prerequisites

To open the Asset Viewer from QuickTest or from Quality Center, the asset must be saved in a version control-enabled Quality Center project.

Open the Asset Viewer from any of the following:

The main QuickTest window

- 1 Open the test or function library for which you want to view an earlier version.
- 2 Select **File > ALM/QC Version Control > Version History**. The Version History dialog box opens.
- 3 Select a version and click **View**. The Asset Viewer opens.

The Object Repository Manager

- 1 Open the Object Repository Manager (**Resources > Object Repository Manager**).
- 2 Browse to and open the shared object repository for which you want to view an earlier version. For more information, see "Open a shared object repository" on page 259.
- 3 Select **File > ALM/QC Version Control > Version History**. The Version History dialog box opens.
- 4 Select a version and click **View**. The Asset Viewer opens.

The Recovery Scenario Manager

- 1 Open the Recovery Scenario Manager (**Resources > Recovery Scenario Manager**).
- 2 Open the recovery scenario file for which you want to view an earlier version. For more information, see "Recovery Scenario Manager Dialog Box" on page 1534.

- 3** Click the **Version Control** down arrow and select **Version History**.
- 4** Select a version and click **View**. The Asset Viewer opens.

Quality Center

Connect to the project containing the asset you want to view, and do one of the following:

To view the current version of an asset:

In the Test Resources module, select the resource and click the **Resource Viewer** tab.

To view the current or earlier version of an asset:

- 1** Do one of the following:
 - Click the **Test Plan** button in the sidebar to open the Test Plan module.
 - Click the **Test Resources** button to open the **Test Resources** module. This module contains the resource files associated with your test, such as function libraries, shared object repositories, data tables, and recovery scenarios.
- 2** In the tree, select the file for which you want to view an earlier version.
- 3** Click the **History** tab, and then click the **Versions and Baselines** tab.
- 4** In the **View by** box, select **Versions**.
- 5** In the grid, select a version, and then click the **View** button. (You cannot view a version that is currently checked out.) A window opens with buttons in the sidebar enabling you to access version-specific information for the selected asset. (These buttons are identical to the tabs displayed in the right pane of the main window for the latest version of the selected asset.) For more information, see the Quality Center or HP ALM user guide.

The Command Line Interpreter (cmd.exe)

Note: You use the Asset Comparison Tool executable path to open the Asset Viewer.

- 1** Open the Command Line Interpreter.
- 2** Enter the command using the following syntax:

"<Asset Comparison Tool executable path>" P1: "<file path 1>"
where **P1** = the file system path to the first asset.

Note: Make sure you insert a blank space after each argument. The options are not case-sensitive and can be entered in any order.

Example:

```
"C:\Program Files\HP\QuickTest Professional\Bin\QTPDiffApplication.exe" P1: "C:\Program  
Files\HP\QuickTest Professional\Tests\Test1"
```

How to Work with the Asset Comparison Tool and Asset Viewer

The following steps describe the tasks most often performed using the Asset Comparison Tool and the Asset Viewer.

View a comparison of two asset versions (Asset Comparison Tool)

For details, see "Asset Comparison Tool" on page 1681.

Drill down to compare or view a specific element

Note: You can drill down any asset that has a blue drilldown arrow  adjacent to it.

- Click the blue drilldown arrow  adjacent to any asset that can be compared. (The pointer changes into a pointing hand in the proximity of the drilldown arrow.)
- Double-click the asset.
- Right-click the asset and select **View Drilldown**.
For more details, see "Shortcut Menu Commands" on page 1685.
- Select an asset and press ENTER on your keyboard.

View the QuickTest location of an element

Right-click the relevant node and select **View Sample Snapshot**. The screen capture displays an example of the relevant dialog box. The option (or area) for the node you right-clicked is highlighted in the screen capture. For more details, see "View a screen capture depicting the QuickTest location of an element" on page 1671.

Modify text and background colors

Modify the text and background colors for the filter types (modified, added, deleted, and so on) in the Asset Comparison Tool window using the Color Settings dialog box (described on page 1687).

When you modify the background color of a filter type, the color of the filter type in the legend at the top of the window changes accordingly. These changes remain in effect unless you change them again or restore the default settings.

View the number of differences for a specific element

In the Asset Comparison Tool, collapse the node representing an element.

If the sub-elements of that element are different between versions, a legend is displayed adjacent to the node. The legend indicates the number of differences that exist under the collapsed element.

In the following example, three sub-elements were modified, one was deleted, and seven were added:

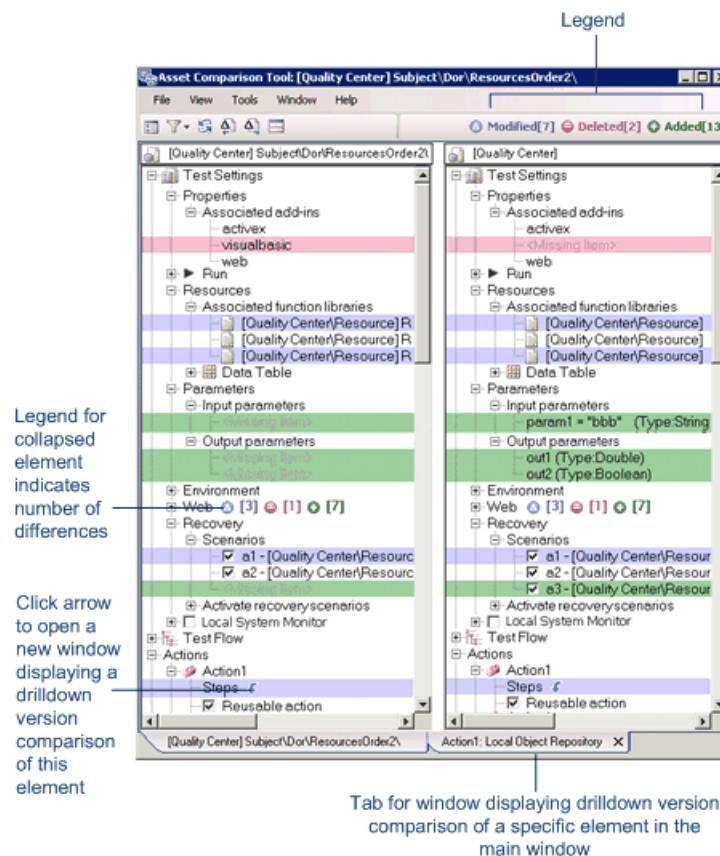


For details, see "Legend" on page 1686.

Reference

Asset Comparison Tool

This window enables you to compare two versions of a particular QuickTest asset, such as a test, a function library, a shared object repository, a data table, or a recovery scenario. It also enables you to drill down in an asset to view a comparison of entities that are associated with the asset, for example, an associated data table or shared object repository.



To access	"How to Open the QuickTest Asset Comparison Tool" on page 1673
Important information	<ul style="list-style-type: none"> ➤ To open the Asset Comparison Tool in QuickTest or Quality Center, the asset must be stored in a Quality Center project that has version control enabled. If the asset is stored in the file system, open the Asset Comparison Tool from the command line interpreter. ➤ The Asset Comparison Tool does not enable you to drill down to view assets that are associated via a relative path. For more information, see "Relative Paths and Quality Center" on page 1656.
Relevant tasks	"How to Work with the Asset Comparison Tool and Asset Viewer" on page 1679
See also	<ul style="list-style-type: none"> ➤ "Asset Viewer" on page 1690 ➤ "Color Settings Dialog Box (Asset Comparison Tool)" on page 1687 ➤ "Filter Dialog Box (Asset Comparison Tool)" on page 1688

User interface elements are described below:

Menu, Toolbar, and Button Options

UI Elements		Description
	File > Exit	Closes the Asset Comparison Tool window. Shortcut key: ALT+F4
	View > Next Difference	Finds the next difference between the elements in the compared versions. Shortcut key: CTRL+DOWN ARROW
	View > Previous Difference	Finds the previous difference between the elements in the compared versions. Shortcut key: CTRL+UP ARROW

UI Elements	Description
	View > Refresh Performs a new comparison of the selected asset versions. Note: This is useful if you are comparing the current version of an asset. If you modify and save the asset, you can use the Refresh command to view an updated comparison. Shortcut key: F5
	Tools > Color Settings Opens the Color Settings dialog box (described on page 1687), enabling you to define the text and background color for each filter type.
	Tools > Filter Opens the Filter dialog box (described on page 1688), enabling you to show or hide the following types of filter elements in the comparison window: <ul style="list-style-type: none"> ➤ Modified  ➤ Deleted  ➤ Added  ➤ Identical Tip: The legend in the top-right corner of the comparison window indicates how many elements match each filter type. The legend adjacent to a collapsed node indicates how many sub-nodes match each filter type. For details, see "Legend" on page 1686.

UI Elements	Description
	<p>Window > Close Window</p> <p>Closes the currently active tabbed comparison window if it was opened from the main comparison window. Enabled only if more than one tabbed comparison window is open.</p> <p>Tips:</p> <ul style="list-style-type: none"> ▶ You can open another tabbed window to view a comparison of an asset that is associated with the currently compared asset, such as a shared object repository or data table. You do this by clicking the blue drilldown arrow  adjacent to any asset that can be compared. ▶ You can also close the comparison tabbed window by clicking the X in the tab at the bottom of the window.
	<p>Window > View Horizontal or View Vertical</p> <p>View Horizontal. Displays the open documents one above the other.</p> <p>View Vertical. Displays the open documents side-by-side.</p>
	<p>Window > <Compared Asset Path></p> <p>Enables you to navigate between the open comparison windows.</p>
	<p>Help > Asset Comparison Tool Help</p> <p>Opens the Asset Comparison Tool Help.</p> <p>Shortcut key: F1</p>
	<p>Previous 2000 Lines</p> <p>If the testing document has more than 2000 lines, this button is displayed at the top of the comparison pane. Click to hide the current 2000 lines and display the previous 2000 lines of the testing document.</p>
	<p>Next 2000 Lines</p> <p>If the testing document has more than 2000 lines, this button is displayed at the bottom of the comparison pane. Click to hide the current 2000 lines and display the next 2000 lines of the testing document.</p>

Shortcut Menu Commands

Command	Description
View Drilldown of Selected Asset	<p>Opens a drilldown version comparison of the selected asset in a new window. (Relevant only for assets that can be compared.)</p> <p>Shortcut key: ENTER</p> <p>Tips:</p> <ul style="list-style-type: none"> ▶ You can also click the blue drilldown arrow  adjacent to the node to open a drilldown version comparison in a new window. ▶ You cannot drill down to view assets that are associated via a relative path. For details, see "Relative Paths and Quality Center" on page 1656.
View Sample Snapshot	<p>Opens a window containing a sample image of the selected element in QuickTest, for example, the Resources pane in the Test Settings dialog box. The element itself is highlighted in the snapshot.</p> <p>Shortcut key: CTRL+Q</p>

Legend

If the sub-elements of an element are different between versions, and you collapse the node representing that element, a legend is displayed adjacent to the node. This legend indicates the number of differences that exist under the collapsed element.

The following is an example of the filter legend displayed in the top-right corner of the Asset Comparison Tool window:

Modified[7] Deleted[2] Added[13]

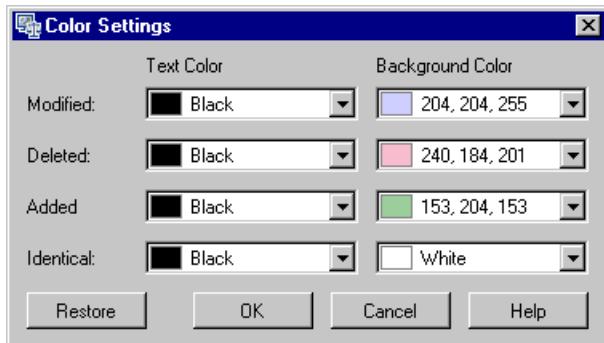
Important information	<ul style="list-style-type: none"> ► If you modify the background color of a filter type (using the Color Settings dialog box), the color of the filter type in the legend changes accordingly. ► If you collapse an asset in the comparison window, the tool displays a legend for that asset, as shown in the following example:
	[3] [1] [7]
Relevant tasks	"How to Work with the Asset Comparison Tool and Asset Viewer" on page 1679
See also	"Color Settings Dialog Box (Asset Comparison Tool)" on page 1687

User interface elements are described below:

Symbol	Description	Number
	Modified	Indicates the number of modified elements in the comparison.
	Deleted	Indicates the total number of elements that were deleted from either of the versions being compared.
	Added	Indicates the total number of elements that were added to either of the versions being compared.

Color Settings Dialog Box (Asset Comparison Tool)

This dialog box enables you to modify the text and background colors for the various filter elements in the Asset Comparison Tool window. The changes remain in effect for all subsequent sessions.



To access	In the Asset Comparison Tool window: ► Select the Tools > Color Settings menu command. ► Click the Color Settings toolbar button  .
Important information	If you change the background color for a filter type, the legend in the top-right corner of the Asset Comparison Tool window changes accordingly. These changes remain in effect unless you change them again or restore the default settings.
See also	"Asset Comparison Tool" on page 1681

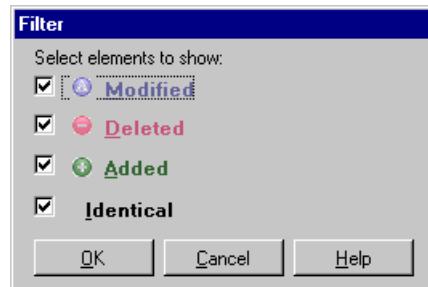
User interface elements are described below:

UI Elements	Description
Modified	Text color and background color for the relevant filter elements.
Deleted	
Added	You can:
Identical	<ul style="list-style-type: none"> ▶ Click a down arrow  to select a color from the list of colors in the Custom, Web, or System tabs. ▶ Enter an RGB value directly in the edit box.
Restore	Restores the default color values for each of the filter elements.

Filter Dialog Box (Asset Comparison Tool)

This dialog box enables you to show or hide elements in the comparison window according to filter criteria.

To access	In the Asset Comparison Tool window: <ul style="list-style-type: none"> ▶ Select the Tools > Filter menu command. ▶ Click the Filter toolbar button .
See also	"Asset Comparison Tool" on page 1681

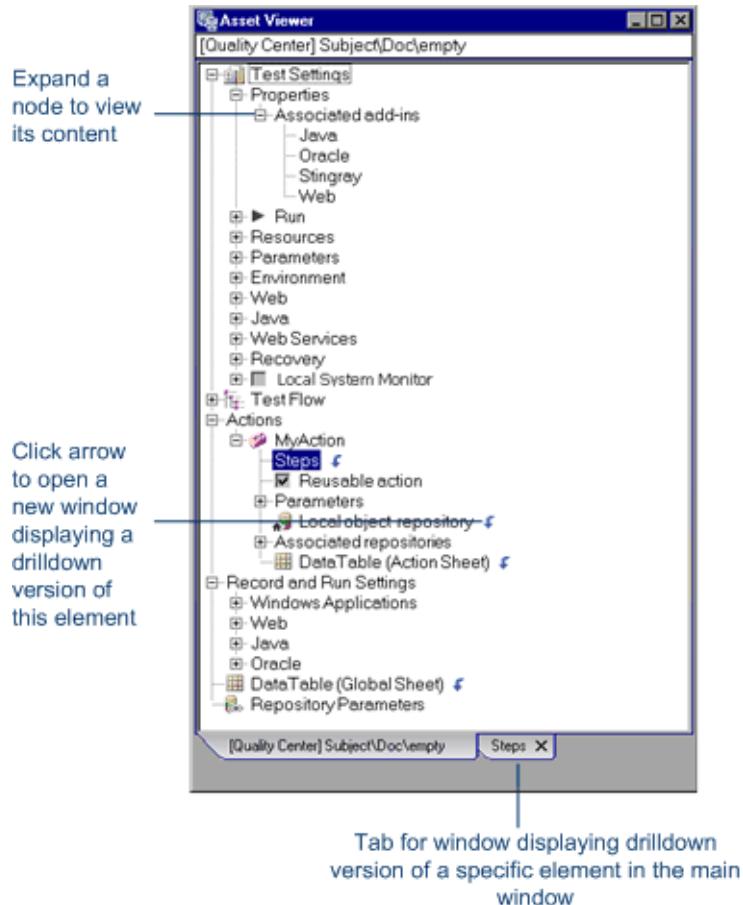


User interface elements are described below:

UI Elements	Description
Select elements to show	<p>Select or clear a check box. The comparison window displays only those elements that match the defined filter. You can show or hide the following types of elements:</p> <ul style="list-style-type: none">►  Modified►  Deleted►  Added►  Identical

Asset Viewer

This window provides a functional overview of an asset, enabling you to view its configurations and settings in a viewer format. The tree view enables you to drill down to view or verify a particular setting without needing to open different dialog boxes or even QuickTest.



Expand a node to view its content

Click arrow to open a new window displaying a drilldown version of this element

Tab for window displaying drilldown version of a specific element in the main window

To access	"How to Open the QuickTest Asset Viewer" on page 1676
Important information	<ul style="list-style-type: none"> ► To open the Asset Viewer in QuickTest or Quality Center, the asset must be stored in a Quality Center project that has version control enabled. If the asset is stored in the file system, open the Asset Viewer from the command line interpreter. ► The Asset Viewer does not enable you to drill down to view assets that are associated via a relative path. For more information, see "Relative Paths and Quality Center" on page 1656.
Relevant tasks	"How to Work with the Asset Comparison Tool and Asset Viewer" on page 1679
See also	"Asset Comparison Tool" on page 1681

User interface elements are described below:

Button Options

UI Elements	Description
Previous 2000 Lines	If the testing document has more than 2000 lines, this button is displayed at the top of the pane. Click to hide the current 2000 lines and display the previous 2000 lines of the testing document.
Next 2000 Lines	If the testing document has more than 2000 lines, this button is displayed at the bottom of the pane. Click to hide the current 2000 lines and display the next 2000 lines of the testing document.

Context Menu Commands

Command	Description
View Drilldown of Selected Asset	<p>Opens a drilldown version comparison of the selected asset in a new window. (Relevant only for assets that can be compared.)</p> <p>Shortcut key: ENTER</p> <p>Tips:</p> <ul style="list-style-type: none">▶ You can also click the blue drilldown arrow  adjacent to the node to open a drilldown version comparison in a new window.▶ You cannot drill down to view assets that are associated via a relative path. For details, see "Relative Paths and Quality Center" on page 1656.
View Sample Snapshot	<p>Opens a window containing a sample image of the selected element in QuickTest, for example, the Resources pane in the Test Settings dialog box. The element itself is highlighted in the snapshot.</p> <p>Shortcut key: CTRL+Q</p>

Troubleshooting and Limitations - Asset Comparison Tool

This section describes troubleshooting and limitations for the Asset Comparison Tool and the Asset Viewer.

Sometimes, checkpoint and output value comparison information disappears from the bottom panes of the Asset Comparison Tool after refreshing it. This can occur if you:

- 1** Open the Asset Comparison Tool to view a comparison.
- 2** Switch to the Object Repository Manager.
- 3** Modify any of the current comparison's checkpoints or output values.
- 4** Save your changes.
- 5** Switch back to the Asset Comparison Tool.
- 6** Refresh the Asset Comparison Tool.
- 7** Select the checkpoint or output value that you modified.

Workaround: If this occurs, close the Asset Comparison Tool, and then open it again after you save your changes.

Version Control in Quality Center 10.00 or HP ALM

Note: The references to Quality Center in this chapter apply to Quality Center 10.00 and HP ALM. Note that some features and options may not be supported in the Quality Center or HP ALM edition you are using. For information on Quality Center or HP ALM editions, see the *HP Quality Center User Guide* or the *HP Application Lifecycle Management User Guide*.

This chapter includes:

Concepts

- ▶ Managing Versions of Assets in Quality Center Overview on page 1696

Tasks

- ▶ How to Check In the Currently Open Asset on page 1703
- ▶ How to Check Out the Latest Version of an Asset on page 1704
- ▶ How to Cancel a Check-Out Operation on page 1705

Reference

- ▶ Version Management Commands on page 1706
- ▶ Check Out Dialog Box on page 1707
- ▶ Check In Dialog Box on page 1708
- ▶ Version History Dialog Box on page 1709
- ▶ Baseline History Dialog Box on page 1712

Troubleshooting and Limitations - Quality Center Version Control
on page 1715

Concepts

Managing Versions of Assets in Quality Center Overview

When QuickTest is connected to a Quality Center project with version control support, you can update and revise your QuickTest assets while maintaining earlier versions of each asset. This helps you keep track of the changes made to each asset and see what was modified from one version to another. Assets can include tests, function libraries, shared object repositories, recovery scenarios, and external data tables.

You can check in the asset at any time. For example, you may want to check the asset in every day or when you complete a task. While the asset is checked out to you, other users can view the last checked in version of that asset in read-only mode, but they cannot modify the asset or view your changes until you check in the asset.

If the asset is...	You can...
checked in	<ul style="list-style-type: none">▶ Open the asset in read-only mode using the Open option. You cannot modify the asset.▶ Open the asset and check it out immediately using the Open and Check out option in the Open <Asset Type> dialog box. You can modify the asset as needed. <p>Tip: If a test or function library is checked into a project with version control, the document tab and the QuickTest title bar both indicate its (Read-Only) status.</p> <p>Note: If you check out a test, and that test is associated with a data table that is currently checked in to the Quality Center project, the data in that data table is read-only, even though the test is editable. To modify the data table, you must first check out the data table in the Quality Center Test Resources module.</p>

If the asset is...	You can...
checked out to your Quality Center user name	<p>Open the asset in read-write mode, using the Open option and modify the asset as needed.</p> <p>Tip: If a test or function library is checked out to your Quality Center user name, a green, unlocked icon  on the document tab indicates this status.</p>
checked out to another Quality Center user	<p>Open the asset in read-only mode using the Open option. QuickTest displays a message indicating that the asset is checked out to another Quality Center user. You can view the last checked in version of the asset now, and you can check out the asset later after the other user checks in the asset.</p> <p>Tip: If a test or function library is checked out to another Quality Center user name, the document tab indicates this status by displaying a red, locked icon  and (Read-Only) adjacent to the document's name.</p>

In QuickTest, you can check out only the latest version of an asset, although you can view and compare earlier versions. This is because assets that are stored in Quality Center are often linked to or **dependent on** one another.

For example, if you try to run an earlier version of a test with a later version of a shared object repository, your test might fail because the objects in the object repository would not necessarily match the objects in the test. For more details, see "Troubleshooting and Limitations - Quality Center Version Control" on page 1715.

Viewing Version Control Information When Opening a Test

When you open a test from a Quality Center project with version control support, you can view version control information for the test by clicking the **Views** down arrow and selecting **Details**.

The **Checked Out To** column specifies the user name of the Quality Center user to whom the test is checked out, if it is checked out. If the test is currently checked in to the version control database, there is no indication in the dialog box.

This section also includes:

- "How Quality Center Manages Assets" on page 1698
- "View and Compare Asset Versions" on page 1699
- "Adding Assets to the Version Control Database in a Quality Center Project" on page 1699
- "Checking Assets Out of the Version Control Database" on page 1700
- "Checking Assets into the Version Control Database" on page 1700
- "Viewing Baseline History" on page 1701
- "Version History Versus Baseline History" on page 1702

How Quality Center Manages Assets

You manage asset versions by checking assets in and out of the version control database.

You add an asset to the version control database by saving it in a Quality Center project with version control support. When you save an asset for the first time, QuickTest automatically checks the asset into the Quality Center version control database, assigns it version number 1, and automatically checks the asset out for you so that you can continue working on it. When you check the asset in, the asset retains version number 1, since this is the first version that can contain content. Then, each time the asset is checked out and in again, the version number increases by 1.

Note: If you create an asset directly in Quality Center, the asset is assigned version number 1 and is immediately checked out to you. In Quality Center, version number 1 represents the created asset without content. When you next check the asset in, Quality Center assigns it version number 2.

 **View and Compare Asset Versions**

You can view and compare the versions of an asset using the Asset Comparison Tool. For more information, see "Viewing and Comparing Versions of QuickTest Assets" on page 1669.

If your project administrator creates project baseline versions when a milestone is reached during product development, you can view and compare the asset versions stored in these baselines. For more information, see "Viewing Baseline History" on page 1701.

Note: With the exception of the **Baseline History** option, the **Quality Center Version Control** options in the **File** menu are available only when you are connected to a Quality Center project with version control support, and an asset stored in Quality Center is open in the QuickTest window.

**Adding Assets to the Version Control Database in a Quality Center Project**

When you use **Save As** to save a new asset in a Quality Center project with version control support, QuickTest automatically saves the asset in the project, checks the asset into the version control database with version number 1, and then checks it out so that you can continue working. This is an administrative version of the asset, similar to a placeholder. The version number indicates that the asset exists in the database. When you later check in the asset, the version number remains version number 1—the first version that you are checking in. Subsequent checkins increase the version number by 1.

Saving your changes to an existing asset does not check them in. Even if you save and close the asset, the asset remains checked out until you choose to check it in. For more information, see "Checking Assets into the Version Control Database" on page 1700.



Checking Assets Out of the Version Control Database

When you open an asset that is currently checked in to the version control database, it is opened in read-only mode. You can review the checked-in asset. You can also run the asset and view the results.

To modify the asset, you must check it out. When you check out an asset, Quality Center copies the asset to your unique check-out directory (automatically created the first time you check out an asset), and locks the asset in the project database. This prevents other users of the Quality Center project from overwriting any changes you make to the asset. However, other users can still run the version that was last checked in to the database.

You can save and close the asset, but it remains locked until you return the asset to the Quality Center database. To release the asset, either check the asset in, or undo the check out operation. For more information on checking assets in, see "Checking Assets into the Version Control Database" on page 1700. For more information on undoing the check-out, see "How to Cancel a Check-Out Operation" on page 1705.

In QuickTest, the check out option accesses the latest version of the asset. In Quality Center, you can also check out earlier versions of the asset. For more information, see "Version History Dialog Box" on page 1709 and Quality Center or HP ALM user guide.



Checking Assets into the Version Control Database

While an asset is checked out, Quality Center users can run the previously checked-in version of your asset. For example, suppose you check out version 3 of an asset and make a number of changes to it and save the asset. Until you check the asset back into the version control database as version 4, Quality Center users can continue to run version 3.

When you have finished making changes to an asset and you are ready for Quality Center users to use your new version, you check it in to the version control database.

Note: If you do not want to check your changes into the Quality Center database, you can undo the check-out operation. For more information, see "How to Cancel a Check-Out Operation" on page 1705.

When you check an asset back into the version control database, Quality Center deletes the asset copy from your checkout directory and unlocks the asset in the database so that the asset version will be available to other users of the Quality Center project.

Viewing Baseline History

In Quality Center, a project administrator can create baselines that provide "snapshots" of an entire project (or part of a project) at different stages of development. A **baseline** represents a version of a project at a specific point in a project's life cycle. For example, baselines are often created for each milestone or when specific phases in a project are completed.

Baselines can be created for Quality Center projects that are enabled for version control, and for projects for which version control is not enabled.

The project administrator creates the baseline in the Libraries tab of the Management module in Quality Center. Creating a baseline is a two-fold process. The administrator first creates a library, which specifies the root folders from which to import the data. The administrator makes sure to include all of the associated resource files, such as shared object repositories and function libraries. The administrator then creates the actual baseline, which comprises the latest versions of every asset included in the library. If the project is version control-enabled, then these are the latest checked in versions of every asset.

During the creation process, Quality Center verifies that all of these assets (such as associated resource files) are included in the baseline. If any assets are not included, Quality Center informs the administrator so that the library and baseline can be modified accordingly. For more information, see the Quality Center or HP ALM user guide.

In Quality Center, these baselines can be viewed and compared in their entirety.

In QuickTest, you can view and compare the assets saved in these baselines. This enables you to review the content of an asset at a specific phase in the project time line.

You can also run a test from a baseline.

Version History Versus Baseline History

This section focuses on the differences between version history and baseline history and describes when to use each.

- You use version control to check in and check out assets as needed. For example, you may want to check in an asset on a daily basis or only when significant results are achieved. This enables you to monitor the asset's development.

If you want to view the content of an asset on a particular date or after a particular user checked in the asset, use the **Version History** option to view or compare the asset.

- The Quality Center project administrator creates baselines that represent "snapshots" of a project's assets at various milestones in a project's life cycle. Each baseline links to the assets specified by the administrator when the baseline was created. The asset version represented in the baseline is always the version that was checked in when the baseline was created.

If you want to view an asset as it was saved for a particular milestone, use the **Baseline History** option.

Tasks

How to Check In the Currently Open Asset

This task includes the following steps:

- "Prerequisites" on page 1703
- "Check in the asset using the Check In dialog box" on page 1703

1 Prerequisites

Confirm that the currently open asset is checked out to you. For more information, see "Version History Dialog Box" on page 1709.

Note: If the open asset is currently checked in, the **Check In** option is disabled. If you open an asset that is checked out to another user, all **Quality Center Version Control** options, except the **Version History** option, are disabled.

2 Check in the asset using the Check In dialog box

Select **File > ALM/QC Version Control > Check In**. For details, see "Check In Dialog Box" on page 1708.

How to Check Out the Latest Version of an Asset

This task includes the following steps:

- "Prerequisites" on page 1704
- "Check out the asset using the Open dialog box" on page 1704
- "Check out the asset using the File menu" on page 1705

Prerequisites

Before you check out an asset, make sure the asset you want to check out is currently checked in. If you open an asset that is checked out to you, the **Check Out** option is disabled. If you open an asset that is checked out to another user, all Quality Center version control options, except the **Version History** option, are disabled.

Note about version numbers: Prior to Quality Center 10.00, version numbers consisted of three segments separated by periods, for example 1.7.4. In Quality Center 10.00 and HP ALM, version numbers consist of a single segment, for example 12.

Check out the asset using the Open dialog box

- 1 Open the Open <Asset type> dialog box as follows:

If the asset is a:	Do this:
Test or Function Library	In the main QuickTest window, select File > Open > Test or Function Library , or click the Open down arrow and select the asset type from the list.
Shared Object Repository	In the Object Repository Manager, select File > Open or click the Open button.
Recovery Scenario	In the Recovery Scenario Manager, click the Open button.

- 2 Browse to and select the asset.

- 3 Click the **Open** down arrow and select **Open and Check out**. The asset opens, checked out to you. For testing documents, the title bar indicates the checkout status. For details, see "Managing Versions of Assets in Quality Center Overview" on page 1696.

Check out the asset using the File menu

- 1 Open the asset you want to check out.
- 2 Open the Check Out Dialog Box (**File > ALM/QC Version Control > Check Out**). For details, see "Check Out Dialog Box" on page 1707.

Note: You can save changes and close the asset without checking the asset in, but your changes will not be available to other Quality Center users until you check it in. If you do not want to check your changes in, you can undo the check-out. For more information on checking assets in, see "Checking Assets into the Version Control Database" on page 1700. For more information on undoing the check-out, see "How to Cancel a Check-Out Operation" on page 1705.



How to Cancel a Check-Out Operation

This task describes how to cancel a check out operation so that the asset will be available for check out by other Quality Center users. For example, you might cancel a check out operation when you check out an asset and then decide that you do not want to upload the modified asset to Quality Center.

- 1 Open the asset if it is not already open.
- 2 Select **File > ALM/QC Version Control > Undo Check out**.
- 3 Click Yes to confirm the cancellation of your check out operation.

The check out operation is cancelled. The checked out asset closes, and the previously checked in version reopens in read-only mode.

Reference

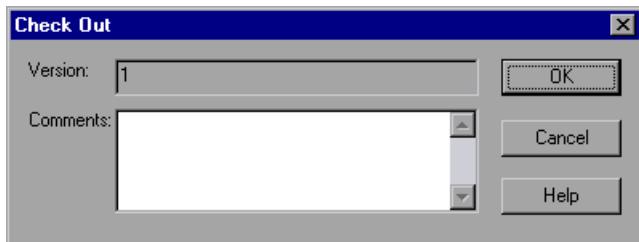
Version Management Commands

The following version control commands are available in QuickTest and can be used when connected to a Quality Center project that has version control enabled:

- **Check Out.** Enables you to check a version-controlled asset out of the version control database. For more information, see "Checking Assets Out of the Version Control Database" on page 1700.
- **Undo Check Out.** Enables you to cancel the check out of a version-controlled asset from the version control database. For more information, see "How to Cancel a Check-Out Operation" on page 1705.
- **Check In.** Enables you to check an asset in to the version control database. For more information, see "Checking Assets Out of the Version Control Database" on page 1700.
- **Version History.** Enables you to view or compare the versions of a particular asset. For more information, see "Managing Versions of Assets in Quality Center Overview" on page 1696.
- **Baseline History.** Enables you to view or compare the versions of a particular asset as it was saved in a project's baselines. For more information, see "Viewing Baseline History" on page 1701.

Check Out Dialog Box

This dialog box enables you to check a Quality Center asset out of the Quality Center version control database so that you can edit it.



Note: This dialog box may look different if you are connected to a Quality Center 9.2 project. For more information, see "Quality Center 9.2 Check Out Dialog Box" on page 1725.

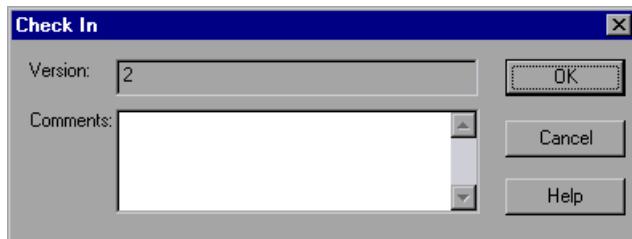
To access	File > ALM/QC Version Control > Check Out
Relevant tasks	"How to Check Out the Latest Version of an Asset" on page 1704

User interface elements are described below:

UI Elements	Description
Version	The version number of the asset.
Comments	Enter any comments for the version in the Comments area.

Check In Dialog Box

This dialog box enables you to check a Quality Center asset in to the Quality Center version control database.



Note: This dialog box may look different if you are connected to a Quality Center 9.2 project. For more information, see "Quality Center 9.2 Check Out Dialog Box" on page 1725.

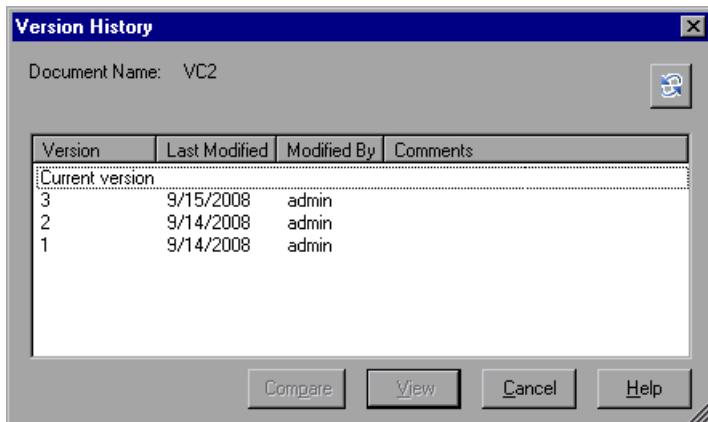
To access	File > ALM/QC Version Control > Check In
Relevant tasks	"How to Check In the Currently Open Asset" on page 1703

User interface elements are described below:

UI Elements	Description
Version	The new version number. By default, the new version number is one number higher than the previously checked in version.
Comments	If you entered a description of your change when you checked out the asset, the description is displayed in the Comments box. You can enter or modify the comments in the box.

Version History Dialog Box

This dialog box enables you to view the version history for an asset, view the content of a previous asset version, and compare two asset versions.



Note: This dialog box may look different if you are connected to a Quality Center 9.2 project. For more information, see "Quality Center 9.2 Version History Dialog Box" on page 1726.

To access	<ul style="list-style-type: none">▶ From most assets: Open the asset and select the File > ALM/QC Version Control > Version History menu command.▶ From a recovery scenario: In the Recovery Scenario Manager, open the recovery scenario, click the Version Control down arrow, and select Version History.
------------------	--

Important information	<p>To view a version for an asset: Select a version and click View.</p> <p>To compare two versions of an asset: Select two versions and click Compare.</p> <ul style="list-style-type: none"> ▶ This dialog box may look different if you are connected to a Quality Center 9.2 project. For more information, see "Quality Center 9.2 Version History Dialog Box" on page 1726. ▶ You cannot check out an earlier version of an asset from this dialog box. (You can check out earlier version of most assets directly from the Quality Center project. For more information on checking out assets from Quality Center, see the Quality Center or HP ALM user guide.)
See also	"Managing Versions of Assets in Quality Center Overview" on page 1696

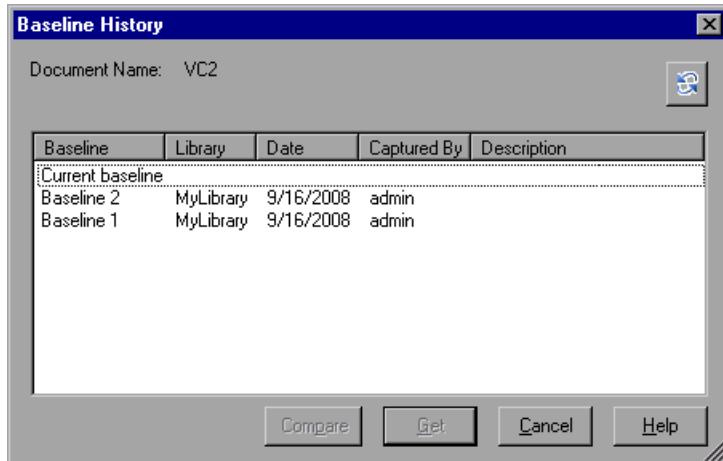
User interface elements are described below:

UI Elements		Description
	Document Name	The name of the currently open asset.
	Refresh button	Reloads the versions in the Version History dialog box with the latest changes.
	Version column	A list of all versions of the asset.
	Last Modified column	The date that each version was checked in.
	Modified By column	The user who checked in each listed version.
	Comments column	The comments that were entered when the selected asset version was checked in.

UI Elements	Description
	<p>Compare button</p> <p>Enables you to compare two versions of the currently open asset.</p> <p>To compare two versions: Select the versions you want to compare and click Compare. QuickTest opens the two asset versions in the Asset Comparison Tool. For more information, see "Asset Comparison Tool" on page 1681.</p>
	<p>View button</p> <p>Enables you to view the selected version of the current asset.</p> <p>To view a version of an asset: Select an asset version and click View. QuickTest opens the checked in version of the asset in the Asset Viewer. For more information on the Asset Viewer, see "Asset Viewer" on page 1690.</p>

Baseline History Dialog Box

This dialog box enables you to view and compare read-only baseline "snapshots" of an asset.



To access	<ul style="list-style-type: none"> ▶ Most assets: Open the asset and select the File > ALM/QC Version Control > Baseline History menu command. ▶ Recovery scenario: In the Recovery Scenario Manager, open the recovery scenario, click the Version Control down arrow, and select Baseline History.
Important information	In the Quality Center Test Lab module, you can use the Pin to Baseline option to run a baseline version of an asset. For more information, see the Quality Center or HP ALM user guide.
See also	"Viewing Baseline History" on page 1701

User interface elements are described below:

UI Elements		Description
	Document Name	Specifies the name of the currently open asset.
	Refresh button	Reloads the baselines in the Baseline History dialog box with the latest changes. For example, if a baseline is added while this dialog box is open, clicking Refresh updates the list of baselines.
	Baseline column	Lists all of the baselines that include this asset. Baselines are defined in the Quality Center project (Management module > Libraries tab).
	Library column	Lists the libraries from which each baseline was created.
	Date column	Lists the date that each baseline was created.
	Captured By column	Lists the Quality Center user who created each listed baseline.
	Description column	Displays any comments that were added when the baseline was created.

UI Elements	Description
	<p>Compare button</p> <p>Enables you to view a comparison of the currently open asset in two baselines.</p> <p>To compare two baselines: Select the baselines you want to compare and click Compare. QuickTest opens the two baseline versions of the asset in the Asset Comparison Tool. For more information, see "Asset Comparison Tool" on page 1681.</p>
	<p>Get button</p> <p>Enables you to open the current asset from the selected baseline.</p> <p>To view the asset as it was stored in a baseline: Select a baseline from the list and click View.</p> <p>When you click Get, QuickTest:</p> <ul style="list-style-type: none"> ➤ Closes the currently open asset. ➤ Opens the same asset from the baseline you selected. ➤ Loads the baseline version of the external actions and resource files that are associated with the asset, if any, when they are called. <p>Note: If an external action or resource file is associated via a relative path, loads the latest version of the action or resource file instead of the version from the baseline.</p>

Troubleshooting and Limitations - Quality Center Version Control

This section describes troubleshooting and limitations for Quality Center version control.

- If you need to check out an earlier version of an asset, for example, to roll back to an earlier version, contact your Quality Center project administrator. Your administrator needs to ensure that the correct versions of all relevant assets become the latest versions.
- When working with a version-control-enabled Quality Center project, it takes a long time to save a test for the first time (up to twice as long as saving the same test in a project without version control support enabled). This delay does not occur on subsequent saves of the test.

Version Control in Quality Center 9.2

Note: References to Quality Center features and options in this chapter apply to Quality Center 9.2. However, they may not be supported in the Quality Center 9.2 version you are using. For more information on Quality Center editions, see the *HP Quality Center User Guide*.

This chapter includes:

Concepts

- Version Control for QuickTest Tests in Quality Center 9.2 Overview on page 1718

Tasks

- How to Add Tests to the Quality Center 9.2 Version Control Database on page 1721
- How to Perform a Version Control Operation on Tests Stored in Quality Center 9.2 on page 1722

Reference

- Quality Center 9.2 Check In Dialog Box on page 1724
- Quality Center 9.2 Check Out Dialog Box on page 1725
- Quality Center 9.2 Version History Dialog Box on page 1726

Concepts

Version Control for QuickTest Tests in Quality Center 9.2 Overview

When QuickTest is connected to a Quality Center 9.2 project with version control support, you can update and revise your automated test scripts while maintaining earlier versions of each test. This helps you keep track of the changes made to each test, see what was modified from one version of a test to another, or return to a previous version of the test.

The Quality Center test repository always contains the latest version of a test, and Quality Center always uses the latest version for all test runs.

Note:

- A Quality Center server with version control support requires the installation of version control software as well as the Quality Center Version Control Add-in. For more information, see your Quality Center documentation.
 - The **ALM/QC Version Control** options in the **File** menu are available only when you are connected to a Quality Center project database with version control support and you have a Quality Center test open.
-

When a test is checked in, you can open it in read-only mode to review it. You can also run the test and view the results. To modify a test, you must check it out.

This section also includes:

- "Checking In a Test" on page 1719
- "Checking Out a Test" on page 1719
- "Undoing a Check Out" on page 1720
- "Accessing Earlier Test Versions" on page 1720
- "Opening Tests from a Quality Center 9.2 Project with Version Control Support" on page 1720



Checking In a Test

After you finish making changes to a test and are ready for Quality Center users to use your new version, you check it in to the version control database.

When you check a test back into the version control database, Quality Center deletes the test copy from your checkout directory and unlocks the test in the database so that the test version will be available to other users of the Quality Center project.



Checking Out a Test

When you check out a test, Quality Center copies the test to your unique check-out directory (automatically created the first time you check out a test), and locks the test in the project database. This prevents other users of the Quality Center project from overwriting any changes you make to the test. However, other users can still run the version that was last checked in to the database.

You can save and close the test, but it remains locked until you return the test to the Quality Center database. To release the test, you either check the test in, or undo the check out operation. For details, see "Quality Center 9.2 Check In Dialog Box" on page 1724 and "Undoing a Check Out" on page 1720.

By default, when you check out a test, the latest version of the test is checked out. You can also check out earlier versions of the test. For details, see "Accessing Earlier Test Versions" on page 1720 and "Quality Center 9.2 Version History Dialog Box" on page 1726.

While a test is checked out, Quality Center users can run the previously checked-in version of your test. For example, suppose you check out version 1.2.3 of a test and make changes to it and save the test. Until you check the test back in to the version control database as version 1.2.4 (or another number that you assign), Quality Center users can continue to run version 1.2.3.



Undoing a Check Out

If you do not want to check your changes into the Quality Center database, you can undo the check-out operation. For details, see "Cancel a check out" on page 1722.

If you check out a test and then decide that you do not want to upload the modified test to Quality Center, you should cancel the check out operation so that the test will be available for check out by other Quality Center users.



Accessing Earlier Test Versions

You can use the Version History dialog box to view version information for any open test that is stored in the Quality Center version control database, regardless of its current status. You can view an earlier version of a test in read-only mode, or you can check out an earlier version and then check it in as the latest version of the test.

For more details, see "Quality Center 9.2 Version History Dialog Box" on page 1726.



Opening Tests from a Quality Center 9.2 Project with Version Control Support

When you open a test from a Quality Center project with version control support, the Open Test dialog box displays icons that indicate the version control status of each test in the selected subject. The test opens in read-write or read-only mode depending on the current version control status of the test:

- If the test is currently checked into the version control database or is checked out to another user, the test opens in read-only mode.
- If the test is checked out to you, the test opens in read-write mode.

Tasks

How to Add Tests to the Quality Center 9.2 Version Control Database

The following steps describe how to add a test to the Quality Center 9.2 version control database.

Save a test in a Quality Center project with version control

Use **Save As** to save a new test in a Quality Center project with version control support. For details, see "Save Test Dialog Box" on page 412.

Results

QuickTest automatically saves the test in the project, checks the test into the version control database with version number 1.1.1 and then checks it out so that you can continue working.

The QuickTest status bar indicates each of these operations as they occur. Note, however, that saving your changes to an existing test does not check them in. Even if you save and close the test, the test remains checked out until you choose to check it in. For more information, see "Checking In a Test" on page 1719.

How to Perform a Version Control Operation on Tests Stored in Quality Center 9.2

The following steps describe how to perform various version control operations on tests that are stored in a Quality Center 9.2 project with version control enabled.

Prerequisite

Open the test on which you want to perform a version control operation. For details, see "Opening Tests from a Quality Center 9.2 Project with Version Control Support" on page 1720.

Check in a test

Select **File > ALM/QC Version Control > Check In**.

For details, see "Quality Center 9.2 Check In Dialog Box" on page 1724.

Check out a test

Select **File > ALM/QC Version Control > Check Out**.

For details, see "Quality Center 9.2 Check Out Dialog Box" on page 1725.

Cancel a check out

1 Select **File > ALM/QC Version Control > Undo Check out**.

2 Click **Yes** to confirm the cancellation of your check out operation.

The check out operation is cancelled. The checked out test closes, and the previously checked in version reopens in read-only mode.

View an earlier version of a test

1 Choose **File > ALM/QC Version Control > Version History**. The Version History dialog box opens.

2 Select the test version you want to view in the **Version details** list.

3 Do one of the following:

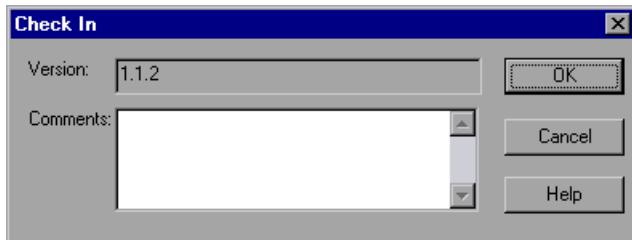
- Click **Get Version** to open a read-only version of the test.
- Click **Check Out** to check out the selected version.

For more details, see "Quality Center 9.2 Version History Dialog Box" on page 1726.

Reference

Quality Center 9.2 Check In Dialog Box

This dialog box enables you to check a Quality Center test into the Quality Center version control database.



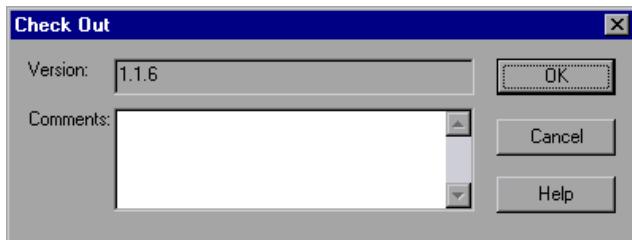
To access	Open the test you want to check out and select File > ALM/QC Version Control > Check In . For details, see "Opening Tests from a Quality Center 9.2 Project with Version Control Support" on page 1720.
Important information	Confirm that the currently open test is checked out to you. For details, see "Quality Center 9.2 Version History Dialog Box" on page 1726.
See also	"Quality Center 9.2 Check Out Dialog Box" on page 1725

User interface elements are described below:

UI Elements	Description
Version	The new version number. By default, the new version number is one number higher than the previously checked in version.
Comments	A description of your change if you entered one when you checked out the test. You can enter or modify the comments in the box.

Quality Center 9.2 Check Out Dialog Box

This dialog box enables you to check a Quality Center test out of the Quality Center 9.2 version control database so that you can edit it.



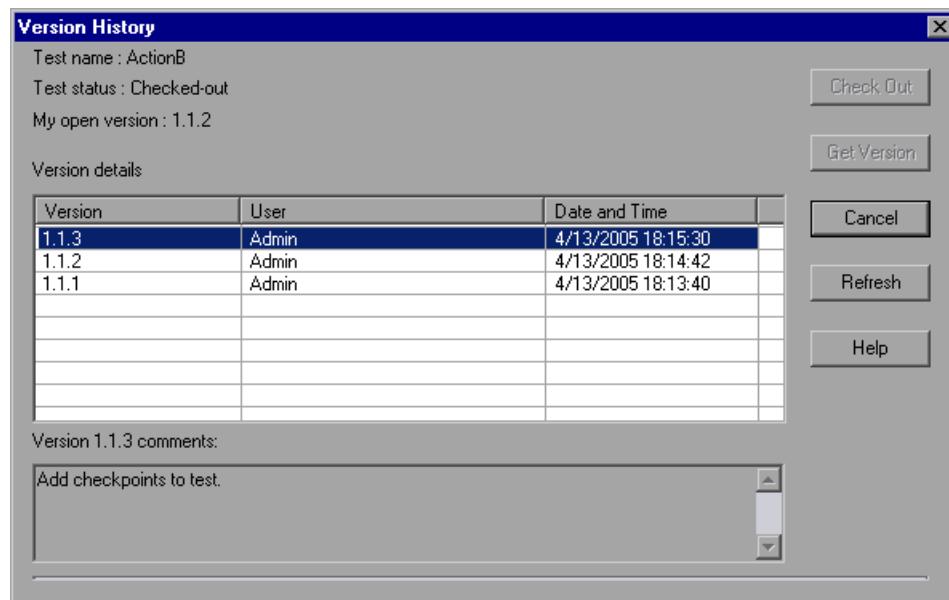
To access	Open the test you want to check out and select File > ALM/QC Version Control > Check Out . For details, see "Opening Tests from a Quality Center 9.2 Project with Version Control Support" on page 1720.
Important information	<ul style="list-style-type: none"> ➤ Before you open a test to check it out, make sure that it is currently checked in. ➤ When you open a test that is checked in to the version control database, it opens in read-only mode. ➤ When you check out a test, the open read-only test closes and automatically reopens as a writable test.
See also	"Quality Center 9.2 Check In Dialog Box" on page 1724

User interface elements are described below:

UI Elements	Description
Version	The test version to check out. By default, the check out option checks out the latest version of the test if the test is checked in. If you selected an earlier version of the test in the Version History dialog box, the Version box displays the version you selected. For details on working with previous versions, see "Quality Center 9.2 Version History Dialog Box" on page 1726.
Comments	Information about the changes you plan to make.

Quality Center 9.2 Version History Dialog Box

This dialog box enables you to view version information about the currently open test and to view or retrieve an earlier version of the test.



To access	Open a test that has been previously checked in and select File > ALM/QC Version Control > Version History . For details, see "Opening Tests from a Quality Center 9.2 Project with Version Control Support" on page 1720.
See also	"Accessing Earlier Test Versions" on page 1720

User interface elements are described below:

UI Elements	Description
Test name	The name of the currently open test.
Test status	<p>The status of the test. The test can be:</p> <ul style="list-style-type: none"> ➤ Checked-in. The test is currently checked in to the version control database. It is currently open in read-only format. You can check out the test to edit it. ➤ Checked-out. The test is checked out by you. It is currently open in read-write format. ➤ Checked-out by <another user>. The test is currently checked out by another user. It is currently open in read-only format. You cannot check out or edit the test until the specified user checks in the test.
My open version	The test version that is currently open on your QuickTest computer.
Check Out	<p>Checks out the test that is currently selected in the Version Details area.</p> <p>When you click this button, QuickTest opens a confirmation message.</p> <p>After you check out the version and finish working on it, you can check in the test as the latest version, or you can undo the checkout. For details, see "Quality Center 9.2 Check In Dialog Box" on page 1724 and "Cancel a check out" on page 1722.</p>

UI Elements	Description
Get Version	<p>Opens the test that is currently selected in the Version Details area in read-only mode.</p> <p>When you click this button, QuickTest reminds you that the test will open in read-only mode because it is not checked out.</p> <p>Tips:</p> <ul style="list-style-type: none"> ➤ To confirm the version number that you now have open in QuickTest, look at the My open version value. ➤ After using the Get Version option to open an earlier version in read-only mode, you can check out the open test by choosing File > ALM/QC Version Control > Check Out. This is the same as using the Check Out button in the Version History dialog box.
Version details	<p>The version details for the test.</p> <ul style="list-style-type: none"> ➤ Version. A list of all versions of the test. ➤ User. The user who checked in each listed version. ➤ Date and Time. The date and time that each version was checked in. <p>You can select a test version to view its comments, and to click Check Out or Get Version.</p>
Version comments	<p>The comments that were entered when the selected test version was checked in.</p>

57

HP ALM Sprinter

This chapter includes:

Concepts

- HP ALM Sprinter Overview on page 1730

Concepts

HP ALM Sprinter Overview

Although QuickTest automated tests can answer many of your testing needs, you may also need to perform some of your tests manually. You can run your manual tests using HP ALM Sprinter, HP's solution for manual testing. Sprinter provides advanced functionality and tools to make manual testing more efficient and effective.

Manual testing often requires that you leave your testing application to accomplish tasks related to your test. For example, you may need to use graphic software to take a screen capture of your application, you may want to record a movie of the application during the test, and you need to switch to your defect tracking software to report defects.

Sprinter addresses these needs in the manual testing process, and enables you to accomplish these tasks without disrupting your test flow. Sprinter includes many tools to help you detect and submit defects. Sprinter can also perform many of the repetitive and tedious tasks of manual testing for you. These features ensure that you can perform all the tasks necessary for your manual test with minimum interruptions to your testing work.

Sprinter is fully integrated with HP ALM, enabling you to get the maximum benefit from both solutions.

Note: QuickTest Professional and Sprinter share a variety of system resources. Consider the following when deciding whether to install Sprinter on your QuickTest computer:

- Sprinter and QuickTest Professional can be installed on the same computer.
 - Sprinter and QuickTest Professional cannot be run simultaneously on the same computer.
 - Any changes to the installation of one of these products will affect the other. If you uninstall, modify, or upgrade one product, the other may fail. You will need to repair the installation of the affected product. For more details, see the *HP QuickTest Professional Readme* and the *HP ALM Sprinter Readme*.
-

With Sprinter you can:

- **Run HP ALM manual tests and Business Process tests with new step display:**
 - **Clear steps display.** Steps are presented in a clear, organized, and user-friendly design, making it easier to view step information and update step status.
 - **Move easily between tests in your run.** You can easily move between the tests in your run without interrupting your test flow. Sprinter updates all your displayed step and run information to match your current test.
 - **Edit actual parameter values during your test run.** You can easily edit the actual values of parameters in your test, during your test run.
 - **Multiple views.** Change the way you view your steps depending on your testing needs. View in normal mode when more details are needed, or view in Subtitles mode if you need to see more of your application.
 - **Actual value including screen captures.** Attach a plain or annotated screen capture of your application to the step's actual value.

- **Create formal tests from exploratory tests with no pre-defined steps.** If you run a test without pre-defined steps, Sprinter can keep a record of all the user actions you took during your test. You can then export this list to an Excel spreadsheet, modify the text as needed and import the spreadsheet to a test in HP ALM. This converts an exploratory test to a formal test, with pre-defined steps.
- **Submit defects to HP ALM.** Submit an HP ALM defect directly from within Sprinter. You can optionally let Sprinter create a defect scenario by automatically generating a text description of all the user actions or steps in your test. You can also attach a screen capture or a movie of your application to the defect.
- **Create and annotate screen captures of your application.** Sprinter provides tools that enable you to take and annotate a screen capture of your application at any point in the testing process. Tools are included that make measuring and comparing user interface elements easier. Report defects in the display by attaching the annotated screen capture to a Quality Center defect, saving it as a file, or attaching it to an email. You can also include annotated screen captures in the Actual Result of a step.
- **Let Sprinter perform some manual testing tasks for you.**
You can create and run **macros** to automate a set of actions in your application. Sprinter can also **inject data** automatically into fields in your application.
- **Replicate your actions on another computer.** Replicate your user actions on multiple computers with different configurations (operating system, browser). Sprinter detects differences in the displays of these computers and enables you to report defects on these differences.
- **View test results.** Sprinter includes a powerful Storyboard that displays each action you performed in your test. For each action you can see a screen capture of the action, any defects that you reported, defect reminders, and comments. If you ran the test with multiple configurations you can view the differences between the displays of different computers.

For more information, contact your HP representative.

Part XII

Working with Other HP Products

58

Service Test Integration

This chapter includes:

Concepts

- Service Test Integration Overview on page 1736

Tasks

- How to Integrate with Service Test on page 1739

Reference

- Call to Service Test Test Dialog Box on page 1741

Concepts

Service Test Integration Overview

Service Test enables you to test your GUI-less applications (also known as headless applications). For example, you can use Service Test to test standard Web Services, non-SOAP Web Services, such as REST, and so on.

You can include calls from your QuickTest test to Service Test tests if:

- Service Test 11.00 is installed on the QuickTest computer, and
- QuickTest is using an HP Unified Functional Testing (also known as **UnifiedFunctionalTesting**) license.

When you insert a call to a Service Test test, the call is displayed under the relevant QuickTest action in the Test Flow pane.

If QuickTest is connected to a Quality Center project that contains Service Test tests, you can call a Service Test test that is stored in that Quality Center project. When you run the test, make sure that the QuickTest client on which you run the test has access to an HP Unified Functional Testing license.

You insert and modify calls to Service Test tests using the Call to Service Test Test dialog box described on page 1741. When you insert a call, you specify the parameter values that QuickTest will use when running the test.

QuickTest creates an XML file containing these parameter values and stores the file in the relevant action's folder. You can modify the parameter values, if needed, using the Call to Service Test Test dialog box.

Example:

Suppose you want to insert a step that calls the MyServiceTest test (MyServiceTest.st) in the first action in MyQTTest.

After you insert a call, QuickTest creates C:\Program Files\HP\QuickTest Professional\Tests\MyQTTest\Action1\MyServiceTestParams-1.xml. This XML file contains all of the parameter values that are used when QuickTest runs the MyServiceTest test when the step calling that test is reached during the run session. (If you need to rename the XML file for any reason, make sure that you also modify the name of the XML file in the step that calls it.)

How QuickTest Runs the Called Service Test Test

When a step containing a call to a Service Test test is reached, QuickTest opens the called Service Test test and runs it. During the run session, the Service Test Monitor window displays a log of the steps that are being performed in the Service Test test, as shown in the following example. For details on what is contained in these steps, see your HP Service Test documentation.

Service Test Monitor - STTest1.st is running...		
Time	Type	Message
2:05 PM	Info	Executing Service Test
2:05 PM	Info	Environment Profiles: HP.SOAQ.VTD.SOAReplayAPI.EnvironmentProfiles
2:05 PM	Info	Test Input Parameters: <Arguments><param1>dog</param1></Arguments>
2:05 PM	Info	Executing Service Test
2:05 PM	Info	Environment Profiles: HP.SOAQ.VTD.SOAReplayAPI.EnvironmentProfiles
2:05 PM	Info	Test Input Parameters: <Arguments><param1>dog</param1></Arguments>
2:05 PM	Info	Executing Service Test
2:05 PM	Info	Environment Profiles: HP.SOAQ.VTD.SOAReplayAPI.EnvironmentProfiles
2:05 PM	Info	Test Input Parameters: <Arguments><param1>dog</param1></Arguments>

The run session behavior is different depending on whether the QuickTest test is run from QuickTest or from Quality Center.

- **QuickTest.** The test runs until the step calling the Service Test test is reached.

If an HP Unified Functional Testing license is currently in use, Service Test opens and runs the Service Test test. When the Service Test test is finished, QuickTest continues running your QuickTest test from the next step.

If QuickTest is not running with this license type, QuickTest fails the test.

- **Quality Center.** If an HP Unified Functional Testing license is available, QuickTest opens with that license and runs the test. If this license type is not available, QuickTest does not open and the test does not run.

After the run session, the results display information about the QuickTest test and the called Service Test test. You can view these results in the Run Results Viewer. For details, see "QuickTest Tests Containing Calls to Service Test Tests" on page 1190.

Tasks

How to Integrate with Service Test

This task describes how to insert a call to a Service Test test.

This task includes the following steps:

- ▶ "Prerequisites" on page 1739
- ▶ "Insert or modify a call to a Service Test test" on page 1739
- ▶ "Results" on page 1740

1 Prerequisites

- ▶ Service Test must be installed on the QuickTest computer. For details, see the Service Test documentation.
- ▶ A **UnifiedFunctionalTesting** license must be loaded. For details, see "Add-in Manager Dialog Box" on page 113.

2 Insert or modify a call to a Service Test test

Use the Call to Service Test Test dialog box (described in "Call to Service Test Test Dialog Box" on page 1741), which you open as follows:

- ▶ To insert a new call, select **Insert > Call to Service Test Test**.
- ▶ To modify an existing call, right-click the step and select **Edit Call to Service Test Test**.

Note: Do not insert a call to a Service Test test that contains a call to a QuickTest test, as this can cause unexpected behavior.

3 Results

QuickTest inserts a step that calls the Service Test test, for example:

```
var_CallServiceTest = CallServiceTest ("MyST.st","MySTParams-1.xml")
```

QuickTest also creates an XML file containing the parameter values used in the Service Test test and stores this file in the action folder.

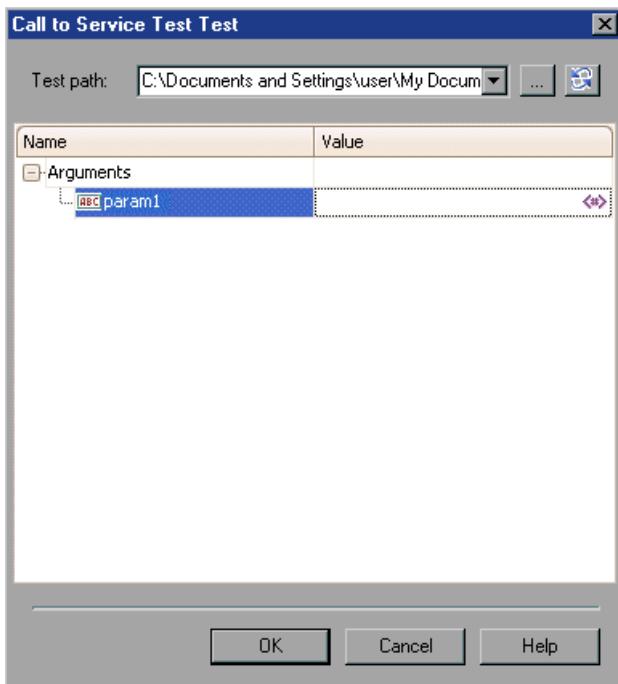
In the above example, the file is named MySTParams-1.xml, and it is stored in C:\Program Files\HP\QuickTest Professional\Tests\MyQTTest\Action1.

Reference

Call to Service Test Test Dialog Box

This dialog box enables you to specify the test path and parameter values for a call to a Service Test test.

The following image shows an example of a call to a Service Test test with one argument for which a value must be defined.



To access	New call: Select Insert > Call to Service Test Test . Existing call: Right-click the step and select Edit Call to Service Test Test .
-----------	---

Important information	Do not insert a call to a Service Test test that contains a call to a QuickTest test, as this can cause unexpected behavior.
Relevant tasks	"How to Integrate with Service Test" on page 1739
See also	"Service Test Integration Overview" on page 1736

User interface elements are described below:

UI Elements	Description
Test path	<p>The path to the Service Test test to which you are inserting a call. You can insert a test path in any of the following ways:</p> <ul style="list-style-type: none"> ➤ browse to a test path ➤ select a test path from the drop down list of recently called Service Test tests ➤ enter a test path manually <p>You can insert a relative or absolute test path.</p>
	Refresh. Updates the arguments in the arguments grid according to the specified test path.
<Argument grid>	<p>The list of arguments in the Service Test test. You can enter a constant value for each argument, or you can use parameter values from a QuickTest data table, random number parameter, or environment variable parameter.</p> <p>You insert a parameter value by clicking the parameterization button  and specifying it in the Value Configuration Options dialog box. For details, see "Value Configuration Options Dialog Box" on page 872.</p> <p>Note:</p> <ul style="list-style-type: none"> ➤ Parameter values are read-only in a Values cell. To modify a parameter value, click the X to clear the cell. Then enter a replacement value. ➤ XML structures are not relevant for this option.

Business Process Testing

This chapter includes:

Concepts

- Business Process Testing Overview on page 1743
- Business Process Testing Workflow on page 1748
- Business Process Testing Methodology on page 1750

Reference

- Quality Center Business Components Module on page 1755

Concepts

Business Process Testing Overview

Business Process Testing enables Subject Matter Experts to create tests using a keyword-driven methodology for testing.

Business Process Testing can incorporate the use of QuickTest with Quality Center and can be enabled by purchasing a specific Business Process Testing license. To work with Business Process Testing from within QuickTest, you must connect to a Quality Center project with Business Process Testing support.

This section provides an overview of the Business Process Testing model. For more information, see the *HP Business Process Testing User Guide* and the *HP QuickTest Professional for Business Process Testing User Guide*.

The Business Process Testing model is role-based, allowing non-technical Subject Matter Experts (working in Quality Center) to collaborate effectively with Automation Engineers (working in QuickTest Professional). Subject Matter Experts define and document business processes, business components, and business process tests, while Automation Engineers define the required resources and settings, such as shared object repositories, function libraries, and recovery scenarios. Together, they can build, data-drive, document, and run business process tests, without requiring programming knowledge on the part of the Subject Matter Expert.

Note: The role structure and the tasks performed by various roles in your organization may differ from those described here according to the methodology adopted by your organization. These roles are flexible and depend on the abilities and resources of the personnel using Business Process Testing. For example, the tasks of the Subject Matter Expert and the Automation Engineer may be performed by the same person. There are no product-specific rules or limitations controlling which roles must be defined in a particular organization, or which types of users can do which Business Process Testing tasks (provided that the users have the correct permissions).

The following user roles are identified in the Business Process Testing model:

- "Subject Matter Expert" on page 1745
- "Automation Engineer" on page 1746

 **Subject Matter Expert**

The Subject Matter Expert has specific knowledge of the application logic, a high-level understanding of the entire system, and a detailed understanding of the individual elements and tasks that are fundamental to the application being tested. This enables the Subject Matter Expert to determine the operating scenarios or business processes that must be tested and identify the key business activities that are common to multiple business processes.

Designing Business Components

- Using the Business Components module in Quality Center, the Subject Matter Expert creates business components that describe the specific tasks that can be performed in the application, and the condition or state of the application before and after those tasks. The Subject Matter Expert then defines the individual steps for each business component comprising the business process in the form of manual, or non-automated steps.
- During the design phase, the Subject Matter Expert works with the Automation Engineer to identify the resources and settings needed to automate the components, enabling the Automation Engineer to prepare them. When the resources and settings are ready, the Subject Matter Expert automates the manual steps by converting them to keyword-driven components. Part of this process entails choosing an application area for each component. The application area contains all of the required resource files and settings that are specific to a particular area of the application being tested. Associating each component with an application area enables the component to access these resources and settings.
- While defining components, Subject Matter Experts continue collaborating with the Automation Engineer. For example, they may request new operations (functions) for a component or discuss future changes planned for the component.

Creating Business Tests

- Using the Quality Center Test Plan module, the Subject Matter Expert combines the business components into business process tests using a sequence of components. For example, most applications require users to log in before they can access any of the application functionality. The Subject Matter Expert could create one business component that represents this login procedure. This component procedure can be used in many business process tests, resulting in easier and more cost-efficient test management. The Subject Matter Expert can also create flows, which are collections of business components, and insert them into business process tests.
- The Subject Matter Expert configures the values used for business process tests, runs them in test sets or configurations, and reviews the results. The Subject Matter Expert is also sometimes responsible for maintaining the testing steps for each of the individual business components.



Automation Engineer

The Automation Engineer is an expert in using an automated testing tool, such as QuickTest Professional. The Automation Engineer works with the Subject Matter Expert to identify the resources that are needed for the various business process tests.

The Automation Engineer then prepares the resources and settings required for testing the features associated with each specific component, and stores them in an application area within the same Quality Center project used by the Subject Matter Experts who create and run the business process tests for the specific application.

Using the resources created by the Automation Engineer, the Subject Matter Experts can automate component steps, and create and maintain components and business process tests.

Automation Engineers can also create, debug, and modify components in QuickTest, if required.

Business Process Testing Resource Structure

Each application area serves as a single entity in which to store all of the resources and settings required for a component, providing a single point of maintenance for all elements associated with the testing of a specific part of an application. Application areas generally include one or more shared object repositories, a list of keywords that are available for use with a component, function libraries containing automated functions (operations), recovery scenarios for failed steps, and other resources and settings that are needed for a component to run correctly. Components are linked to the resources and settings in the application area. Therefore, when changes are made in the application area, all associated components are automatically updated.

Application areas are stored in the Test Resources module in Quality Center, enabling visibility for Quality Center users. For example, a user can see at a glance which resources are associated with a particular application area, and which flows and components are using that application area.

The Automation Engineer uses QuickTest features and functionality to create these resources from within QuickTest. For example, in QuickTest, the Automation Engineer can create and populate various object repositories with test objects that represent the different objects in the application being tested, even before the application is fully developed. The Automation Engineer can then add repository parameters, and so forth, as needed. The Automation Engineer can manage the various object repositories using the Object Repository Manager, and merge repositories using the Object Repository Merge Tool. Automation Engineers can also use QuickTest to create and debug function libraries containing functions that use programming logic to encapsulate the steps needed to perform a particular task.

 **Business Process Testing Workflow**

Business Process Testing is flexible and does not enforce any one particular model for incorporating business processes into your testing environment. The actual workflow in an organization may differ for different projects, or at different stages of the application development life cycle.

A methodology of defining low-level components first and then designing business process tests based on the defined components is a legitimate methodology that may meet your needs. This section, however, presents a top-down methodology as a preferred alternative from the perspective of the subject matter expert with a high-level understanding of the entire system.

The top-down methodology presented here for working with Business Process Testing comprises:

- The high-level design and creation of a structure for business process tests. This means designing a modular business process test with reuse in mind. The utilization of components and flows promotes reuse and easier maintenance.

This part of the design phase is often done together by both the Subject Matter Expert and the Automation Engineer.

- The mid-level design, including the:
 - Creation of flows (sets of business components in a logical order that can be executed).
 - Creation of business components (reusable units that perform specific tasks in a business process).
 - Specifying criteria for more granular test coverage (requirements) as necessary.
 - Specifying different test configurations for testing different use-cases and for more "true-to-life" test coverage (requirements).

This part of the design phase is typically done by the Subject Matter Expert.

- The low-level implementation of business component content by creating component steps, setting up iterations (for business process tests, flows, and components), parameterizing, and when necessary, automating.

This part of the design phase can be performed by the Subject Matter Expert, the Automation Engineer, or both together, depending on available skills and resources.

- The execution of the business process tests and flows.

The top-down methodology therefore advocates the creation of business process testing entities according to the following hierarchy:

- Business process tests, which contain flows and/or business components.
- Flows, which contain business components.
- Business components, which contain steps.

Business process tests, business components, and flows are created using the Quality Center or HP ALM Business Components module and Test Plan module. Components can also be created using QuickTest, when QuickTest is connected to a Quality Center project with business process testing support.

Business Process Testing Methodology

Each end-to-end scenario that the Subject Matter Expert creates is a **business process test**. A business process test is composed of a sequence of **components** and/or **flows**. A flow is a collection of components. Each component performs a specific task. A component can pass data to a subsequent component.

This section includes:

- "Components Overview" on page 1750
- "Differences Between Business Components and QuickTest Tests" on page 1751
- "Components in QuickTest Professional" on page 1752
- "Business Process Tests and Flows in the Quality Center Test Plan Module" on page 1753
- "Running and Analyzing Business Process Tests" on page 1753

Components Overview

Components are easily-maintained reusable sets of steps that perform a specific task, and are the building blocks from which an effective business process testing structure can be produced. Each component represents a distinct part of a business process. For example, in most applications users need to log in before they can do anything else. A Subject Matter Expert can create one component that represents the login procedure for an application. That component can then be reused in different business process tests, resulting in easier maintenance, updating, and test management.

Components are comprised of steps. For example, the login component's first step may be to open the application. Its second step could be entering a user name. Its third step could be entering a password, and its fourth step could be clicking the **Enter** button.

You can also add checkpoint and output value steps to your component.

- A **checkpoint** is a verification that compares a current value for a specified property with the expected value for that property. This enables you to identify whether your application is functioning correctly. You can perform standard checkpoints and bitmap checkpoints on component steps. For more information, see "Checkpoints Overview" on page 591.
- An **output value** is a step in which one or more values are captured at a specific point in your component and stored for the duration of the run session. The values can later be used as input at a different point in the run session. For more information, see "Output Values" on page 781.

You can create and edit components in QuickTest by adding steps on any supported environment, parameterizing selected items, and enhancing the component by incorporating functions (operations) that encapsulate the steps needed to perform a particular task. In Quality Center, a Subject Matter Expert creates components and combines them into business process tests, which are used to check that the application behaves as expected.

Differences Between Business Components and QuickTest Tests

If you are already familiar with using QuickTest to create action-based tests, you will find that the procedures for creating and editing components are quite similar. However, due to the design and purpose of the component model, there are certain differences in the way you create, edit, and run components. The guidelines below provide an overview of these differences.

- A component is a single entity. It cannot contain multiple actions or have calls to other actions or to other components.
- When working with components, all external resource files are stored in the Test Resources module of the Quality Center project to which you are currently connected.
- The name of the component node in the Keyword View is the same as the saved component. You cannot rename the node.

- Business components are created in the Keyword View, not the Expert View.
- You associate resources via the component's application area, and not directly with the component.



Components in QuickTest Professional

Generally, components are created by Subject Matter Experts in Quality Center, although they can also be created in QuickTest.

In QuickTest, you create components by adding steps manually—if the object repository is populated and the required operations are available. You can also create components in QuickTest by recording steps on any supported environment. You can parameterize selected items. You can also view and set options specific to components.

QuickTest enables you to create and modify two types of components: **business components** and **scripted components**. A business component is an easily-maintained, reusable unit comprising one or more steps that perform a specific task. A scripted component is an automated component that can contain programming logic. Scripted components share functionality with both test actions and business components.

For example, you can use the Keyword View, the Expert View, and other QuickTest tools and options to create, view, modify, and debug scripted components in QuickTest. Due to their complexity, scripted components can be edited only in QuickTest.

In Quality Center, the Subject Matter Expert can open components created in QuickTest. The Subject Matter Expert can then view and edit business components, but can only view the details for scripted components.

Business Process Tests and Flows in the Quality Center Test Plan Module

The Subject Matter Expert first creates a business process test or flow in the Test Plan module. A **flow** is a collection of business components in a fixed sequence that can be used as a unit in multiple business process tests. Flows are available only in HP ALM / Quality Center.

To populate the business process test or flow, the Subject Matter Expert then selects (drags and drops) the relevant components and configures their run settings.

Each component can be used differently by different business process test configurations or flows. For example, for each test configuration, the component can use different input parameter values or run a different number of iterations.

If, while creating a business process test or flow, the Subject Matter Expert realizes that a component has not been defined for an element that is necessary for the business process test or flow, the Subject Matter Expert can submit a component request from the Test Plan module.

Running and Analyzing Business Process Tests

You can use the run and debug options in QuickTest to run and debug an individual component.

You can debug a business process test by running the test from the Test Plan module in Quality Center. When you choose to run from this module, you can choose which components to run in debug mode. (This pauses the run at the beginning of a component.)

When the business process test has been debugged and is ready for regular test runs, the Subject Matter Expert runs it from the Test Lab module similar to the way any other test is run in Quality Center. Before running the test, the Subject Matter Expert can define run-time parameter values and iterations using the **Iterations** column in the Test Lab module grid.

Note: When you run a business process test from Quality Center, the test run may also be influenced by settings in the QuickTest Remote Agent. For more information on the QuickTest Remote Agent, see "QuickTest Remote Agent Preferences" on page 1613.

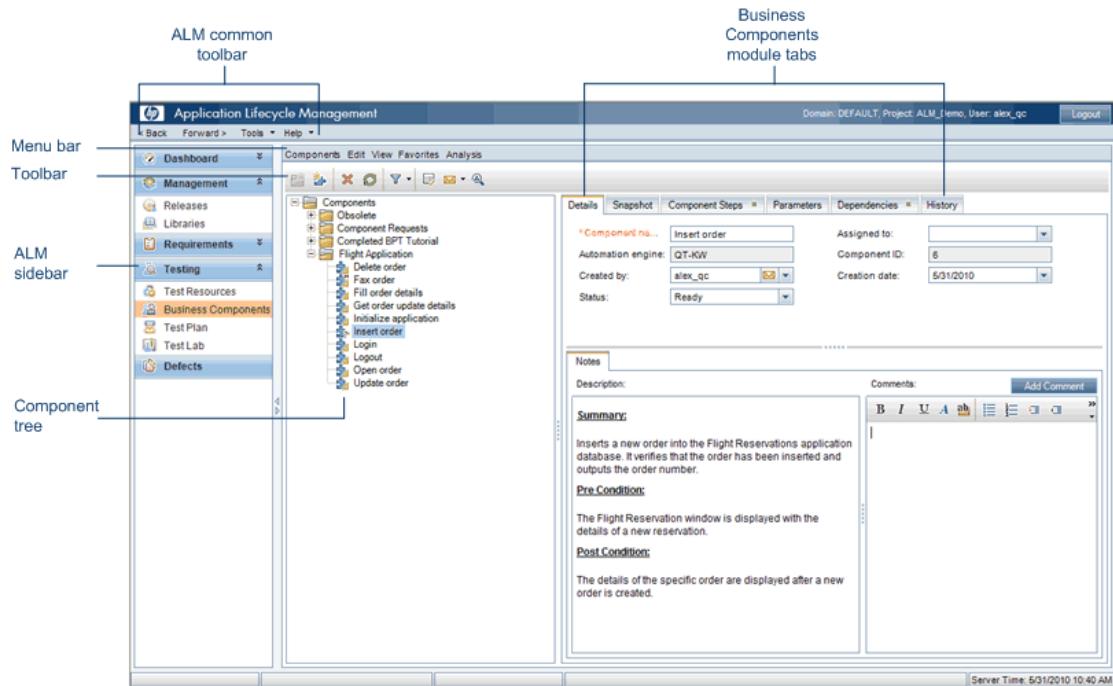
From the Test Lab module, you can view the run results of each configuration of the business process test. The results include the value of each parameter, and the results of individual steps reported by QuickTest.

You can click the **Launch Report** link to open the complete QuickTest run results. The hierarchical results contains all the different iterations and components within the business process test run.

Reference

Quality Center Business Components Module

The Subject Matter Expert can create a new component and define it in the HP ALM Business Components module.



The HP ALM Business Components module enables you to provide a complete overview of the component's content and includes the following tabs:

- **Details.** Provides a general summary of the component's purpose or goals, and the condition of the application before and after a component is run (its pre-conditions and post-conditions). You can specify details and implementation requirements for the currently selected business component.

- **Snapshot.** Displays an image that provides a visual cue or description of the component's purpose or operations. You can capture a snapshot image from the application and attach it to the currently selected business component.
- **Component Steps.** Enables you to create or view the manual steps of your business component, and to automate it if required.
- **Parameters.** Specifies the input and output component parameters and parameter values for the business component. This allows the component to receive data from an external source and to pass data to other components or flows.
- **Dependencies tab.** Displays a list of assets that are linked to the currently selected business component, enabling you to view the dependency relationships between components, tests, flows, and resources (including application areas).
- **History tab.** Displays a log of changes made to the component.
- **Live Analysis tab.** When a folder is selected, this tab is available for creating a graphical representation of data related to business components.

60

HP Performance Testing and Business Service Management Products

This chapter includes:

Concepts

- HP Performance Testing and Business Service Management Products Overview on page 1758
- Designing Tests for HP Performance Testing Products on page 1761
- Running Tests from HP Performance Testing Products on page 1762
- Designing Tests for HP Business Process Monitor on page 1763
- Running Tests from HP Business Process Monitor on page 1764
- Measuring Transactions on page 1765
- Silent Test Runner on page 1768

Tasks

- How to Insert and Run Tests in Performance Center and LoadRunner on page 1770

Reference

- End Transaction Dialog Box on page 1771
- Start Transaction Dialog Box on page 1772
- Silent Test Runner Dialog Box on page 1774

Concepts

HP Performance Testing and Business Service Management Products Overview

QuickTest enables you to create complex tests that examine the full spectrum of your application's functionality to confirm that every element of your application works as expected in all situations.

After you use QuickTest to create and run a suite of tests that test the functional capabilities of your application, you may want to test how much load your application can handle or to monitor your application as it runs.

- **HP performance testing products (LoadRunner and Performance Center)** test the performance and reliability of an entire system under controlled and peak load conditions. To generate load, these performance testing products run hundreds or thousands of virtual users. These virtual users provide consistent, repeatable, and measurable load to exercise your application just as real users would.
- **HP Business Service Management (formerly HP Business Availability Center)** enables real-time monitoring of the end user experience. Business Process Monitor runs virtual users to perform typical activities on the monitored application.

If you have already created and perfected a test in QuickTest that is a good representation of your users' actions, you may be able to use your QuickTest test as the basis for performance testing and application management activities.

You can use **Silent Test Runner** to check in advance that a QuickTest test will run correctly from LoadRunner, Performance Center, and Business Process Monitor.

QuickTest Features for Use with Performance Testing and Business Service Management

QuickTest offers several features that are designed specifically for integration with LoadRunner, Performance Center, and Business Process Monitor.

Note: These products are designed to run tests using virtual users representing many users simultaneously performing standard user operations, some QuickTest features may not be available when integrating these products with QuickTest.

You can use the **Services** object and its associated methods to insert statements that are specifically relevant to Performance Testing and Business Service Management. These include:

AddWastedTime	EndTransaction	SetTransaction
EndDistributedTransaction	LogMessage	SetTransactionStatus
GetEnvironmentAttribute	Rendezvous	ThinkTime
StartDistributedTransaction	StartTransaction	UserDataTableUserDataTable

For details on these methods, see the **Services** section of the *HP QuickTest Professional Object Model Reference* and your HP performance testing or Business Service Management documentation.

For details on transactions, see "Measuring Transactions" on page 1765.

For details on inserting StartTransaction and EndTransaction statements using QuickTest menu or toolbar options, see "Start Transaction Dialog Box" on page 1772 and "End Transaction Dialog Box" on page 1771.

Advantages of Running QuickTest Tests in LoadRunner

The main advantages of running QuickTest tests in LoadRunner are:

- To check how your application's functionality is affected by heavy load
- To measure the response time that a typical user experiences on the client side while your application is under load (end-to-end response time)

For example, you can add QuickTest tests to specific points in a LoadRunner scenario to confirm that the application's functionality is not affected by the extra load at those sensitive points.

Another advantage of using a GUI Vuser script as part of your LoadRunner scenario is that the GUI Vuser script runs on your screen during the scenario, enabling you to watch the actual steps executed by the Vuser in real time.

Designing Tests for Use HP Performance Testing and HP Business Service Management Products

If you plan to use the same test in both QuickTest and LoadRunner, Performance Center, and/or Business Process Monitor, you should take into account the different options supported in each product as you design your test. For details, see:

- "Designing Tests for HP Performance Testing Products" on page 1761
- "Designing Tests for HP Business Process Monitor" on page 1763

Running Tests from HP Performance Testing and HP Business Service Management Products

The run mechanisms used in all HP Performance Testing and HP Business Service Management products are the same. This means that you can create tests that are compatible with LoadRunner, Performance Center, and Business Process Monitor, enabling you to take advantage of tests or test segments that have already been designed and debugged in QuickTest.

For example, you can add QuickTest tests to specific points in a performance test to confirm that the application's functionality is not affected by the extra load at those sensitive points. You can also run QuickTest tests on Business Process Monitor to simulate end user experience and ensure that your application is running correctly and in a timely manner.

If you plan to use the same test in both QuickTest and LoadRunner, Performance Center, and/or Business Process Monitor, you should take into account the different options supported in each product before you run your test. For details, see:

- "Running Tests from HP Performance Testing Products" on page 1762
- "Running Tests from HP Business Process Monitor" on page 1764

Designing Tests for HP Performance Testing Products

Consider the following guidelines when designing tests for use with performance testing products:

- The QuickTest tests you use with LoadRunner and Performance Center should be simple, designed to pinpoint specific operations, and should avoid using external actions and references to other external files (including resources stored in Quality Center). Also, when working with action iterations, corresponding StartTransaction and EndTransaction statements must be contained within the same action.
- Every QuickTest test must contain at least one transaction to provide useful information in the performance test. LoadRunner and Performance Center use only the data that is included within a transaction, and ignore any data in a test outside of a transaction.
- Do not include references to external actions or other external resources (including resources stored in Quality Center), such as an external data table file, environment variable file, shared object repositories, function libraries, and so forth. This is because LoadRunner or Performance Center may not have access to the external action or resource.
(However, if the resource can be found on the network, QuickTest will use it. For example, you can try defining external resources via an absolute path, or by adding them as supplementary files and transferring them to Load Generator in the QuickTest test folder.)
- Make sure that the last step(s) in the test closes the application being tested, as well as any child processes that are running. This enables the next iteration of the test to open the application again.

For details on working with LoadRunner or Performance Center, see your HP performance testing documentation.

Running Tests from HP Performance Testing Products

Consider the following guidelines when running QuickTest tests from HP Performance Testing Products:

- You can run only one GUI Vuser concurrently per computer. (A GUI Vuser is a Vuser that runs a QuickTest test.)
- Ensure that QuickTest is closed on the QuickTest computer before running a QuickTest test in Performance Center or LoadRunner.
- The settings in the LoadRunner or Performance Center Run-time Settings dialog box are not relevant for QuickTest tests.
- You cannot use the **ResultDir** QuickTest environment variable when running a performance test.
- Transaction breakdown is not supported for tests (scripts) created with QuickTest.
- QuickTest cannot run on a computer that is:
 - logged off or locked. In these cases, consider running QuickTest on a terminal server.
 - already running a QuickTest test. Make sure that the test is finished before starting to run another QuickTest test.

Tip: You can simulate how the test will run from a performance testing product by using Silent Test Runner. For details, see "Silent Test Runner" on page 1768.

Designing Tests for HP Business Process Monitor

Note: As of Business Service Management 9.0, profiles are no longer used. Instead, Business Process Monitor uses Business Transaction Flows, which are included in the parent application's run unit and can be run as part of that unit or independently, as needed.

Consider the following guidelines when designing tests for use with Business Process Monitor:

- The QuickTest tests you use with Business Process Monitor should be simple, designed to pinpoint specific operations, and should avoid using external actions and references to other external files (including resources stored in Quality Center). Also, when working with action iterations, corresponding StartTransaction and EndTransaction statements must be contained within the same action.
- Every QuickTest test must contain at least one transaction to provide useful information in Business Process Monitor. Business Process Monitor uses only the data that is included within a transaction, and ignores any data in a test outside of a transaction.
- Business Process Monitor does not support running QuickTest Professional tests that require access to external resources, including resources stored in Quality Center (such as a shared object repository, function library, external data table, external actions, and so forth). Tests that require external resources may fail to run on Business Process Monitor. (However, if the resource can be found on the network, QuickTest will use it.)
- Make sure that the last steps in the test close the application being tested, as well as any child processes that are running. This cleanup step enables the next test run to open the application again.
- When measuring a distributed transaction over two different Business Process Monitor profiles or Business Transaction Flows (depending on the version), the profile with the StartDistributedTransaction statement must be run before the profile with the associated EndDistributedTransaction.

- When measuring distributed transactions, make sure that you relate the tests to a single Business Process Monitor instance. Business Process Monitor searches for the end transaction name in all instances, and may close the wrong distributed transaction if it is included in more than one instance.
- When measuring a distributed transaction over two Business Process Monitor profiles, make sure that the timeout value you specify is large enough so that the profile or Business Transaction Flow (depending on the version) that contains the StartDistributedTransaction step and all the profiles that run before the profile that contains the EndDistributedTransaction step, will finish running in a time that is less than the value of the specified timeout.

Running Tests from HP Business Process Monitor

Consider the following guidelines when running QuickTest tests from HP Business Process Monitor:

- Before you try to run a QuickTest test in Business Process Monitor, check which versions of QuickTest are supported by your version of Business Process Monitor. For details, see the Business Process Monitor documentation.
- To run a QuickTest test in Business Process Monitor, QuickTest must be installed and closed on the Business Process Monitor computer
- Business Process Monitor can run only one QuickTest test at a time. Make sure that the previous QuickTest run session is finished before starting to run another QuickTest test.
- Transaction breakdown is not supported for tests created with QuickTest.
- QuickTest tests must be zipped before uploading them to Business Service Management Admin.

If you make changes to your local copy of a QuickTest test after uploading it to Business Service Management, upload the zipped test again to enable Business Process Monitor to run the test with your changes.

- QuickTest cannot run tests on a computer that is logged off, locked, or running QuickTest as a non-interactive service.

- You cannot use the **ResultDir** QuickTest environment variable when running a test in Business Process Monitor.

For details on working with Business Service Management, see the relevant documentation—specifically the sections describing QuickTest Professional scripts in the Business Process Monitor Administration guide and the End User Management guide.

Tip: You can simulate how the test will run from Business Process Monitor by using Silent Test Runner. For details, see "Silent Test Runner" on page 1768.



Measuring Transactions

You can measure how long it takes to run a section of your test by defining **transactions**. A transaction represents the process in your application that you are interested in measuring. Your test must include transactions to be used by LoadRunner, Performance Center, or the Business Process Monitor. These products use only the data that is included within a transaction, and ignore any data in a test outside of a transaction.

You define transactions within your test by enclosing the appropriate sections of the test with **start** and **end** transaction statements. For example, you can define a transaction that measures how long it takes to reserve a seat on a flight and for the confirmation to be displayed on the client's terminal.

During the run session, the StartTransaction step signals the beginning of the time measurement. The time measurement continues until the EndTransaction step is reached. The test results for the EndTransaction step include the transaction's name, end status, total duration, and wasted time.

During a run session, QuickTest runs background processes that add to the time it takes to run a test. Wasted time is the time within the total duration that was added as a result of QuickTest running the transaction. If the application ran the transaction without QuickTest, the total duration would equal the total duration minus the wasted time.

Note: If you start a transaction while there is already open transaction with the same name, the previous transaction is ended with **Fail** status and then the new transaction is started.

There is no limit to the number of transactions that can be added to a test.

Tip:

You can:

- Insert a transaction within a transaction.
 - Insert a variety of transaction-related statements using the Step Generator or Expert View. For details, see the **Services** section of the *HP QuickTest Professional Object Model Reference*.
 - Enter Start Transaction and End Transaction steps using options in the QuickTest window. For details, see "Start Transaction Dialog Box" on page 1772 and "End Transaction Dialog Box" on page 1771.
-

For details on the statements you can use in transactions, see the *HP QuickTest Professional Object Model Reference*.

Example of a test with a transaction:

Part of a sample test with a transaction is shown below, as it is displayed in the Keyword View:

Start transaction			
	Services	StartTransaction	"ReserveSeat"
	Find a Flight: Mercury		Start the "ReserveSeat" transaction.
	fromPort	Select	Select the "London" item in the "fromPort" list.
	toPort	Select	Select the "Frankfurt" item in the "toPort" list.
	toDay	Select	Select the "12" item in the "toDay" list.
	servClass	Select	Select radio button "Business" in the "servClass" radio button group.
	airline	Select	Select the "Blue Skies Airlines" item in the "airline" list.
	findFlights	Click	Click the "findFlights" image.
	Select a Flight: Mercury...		
	outFlight	Select	Select radio button "Blue Skies Airlines" in the "outFlight" radio button group.
	inFlight	Select	Select radio button "Blue Skies Airlines" in the "inFlight" radio button group.
	reserveFlights	Click	Click the "reserveFlights" image.
	Services	EndTransaction	"ReserveSeat"
End transaction			End the "ReserveSeat" transaction.
	Book a Flight: Mercury_2		

The same part of the test is displayed in the Expert View as follows:

```

Services.StartTransaction "ReserveSeat"
Browser("Welcome: Mercury Tours").Page("Find a Flight: Mercury").
    WebList("fromPort").Select "London"
Browser("Welcome: Mercury Tours").Page("Find a Flight: Mercury").
    WebList("toPort").Select "Frankfurt"
Browser("Welcome: Mercury Tours").Page("Find a Flight: Mercury").
    WebList("toDay").Select "12"
Browser("Welcome: Mercury Tours").Page("Find a Flight: Mercury").
    WebRadioGroup("servClass").Select "Business"
Browser("Welcome: Mercury Tours").Page("Find a Flight: Mercury").
    WebList("airline").Select "Blue Skies Airlines"
Browser("Welcome: Mercury Tours").Page("Find a Flight: Mercury").
    Image("findFlights").Click 65,12
Browser("Welcome: Mercury Tours").Page("Select a Flight: Mercury").
    WebRadioGroup("outFlight").Select "Blue Skies Airlines"
Browser("Welcome: Mercury Tours").Page("Select a Flight: Mercury").
    WebRadioGroup("inFlight").Select "Blue Skies Airlines"
Browser("Welcome: Mercury Tours").Page("Select a Flight: Mercury").
    Image("reserveFlights").Click 46,8
Services.EndTransaction "ReserveSeat"

```

Silent Test Runner

Silent Test Runner enables you to simulate the way a QuickTest test runs from LoadRunner, Performance Center, and Business Service Management. When you run a test using Silent Test Runner, it runs without opening the QuickTest user interface, and the test runs at the same speed as when it is run from LoadRunner, Performance Center, or Business Service Management. At the end of the test run, you can view information about the test run and transaction times. For details, see "Test Run Information for Silent Runs" on page 1768.

You can also use Silent Test Runner to verify that your QuickTest test is compatible with LoadRunner, Performance Center, and Business Service Management. A test will fail when run using Silent Test Runner if it uses a feature that is not supported by these products. For details on features that are not supported, see "Designing Tests for HP Performance Testing Products" on page 1761, "Running Tests from HP Performance Testing Products" on page 1762, "Designing Tests for HP Business Process Monitor" on page 1763, and "Running Tests from HP Business Process Monitor" on page 1764.

Test Run Information for Silent Runs

Silent Test Runner provides test run information in log files. Each test generates a test run log, and any test with transactions generates an additional transaction summary.

Viewing the Test Run Log

The test run log is saved as **output.txt** in the <QuickTest Professional>\Tests\<test name> folder. A log file is saved for each test run with Silent Test Runner and is overwritten when you rerun the test. To open the log file, click **Test Run Log**.

The log file displays information about the test run. For example, information is shown about each iteration, action call, step transaction, failed step, and so forth. Each line displays a message or error ID. For details on message and error codes in the log file, see your Performance Center or Business Service Management documentation.

Viewing the Transaction Summary

The transaction summary is saved as **transactions.txt** in the **<QuickTest Professional>\Tests\<test name>** folder. A transaction summary is saved for each test that includes transactions and is overwritten when you rerun the test. To open the log file, click **Transaction Summary**. The transaction summary displays a line for each transaction in the test. For each transaction, the status is displayed together with the total duration time and any wasted time (in seconds). The transaction measurements in Silent Test Runner are exactly the same as if the test was run from LoadRunner, Performance Center, or Business Service Management.

Note:

- A transaction summary is available only for a test that contains transactions ending with an `EndTransaction` statement. If a transaction started but did not end because of test failure, it is not included in the transaction summary.
 - Distributed transactions (transactions that start in one test and end in another) are not reported in the transaction summary but are included in the test run log.
 - Any transaction information included in the transaction summary is also included in the test run log.
-

Tasks

How to Insert and Run Tests in Performance Center and LoadRunner

The following are various tasks that you can perform when working with HP Performance Center and LoadRunner.

To insert a QuickTest test in a LoadRunner scenario:

In the Controller Open Test dialog box, browse to the test folder and select **QuickTest Tests** in the **Files of type** box (or select **Astra Tests** in LoadRunner versions older than 9.0). This enables you to view QuickTest tests in the folder.

To use a QuickTest test in Performance Center:

Create a zipped version of the QuickTest test, and upload it to the Performance Center User Site Vuser Scripts Page.

To run multiple GUI Vusers on the same application:

Open a terminal server session for each GUI Vuser. For more details, refer to the HP performance testing documentation.

For details on working with LoadRunner or Performance Center, see your HP performance testing documentation.

Reference

End Transaction Dialog Box

This dialog box enables you to insert a step that signals the end of the time measurement for a transaction.



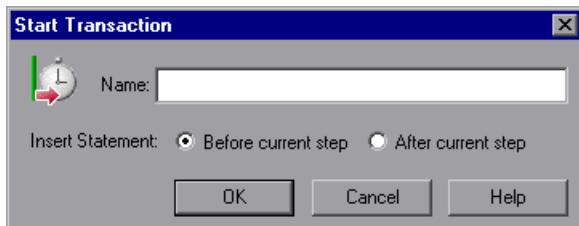
To access	Use one of the following: <ul style="list-style-type: none">➤ Select the Insert > End Transaction menu command.➤ Click the End Transaction toolbar button .
Important information	<ul style="list-style-type: none">➤ There may be cases in which you want to instruct QuickTest to perform all the steps in a transaction, even though an error occurs during the run session. To do this: In the Run pane of the Test Settings dialog box (File > Settings > Run node), select proceed to next step from the When error occurs during run session list.➤ You can also create recovery scenarios or other error handling steps to address these cases. For details, see Chapter 49, "Recovery Scenarios."
See also	<ul style="list-style-type: none">➤ "Measuring Transactions" on page 1765➤ "Start Transaction Dialog Box" on page 1772

User interface elements are described below:

UI Elements	Description
Name	<p>The name of the transaction you want to end.</p> <p>The list contains the name of all transactions that start prior to the selected step in the current action.</p>
Insert Statement	<p>Indicates where the EndTransaction step will be inserted in relation to the selected step.</p> <p>Select Before current step or After current step.</p>

Start Transaction Dialog Box

This dialog box enables you to insert a step that signals the beginning of the time measurement for a transaction.



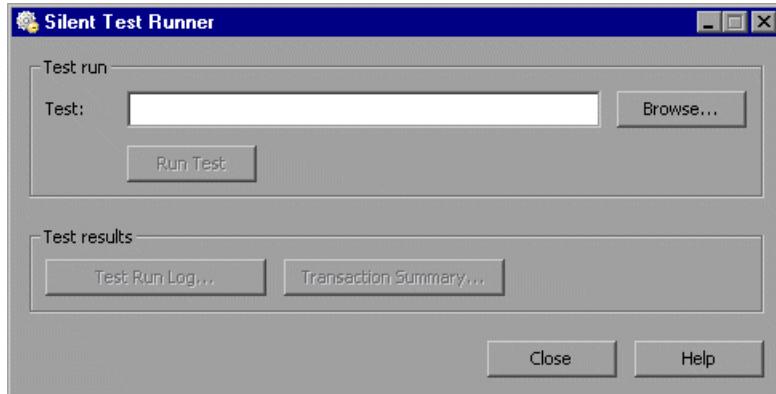
To access	<p>Use one of the following:</p> <ul style="list-style-type: none"> ► Select the Insert > Start Transaction menu command. ► Click the Start Transaction toolbar button .
See also	<ul style="list-style-type: none"> ► "Measuring Transactions" on page 1765 ► "End Transaction Dialog Box" on page 1771

User interface elements are described below:

UI Elements	Description
Name	The name of the transaction you want to measure. Note: You cannot include spaces in a transaction name.
Insert Statement	Indicates where the StartTransaction step will be inserted in relation to the selected step. Select Before current step or After current step .

Silent Test Runner Dialog Box

This dialog box enables you to simulate the way a QuickTest test runs from LoadRunner and Business Service Management and to verify that your QuickTest test is compatible with LoadRunner and Business Service Management.



To access	Select the Start > Programs > QuickTest Professional > Tools > Silent Test Runner menu command.
Important information	<ul style="list-style-type: none"> ➤ You cannot run Silent Test Runner if QuickTest is already open or another test is currently running. You must close QuickTest and wait for its process to end before running your test using Silent Test Runner. ➤ You can invoke only one instance of Silent Test Runner and you can specify only one test to run. ➤ You cannot use the ResultDir QuickTest environment variable when running a test from Silent Test Runner.
See also	<ul style="list-style-type: none"> ➤ "Silent Test Runner" on page 1768 ➤ "Test Run Information for Silent Runs" on page 1768

User interface elements are described below:

UI Elements	Description
Test	<p>The full file system path of the test you want to run.</p> <p>Note: To specify a network path, you must map the network drive.</p>
Run Test	<p>Runs the test specified in the Test box.</p> <p>When you click this button, the test runs without opening the QuickTest user interface. The text Running test... is displayed next to the Run Test button while the test is running.</p> <p>When the test run finishes, the text Running test... is replaced with the text Test run completed. If Silent Test Runner was unable to run your test, the text Test could not be run is displayed.</p> <p>Note: After you start a test run, you cannot stop the test run from Silent Test Runner. Even if you close Silent Test Runner, the test continues to run. To end the run, end the mdrv.exe process manually.</p>
Test Run Log	<p>Displays the most recent run log for the selected test. Each time you run a test with Silent Test Runner, the previous log file is overwritten with the current run results.</p> <p>(Enabled only when the selected test has run with the Silent Test Runner at least once.)</p> <p>For details, see "Viewing the Test Run Log" on page 1768.</p>
Transaction Summary	<p>Displays the summary of the transactions in the test.</p> <p>(Enabled only when the selected test contains at least one transaction and the test has run with the Silent Test Runner at least once.)</p> <p>For details, see "Viewing the Transaction Summary" on page 1769.</p>

Part XIII

Appendices

A

Naming Conventions

The following table lists the restrictions to consider when naming items in QuickTest:

Item	Naming Convention
Action	<ul style="list-style-type: none">► Must be unique in the test.► Cannot begin or end with a space.► Cannot exceed 1,023 characters.► Cannot contain the following characters: ` \ / : * ? " < > % ! { }
Action parameter	<ul style="list-style-type: none">► Case-sensitive.► Must begin with a letter► Cannot contain spaces.► Cannot contain the following characters: ! @ # \$ % ^ & * () + = [] \ { } ; ' : " , . / < >
Checkpoint	<ul style="list-style-type: none">► Cannot begin or end with a space.► Cannot contain " (double quotation mark).► Cannot contain the following character combinations:<ul style="list-style-type: none">► :=► @@
Data Table file	Quality Center: cannot contain the following characters: ! % * { } \ ' : " / < > ? ; ,

Item	Naming Convention
Data Table > Parameter name (column header)	<ul style="list-style-type: none"> ► Must be unique in the sheet. ► Must begin with a letter or underscore. ► Can only contain the following: <ul style="list-style-type: none"> ► Letters ► Numbers ► Periods ► Underscores
Environment variable (parameter)	<ul style="list-style-type: none"> ► Must begin with a letter. ► Can only contain the following: <ul style="list-style-type: none"> ► Letters ► Numbers ► Underscores
Environment variables file	<p>File system: Cannot contain the following characters: ! % * { } \ ' : " / < > ? ;</p> <p>Quality Center: Cannot contain the following characters: ! % * { } \ ' : " / < > ? ; ,</p>
Function library file	<p>File system: Cannot contain the following characters: ! % * { } \ ' : " / < > ? ;</p> <p>Quality Center: Cannot contain the following characters: ! % * { } \ ' : " / < > ? ; ,</p>
Function / Function argument	<ul style="list-style-type: none"> ► Cannot contain non-English letters or characters. ► Must begin with a letter. ► Cannot contain spaces or any of the following characters: ! @ # \$ % ^ & * () + = [] \ { } ; ' : " " , / < > ?
Object repository file	<p>File system: Cannot contain the following characters: ! % * { } \ ' : " / < > ? ;</p> <p>Quality Center: Cannot contain the following characters: ! % * { } \ ' : " / < > ? ; ,</p>

Item	Naming Convention
Output value	<ul style="list-style-type: none"> ➤ Cannot begin or end with a space. ➤ Cannot contain " (double quotation mark). ➤ Cannot contain the following character combinations: <ul style="list-style-type: none"> ➤ := ➤ @@
Quality Center file or folder name	Cannot exceed 90 characters (including the path).
Recovery Scenario	<p>File system: Cannot contain the following characters: ! % * { } \ ' : " / < > ? ;</p> <p>Quality Center: Cannot contain the following characters: ! % * { } \ ' : " / < > ? ; ,</p>
Test name	<ul style="list-style-type: none"> ➤ Cannot exceed 220 characters (including the path). ➤ Cannot begin or end with a space. ➤ Cannot contain the following characters: \\ / : * ? " < > % ' ; ➤ Cannot contain multibyte punctuation symbols and other multibyte special characters, such as multibyte question marks, multibyte spaces, and multibyte brackets.
Test object class (extensibility only)	<ul style="list-style-type: none"> ➤ Cannot contain non-English letters or characters. ➤ Cannot contain spaces or any of the following characters: ! @ # \$ % ^ & * () + = - [] \ { } ; ' : " " , / < > ?
Test object name	<ul style="list-style-type: none"> ➤ must be unique within the same class and hierarchy in the object repository ➤ Cannot begin or end with a space. ➤ Cannot contain " (double quotation mark). ➤ Cannot contain the following character combinations: <ul style="list-style-type: none"> ➤ := ➤ @@

Appendix A • Naming Conventions

Item	Naming Convention
Test object identification properties	<ul style="list-style-type: none">▶ Cannot contain non-English letters or characters.▶ Cannot contain spaces or any of the following characters: ! @ # \$ % ^ & * () + = [] \ { } ; ' : " " , / < > ?
Test object method / Test object method argument	<ul style="list-style-type: none">▶ Cannot contain non-English letters or characters.▶ Cannot contain spaces or any of the following characters: ! @ # \$ % ^ & * () + = [] \ { } ; ' : " " , / < > ?

B

Supported Checkpoints and Output Values Per Add-in

The tables in this chapter show the categories of checkpoints and output values that are supported by QuickTest Professional for each add-in.

For more information about using checkpoints and output values in a specific add-in, see the relevant add-in section.

This chapter includes:

- Supported Checkpoints on page 1784
- Supported Output Values on page 1786

Supported Checkpoints

Table Legend

- S: Supported
- NS: Not Supported
- NA: Not Applicable

For additional information, see "Footnotes" on page 1785.

	Accessibility	Bitmap	Database	Image	Page	Standard	Table	Text	Text Area	XML (Application)	XML (Resource)
ActiveX	NS	S	NA	NS	NA	S	S	S	S	NA	NA
Delphi	NS	S	NA	NS	NA	S	S	S	S	NA	NA
Java	NA	S	NA	NA	NA	S	S	S	S ⁶	NA	NA
.NET Web Forms ⁵	S	S	NA	NA	NA	S	S	S ⁸	S ⁸	S	S
.NET Windows Forms	NA	S	NA	NA	NA	S	S	S ⁸	S ⁸	NA	NA
Oracle	NA	S	NA	NA	NA	S	S	NS	NS	NA	NA
PeopleSoft	S	S	NA	S	S	S	S	S ³	NS	S	S
PowerBuilder ⁴	NS	S	NA	NS	NA	S	S	S	S	NA	NA
SAP Web	S	S	NA	S	S	S	S	S	NS	S	S
SAP Windows	S ⁷	S	NA	S ⁷	S ⁷	S	S	S ⁷	NS	S ⁷	NA
Siebel	S	S	NA	S	S	S	S	S	NS	S	S
Silverlight	NA	S	NA	NA	NA	S	S	S	S	NA	NA
Standard Windows	NS	S	NA	NS	NA	S	S	S	S	NA	NA
Stingray	NA	S	NA	NA	NA	S	S	S	S	NA	NA
Terminal Emulator	NA	S	NA	NA	NA	S	NA	NA	NA	NA	NA
Visual Age	NA	S	NA	NA	NA	S	S	S	S	NA	NA

	Accessibility	Bitmap	Database	Image	Page	Standard	Table	Text	Text Area	XML (Application)	XML (Resource)
Visual Basic	NS	S	NA	NS	NA	S	S	S	S	NA	NA
Web²	S	S	NA	S	S	S	S	S ³	NS	S	NA
Web Services	NA	NA	NA	NA	NA	S	NA	NA	NA	S	NA
WPF	NA	S	NA	NA	NA	S	S	S	S	NA	NA

Footnotes

¹ Only standard and bitmap checkpoints are supported for business components.

² When creating checkpoints for Web objects in components, only bitmap checkpoints and standard checkpoints are available.

³ Checkpoints are supported only for Page, Frame, and ViewLink objects.

⁴ When you insert a checkpoint on a PowerBuilder DataWindow control, QuickTest treats it as a table and opens the Table Checkpoint Properties dialog box (not supported for components).

⁵ For .NET Web Forms, text checkpoints for WbfTreeView, WbfToolbar, and WbfTabStrip objects are not supported.

⁶ The text area checkpoint mechanism for Java Applet objects is disabled by default. You can enable it (for tests only) in the Advanced Java Options dialog box.

⁷ This is supported only when QuickTest records HTML elements using the Web infrastructure, but not when it records using the SAPGUI Scripting Interface (as selected in the SAP pane of the Options dialog box).

⁸ This is supported only when QuickTest is configured to use the OCR (optical character recognition) mechanism.

Supported Output Values

Table Legend

- S: Supported
- NS: Not Supported
- NA: Not Applicable

For additional information, see "Footnotes" on page 1787.

	Accessibility	Bitmap	Database	Page	Standard	Table	Text	Text Area	XML (Application)	XML (Resource)
ActiveX	NS	NA	NA	NA	S	S	S	S	NA	NA
Delphi	NS	NA	NA	NA	S	S	S	S	NA	NA
Java	NA	NA	NA	NA	S	NA	S	S ⁵	NA	NA
NET Web Forms	NA	NA	NA	S	S	S	S ⁷	S ⁷	NA	NA
NET Windows Forms	NA	NA	NA	NA	S	S	S ⁷	S ⁷	NA	NA
Oracle	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA
PeopleSoft	NA	NA	NA	S	S	S	S ³	NS	S	S
PowerBuilder ⁴	NA	NA	NA	NA	S	NA	S	S	NA	NA
SAP Web	NA	NA	NA	S	S	S	S	NS	S	S
SAP Windows	NA	NA	NA	S ⁶	S	S	S ⁶	NS	S ⁶	S
Siebel	NA	NA	NA	S	S	S	S	NS	S	S
Silverlight	NA	NA	NA	NA	S	S	S	S	NA	NA
Standard Windows	NA	NA	NA	NA	S	S	S	S	NA	NA
Stingray	NA	NA	NA	NA	S	S	S	S	NA	NA
Terminal Emulator	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA
Visual Age	NA	NA	NA	NA	NA	S	S	S	NA	NA

	Accessibility	Bitmap	Database	Page	Standard	Table	Text	Text Area	XML (Application)	XML (Resource)
Visual Basic	NA	NA	NA	NA	S	NA	S	S	NA	NA
Web²	NA	NA	NA	S	S	S	S ³	NS	S	NA
Web Services	NA	NA	NA	NA	NA	NA	NA	NA	NA	S
WPF	NA	NA	NA	NA	S	S	S	S	NA	NA

Footnotes

¹ Only standard and bitmap output values are supported for business components.

² When creating output values for Web objects in components, only standard output values are available.

³ Output values are supported only for Page, Frame, and ViewLink objects.

⁴ When you insert an output value step on a PowerBuilder DataWindow control, QuickTest treats it as a table and opens the Table Output Value Properties dialog box (not supported for components).

⁵ The text area output mechanism for Java Applet objects is disabled by default. You can enable it (for tests only) in the Advanced Java Options dialog box.

⁶ This is supported only when QuickTest records HTML elements using the Web infrastructure, but not when it records using the SAPGui Scripting Interface (as selected in the SAP pane of the Options dialog box).

⁷ This is supported only when QuickTest is configured to use the OCR (optical character recognition) mechanism.

Appendix B • Supported Checkpoints and Output Values Per Add-in

C

Frequently Asked Questions

This chapter answers some of the questions that are asked most frequently by advanced users of QuickTest. The questions and answers are divided into the following sections:

This chapter includes:

- Creating Tests on page 1790
- Programming in the Expert View on page 1792
- Working with Dynamic Content on page 1794
- Advanced Web Issues on page 1796
- Standard Windows Environment on page 1800
- Test Maintenance on page 1802
- Testing Localized Applications on page 1805
- Improving QuickTest Performance on page 1806

Creating Tests

This section includes answers to the following questions:

- "How can I record on objects or environments not supported by QuickTest?" on page 1790
- "How can I launch an application from a test?" on page 1791
- "How does QuickTest capture user processes in Web pages?" on page 1791
- "Can I insert calls to WinRunner tests and functions from QuickTest?" on page 1791

How can I record on objects or environments not supported by QuickTest?

You can do this in a number of ways:

- Install and load any of the add-ins that are available for QuickTest Professional. QuickTest supports many developmental environments including Java, Oracle, .NET, SAP Solutions, Siebel, PeopleSoft, terminal emulators, and Web services.
- You can map objects of an unidentified or custom class to standard Windows classes. For more information on object mapping, see "Test Object Mapping for Unidentified or Custom Classes" on page 299.
- QuickTest provides add-in extensibility that you can use to extend QuickTest built-in support for various objects. This enables you to direct QuickTest to recognize an object as belonging to a specific test object class, and to specify the behavior of the test object. You can also extend the list of available test object classes that QuickTest recognizes. This enables you to create tests that fully support the specific behavior of your custom objects.
- You can define **virtual objects** for objects that behave like test objects, and then record in the normal recording mode. For more information on defining virtual objects, see "Virtual Objects" on page 1513.
- You can record your clicks and keyboard input based on coordinates in the **low-level recording** or **analog** modes. For more information on low-level and analog recording, see "Recording Modes" on page 467.

How can I launch an application from a test?

An application can be launched from within a test by adding a **SystemUtil** step to your test, such as:

```
SystemUtil.Run "D:\My Music\Breathe.mp3","","","D:\My Music\Details","open"
```

For Windows-based applications, you should also ensure that in the Windows Applications tab of the Record and Run Settings dialog box, you configure QuickTest to record and run on applications opened by QuickTest.

How does QuickTest capture user processes in Web pages?

QuickTest hooks the Microsoft Internet Explorer browser. As the user navigates the Web-based application, QuickTest records the user operations. (For information on modifying which user operations are recorded, see the section on configuring Web event recording in the *HP QuickTest Professional Add-ins Guide*.) QuickTest can then run the test by running the steps as they originally occurred.

Can I insert calls to WinRunner tests and functions from QuickTest?

The **Insert > Call to WinRunner > Test** and **Insert > Call to WinRunner > Function** commands are no longer available. You can continue to run existing QuickTest tests that contain calls to WinRunner tests and functions, and you can view run results in the Run Results Viewer.

Note: Hewlett-Packard (HP) has discontinued HP WinRunner (WR) 7.5, 7.6, 8.0, 8.2, 9.2 (all editions) and has announced an End-of-Support timeline for these products. For details, see: http://support.openview.hp.com/encore/wr.jsp?jumpid=reg_R1002_USEN

Programming in the Expert View

This section includes answers to the following questions:

- "Can I store functions and subroutines in a function library?" on page 1792
- "How can I enter information during a run session?" on page 1793
- "I have a Microsoft Access database that contains data I would like to use in my test. How do I do this?" on page 1793
- "How do I customize the Run Results?" on page 1794

Can I store functions and subroutines in a function library?

You can define functions within an individual action, or you can create one or more VBScript function libraries containing your functions, and then call them from any test. You can use the QuickTest function library editor to create and debug your function libraries.

You can also register your functions as methods for QuickTest test objects. Your registered methods can override the functionality of an existing test object method for the duration of a run session, or you can register a new method for a test object class.

For more information, see Chapter 29, "User-Defined Functions and Function Libraries".

You can help improve QuickTest performance by storing your functions in function libraries instead of as reusable actions.

How can I enter information during a run session?

The VBScript InputBox function enables you to display a dialog box that prompts the user for input and then continues running the test. You can use the value that was entered by the user later in the run session. For more information on the InputBox function, see the *VBScript Reference*.

The following example shows the InputBox function used to prompt the user for a password:

```
Browser("Mercury Tours").Page("Mercury Tours").WebEdit("username").Set  
"administrator"  
Passwd = InputBox ("Enter password", "User Input")  
Browser("Mercury Tours").Page("Mercury Tours").WebEdit("password").Set Passwd
```

I have a Microsoft Access database that contains data I would like to use in my test. How do I do this?

The Expert View enables you to access databases using ADO and ODBC. Below is a sample test that searches for books written by an author in the "Authors" table of the database.

```
Dim MyDB  
Dim MyEng  
Set MyEng = CreateObject("DAO.DBEngine.35")  
Dim Td  
Dim rs  
  
' Specify the database to use.  
Set MyDB = MyEng.OpenDatabase("BIBLIO.MDB")  
  
' Read and use the name of the first 10 authors.  
Set Td = MyDB.TableDefs("Authors")  
Set rs = Td.OpenRecordset  
rs.MoveFirst  
For i = 1 To 10  
    Browser("Book Club").Page("Search Books").WebEdit("Author Name").Set  
    rs("Author")  
    Browser("Book Club").Page("Search Books").WebButton("Search").Click  
    Next
```

How do I customize the Run Results?

You can add information to the run results report by using the **ReportEvent** method, for example:

`Reporter.ReportEvent 1, "Custom Step", "The user-defined step failed"`

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

The results of each QuickTest run session are saved in a single **.xml** file (called **results.xml**). You can modify this file, as needed. You can use the **HP Run Results Schema** (available from the QuickTest Professional Help) to help you customize your run results.

Working with Dynamic Content

This section includes answers to the following questions:

- "How can I create and run tests on objects that change dynamically from viewing to viewing?" on page 1794
- "How can I check that a child window exists (or does not exist)?" on page 1795
- "How does QuickTest record on dynamically generated URLs and Web pages?" on page 1796
- "How does QuickTest handle tabs in browsers?" on page 1796

How can I create and run tests on objects that change dynamically from viewing to viewing?

Sometimes the content of objects in an application changes due to dynamic content. You can create dynamic descriptions of these objects so that QuickTest will recognize them when it runs the test using regular expressions, the **Description** object, repository parameters, or **SetTOProperty** steps.

How can I check that a child window exists (or does not exist)?

Sometimes a link in one window creates another window.

You can use the **Exist** property to check whether or not a window exists. For example:

```
If Window("Main").ActiveX("Slider").Exist Then  
    ...
```

You can also use the **ChildObjects** method to retrieve all child objects (or the subset of child objects that match a certain description) on the Desktop or within any other parent object.

Example:

```
Set oDesc = Description.Create  
oDesc("Class Name").Value = "Window"  
  
Set coll = Desktop.ChildObjects(oDesc)  
For i = 0 to coll.count -1  
    msgbox coll(i).GetROProperty("text")  
Next
```

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

How does QuickTest record on dynamically generated URLs and Web pages?

QuickTest actually clicks links as they are displayed on the page. Therefore, QuickTest records how to find a particular object, such as a link on the page, rather than the object itself. For example, if the link to a dynamically generated URL is an image, then QuickTest records the "IMG" HTML tag, and the name of the image. This enables QuickTest to find this image in the future and click on it.

How does QuickTest handle tabs in browsers?

QuickTest provides several methods that you can use with the **Browser** test object to manage tabs in your Web browser.

OpenNewTab opens a new tab in the current Web browser.

IsSiblingTab indicates whether a specified tab is a sibling of the current tab object in the same browser window.

Close closes the current tab if more than one tab exists, and closes the browser window if the browser contains only one tab.

CloseAllTabs closes all tabs in a browser and closes the browser window.

For more information on these **Browser**-related methods, see the **Web** section of the *HP QuickTest Professional Object Model Reference*.

Advanced Web Issues

This section includes answers to the following questions:

- "How does QuickTest handle cookies?" on page 1797
- "Where can I find a Web page's cookie?" on page 1797
- "How does QuickTest handle session IDs?" on page 1797
- "How does QuickTest handle server redirections?" on page 1797
- "How does QuickTest handle meta tags?" on page 1798
- "Does QuickTest work with .asp and .jsp?" on page 1798

- "How does QTP support AJAX?" on page 1798
- "Does QuickTest work with COM?" on page 1798
- "Does QuickTest work with XML?" on page 1798
- "How can I access HTML tags directly?" on page 1799
- "Where can I find information on the Internet Explorer Document Object Model?" on page 1799
- "How can I send keyboard key commands (such as shortcut commands) to objects that do not support the Type method?" on page 1800

How does QuickTest handle cookies?

Server side connections, such as CGI scripts, can use cookies both to store and retrieve information on the client side of the connection.

QuickTest stores cookies in the memory for each user, and the browser handles them as it normally would.

Where can I find a Web page's cookie?

The cookie used by the Internet Explorer browser can be accessed through the browser's Document Object Model (DOM) using the **.Object** property. In the following example the cookie collection is returned from the browser:

```
Browser("Flight reservations").Page("Flight reservations").Object.Cookie
```

How does QuickTest handle session IDs?

The server, not the browser, handles session IDs, usually by a cookie or by embedding the session ID in all links. This does not affect QuickTest.

How does QuickTest handle server redirections?

When the server redirects the client, the client generally does not notice the redirection, and misdirections generally do not occur. In most cases, the client is redirected to another script on the server. This additional script produces the HTML code for the subsequent page to be viewed. This has no effect on QuickTest or the browser.

How does QuickTest handle meta tags?

Meta tags do not affect how the page is displayed. Generally, they contain information only about who created the page, how often it is updated, what the page is about, and which keywords represent the page's content. Therefore, QuickTest has no problem handling meta tags.

Does QuickTest work with .asp and .jsp?

Dynamically created Web pages utilizing Active Server Page technology have an **.asp** extension. Dynamically created Web pages utilizing Java Server Page technology have a **.jsp** extension. These technologies are completely server-side and have no bearing on QuickTest.

How does QTP support AJAX?

You can use QuickTest Professional Web Add-in Extensibility to add your own support for custom Web controls. The Web Add-in Extensibility SDK installs a sample toolkit support set that provides partial support for some ASP .NET AJAX controls. You can use this sample to learn how to create your own support for your AJAX controls. For more information, see the *HP QuickTest Professional Web Add-in Extensibility Developer Guide*.

Does QuickTest work with COM?

QuickTest complies with the COM standard.

QuickTest supports COM objects embedded in Web pages (which are currently accessible only using Microsoft Internet Explorer), and you can drive COM objects in VBScript.

Does QuickTest work with XML?

XML is eXtensible Markup Language, a pared-down version of SGML for Web documents, that enables Web designers to create their own customized tags. QuickTest supports XML and recognizes XML tags as objects.

You can also create XML checkpoints to check the content of XML documents in Web pages, frames or files. QuickTest also supports XML output and schema validation.

For more information, see Chapter 21, "XML Checkpoints," and the **XMLUtil** object in the **Utility** section of the *HP QuickTest Professional Object Model Reference*.

How can I access HTML tags directly?

QuickTest provides direct access to the Internet Explorer's Document Object Model (DOM) through which you can access the HTML tags directly. Access to the DOM is performed using the .Object notation.

The test below demonstrates how to iterate over all the tags in an Internet Explorer page. The test then outputs the inner-text of the tags (the text contained between the tags) to the Run Results using the Reporter object.

```
' Use the on error option because not all the elements have inner-text.  
On Error Resume Next  
Set Doc = Browser("CNN Interactive").Page("CNN Interactive").Object  
  
' Loop through all the objects in the page.  
For Each Element In Doc.all  
    TagName = Element.tagName ' Get the tag name.  
    InnerText = Element.innerText ' Get the inner text.  
  
' Write the information to the run results.  
    Reporter.ReportEvent 0, TagName, InnerText  
Next
```

Where can I find information on the Internet Explorer Document Object Model?

For information on the Internet Explorer DOM, browse to the following Web sites:

Document object:

<http://msdn.microsoft.com/en-us/library/ms531073.aspx>

Other DHTML objects:

<http://msdn.microsoft.com/en-us/library/ms533054.aspx>

General DHTML reference:

<http://msdn.microsoft.com/en-us/library/ms533050.aspx>

How can I send keyboard key commands (such as shortcut commands) to objects that do not support the Type method?

For objects that do not support the **Type** method, use the Windows Scripting **SendKeys** method. For more information, see the Microsoft VBScript Language Reference (choose **Help > QuickTest Professional Help > VBScript Reference > Windows Script Host**).

Standard Windows Environment

This section includes the answers to the following questions:

- "How can I record on nonstandard menus?" on page 1800
- "How can I terminate an application that is not responding?" on page 1800
- "Can I copy and paste to and from the Clipboard during a run session?" on page 1801

How can I record on nonstandard menus?

You can modify how QuickTest behaves when it records menus. The options that control this behavior are located in the Windows Applications > Advanced Options pane.

(**Tools > Options > Windows Applications node> Advanced node**).

For more information, see the *HP QuickTest Professional Add-ins Guide*.

How can I terminate an application that is not responding?

You can terminate any standard application while running a test in QuickTest by adding one of the following steps to the test:

- `SystemUtil.CloseProcessByName "app.exe"`
- `SystemUtil.CloseProcessByWndTitle "Some Title"`

Can I copy and paste to and from the Clipboard during a run session?

You can use the Clipboard object to copy, cut, and paste text during a QuickTest run session.

The Clipboard object supports the same methods as the Clipboard object available in Visual Basic, such as:

- Clear
- GetData
- GetText
- SetData
- SetText

For more information on these methods, see <http://msdn.microsoft.com/en-us/library/ms172962.aspx>.

Below is an example of Clipboard object usage:

```
Set MyClipboard = CreateObject("Mercury.Clipboard")
MyClipboard.Clear
MyClipboard.SetText "TEST"
MsgBox MyClipboard.GetText
```

Test Maintenance

This section includes answers to the following questions:

- "How do I maintain my test when my application changes?" on page 1802
- "Can I increase or decrease Active Screen information after I finish recording a test?" on page 1803
- "How can I remove run results files from old tests?" on page 1804

How do I maintain my test when my application changes?

The way to maintain a test when your application changes depends on how much your application changes. This is one of the main reasons you should create a small group of tests rather than one large test for your entire application.

You can also use QuickTest actions to design more modular and efficient tests. Divide your test into several actions, based on functionality. When your application changes, you can modify a specific action, without changing the rest of the test. Whenever possible, insert calls to reusable actions rather than creating identical pieces of script in several tests. This way, changes to your original reusable action are automatically applied to all tests calling that action. For more information, see Chapter 14, "Actions."

If you have many tests and actions that contain the same test objects, it is recommended to work with shared object repositories so that you can update object information in a centralized location.

You can use the **Update Run Mode** option to update changed information for checkpoints or the Active Screen, or to change the set of identification properties used to identify the objects in your application. For more information, see "Update Options Tab (Update Run Dialog Box)" on page 1271.

If there is a discrepancy between the identification property values saved in the object repository and the object property values in the application, you can use the **Maintenance Run Mode** to help correct this. When you run a test in Maintenance Run Mode, QuickTest runs your test, and then guides you through the process of updating your steps and object repository each time it encounters a step it cannot perform due to an object repository discrepancy. For more information, see "Maintenance Run Mode" on page 1242.

Can I increase or decrease Active Screen information after I finish recording a test?

If you find that the information saved in the Active Screen after recording is not sufficient for your test editing needs, or if you no longer need Active Screen information, and you want to decrease the size of your test, there are several methods of changing the amount of Active Screen information saved with your test.

- To decrease the disk space used by your test, you can delete Active Screen information by selecting **Save As**, and clearing the **Save Active Screen files** check box. For more information, see "Save Test Dialog Box" on page 412.
- If you chose not to save all information in the Active Screen when testing a Windows application, you can use one of several methods to increase the information stored in the Active Screen.

Confirm that the Active Screen capture preference in the Active Screen pane of the Options dialog box is set to capture the amount of information you need and then:

- Perform an **Update Run Mode** operation to save the required amount of information in the Active Screen for all existing steps. For more information on the **Update Run Mode** options, see "Update Options Tab (Update Run Dialog Box)" on page 1271.
- Re-record the steps containing the objects you want to add to the Active Screen.

To re-record the step, select the step after which you want to record your step, position your application to match the selected location in your test, and then begin recording. Alternatively, place a breakpoint in your test at the step before which you want to add a step and run your test to the breakpoint. This brings your application to the point from which to record the step. For more information on setting breakpoints, see "Breakpoints" on page 1213.

For more information on changing the amount of information saved in the Active Screen for Windows applications, see "Active Screen Pane (Options Dialog Box)" on page 1434.

How can I remove run results files from old tests?

You can use the Run Results Deletion Tool to view a list of all of the run results in a specific location in your file system or in your Quality Center project. You can then delete any run results that you no longer require.

The Run Results Deletion Tool enables you to sort the run results by name, date, size, and so forth, so that you can more easily identify the results you want to delete.

To open this utility, choose **Start > Programs > HP QuickTest Professional > Tools > Run Results Deletion Tool**.

Testing Localized Applications

I am testing localized versions of a single application, each with localized user interface strings. How do I create efficient tests in QuickTest?

You can parameterize these user interface strings using parameters from the global Environment variable list. This is a list of variables and corresponding values that can be accessed from any test. For more information, see Chapter 22, "Parameterizing Values."

I am testing localized versions of a single application. How can I efficiently input different data in my tests, depending on the language of the application?

If you are running a single iteration of your test, or if you want values to remain constant for all iterations of an action or test, use environment variables, and then change the active environment variable file for each test run.

If you are running multiple iterations of your test or action, and you want the input data to change in each iteration, you can create an external data table for each localized version of your application. When you change the localized version of the application you are testing, you simply switch the data table file for your test in the Resources pane of the Test Settings dialog box. For more information on working with data tables, see Chapter 38, "Data Table Pane." For more information on selecting the data table file for your test, see "Resources Pane (Test Settings Dialog Box)" on page 1475.

Improving QuickTest Performance

This section includes the answers to the following questions:

- "How can I improve the working speed of QuickTest?" on page 1806
- "How can I decrease the disk space used by QuickTest?" on page 1809
- "Is there a recommended length for tests?" on page 1810

How can I improve the working speed of QuickTest?

You can improve the working speed of QuickTest by doing any of the following:

- In the Add-in Manager, load only the add-ins you need for a specific QuickTest session when QuickTest starts. This will improve performance while learning objects and during run sessions. For more information on loading add-ins, see the *HP QuickTest Professional Add-ins Guide*.
- Minimize the number of actions in a test. Ideally, a test should not contain more than a few dozen actions.
- Store your functions in function libraries instead of as reusable actions.
- Run your tests in "fast mode." From the Run pane in the Options dialog box, select the **Fast** option. This instructs QuickTest to run your test without displaying the execution arrow for each step, enabling the test to run faster. For more information on the Run pane of the Options dialog box, see "Run Pane (Options Dialog Box)" on page 1447.
- If you are not using the Active Screen while editing your test, hide the Active Screen while editing your test to improve editing response time. Choose **View > Active Screen**, or toggle the Active Screen toolbar button to hide the Active Screen. For more information, see Chapter 2, "QuickTest at a Glance."

► Decide if and how much information you want to capture and save in the Active Screen. The more information you capture, the easier it is to add steps to your test using the many Active Screen options, but more captured information also leads to slower recording and editing times. You can choose from the following Active Screen options to improve performance:

- If you are testing Windows applications, you can choose to save all Active Screen information in every step, save information only in certain steps, or to disable Active Screen captures entirely. You set this preference in the Active Screen pane of the Options dialog box. For more information, see "Active Screen Pane (Options Dialog Box)" on page 1434.
- If you are testing Web applications, you can disable screen capture of all steps in the Active Screen. From the Active Screen pane of the Options dialog box, click **Custom Level** to open the Custom Active Screen Capture Settings dialog box.

Select the **Disable Active Screen Capture** option. This will improve recording time. For more information on the Active Screen pane of the Options dialog box, see "Active Screen Pane (Options Dialog Box)" on page 1434.

- When you save a new test, or when you save a test with a new name using **Save As**, you can choose not to save the captured Active Screen files with the test by clearing the **Save Active Screen files** option in the Save or Save As dialog box. This is especially useful when you have finished designing your test and you plan to use your test only for test runs. Tests without Active Screen files open more quickly and use significantly less disk space.

For more information on the Active Screen pane of the Options dialog box, see "Active Screen Pane (Options Dialog Box)" on page 1434.

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 "Update Options Tab (Update Run Dialog Box)" on page 1271.

- Decide if and when you want to capture and save images and/or movies of the application for the run results. You can reduce disk space and improve test run time by saving screen captures and movie segments only in certain situations, such as when errors occur, or by not saving them at all. To do this, use the **Save still image captures to results** and **Save movie to results** options in the Run > Screen Capture pane in the Options dialog box. For more information, see "Screen Capture Pane (Options Dialog Box)" on page 1450.
- Save the run results report to a temporary folder to overwrite the results from the previous run session every time you run a test. For more information, see "Run Dialog Box: Results Location Tab (For Tests Stored in the File System)" on page 1073 or "Run Dialog Box: Results Location Tab (For Tests Stored in Quality Center)" on page 1075.
- Use the Results Deletion Tool to remove unwanted or obsolete run results from your system, according to specific criteria that you define. This enables you to free up valuable disk space. For more information, see "Run Results Deletion Tool" on page 1160.

How can I decrease the disk space used by QuickTest?

You can decrease the disk space used by QuickTest by doing any of the following:

- Decide if and when you want to capture and save images and/or movies of the application for the run results. You can reduce disk space and improve test run time by saving screen captures and movie segments only in certain situations, such as when errors occur, or by not saving them at all. To do this, use the **Save still image captures to results** and **Save movie to results** options in the Run > Screen Capture pane in the Options dialog box. For more information, see "Screen Capture Pane (Options Dialog Box)" on page 1450.
- Decide if and how much information you want to capture and save in the Active Screen. The more information you capture, the easier it is to add steps to your test using the many Active Screen options, but more captured information also leads to slower recording and editing times. You can choose from the following Active Screen options to improve performance:
 - If you are testing Windows applications, you can choose to Save all Active Screen information in every step, save information only in certain steps, or to disable Active Screen captures entirely. You set this preference in the Active Screen pane of the Options dialog box. For more information, see "Active Screen Pane (Options Dialog Box)" on page 1434.
 - If you are testing Web applications, you can disable screen capture of all steps in the Active Screen. From the Active Screen pane, click **Custom Level** to open the Custom Active Screen Capture Settings dialog box. Select the **Disable Active Screen Capture** option. This will improve recording time. For more information on the Active Screen pane of the Options dialog box, see "Active Screen Pane (Options Dialog Box)" on page 1434.
 - When you save a new test, or when you save a test with a new name using Save As, you can choose not to save the captured Active Screen files with the test by clearing the **Save Active Screen files** option in the Save or Save As dialog box. This is especially useful when you have finished designing your test and you plan to use your test only for test runs. Tests without Active Screen files use significantly less disk space.

Tip: If you need to recover Active Screen files after you save a test without Active Screen files, re-record the necessary steps or use the **Update Run Mode** option to recapture screens for all steps in your test. For more information, see "Update Options Tab (Update Run Dialog Box)" on page 1271.

Is there a recommended length for tests?

Although there is no formal limit regarding test length, it is recommended that you divide your tests into actions and that you use reusable actions in tests, whenever possible. An action should contain no more than a few hundreds steps and, ideally, no more than a few dozen. For more information, see Chapter 14, "Actions."

D

Custom Process Guidance Packages

This chapter includes:

Concepts

- Custom Process Guidance Packages - Overview on page 1812
- Data Files for Process Guidance Packages on page 1813

Tasks

- How to Create a Custom Package Configuration File on page 1814
- How to Install Custom Process Guidance Packages in QuickTest on page 1815

Reference

- XML Details for Custom Process Guidance Packages on page 1816

Concepts

Custom Process Guidance Packages - Overview

A **process** is a collection of activities, or sub-processes that are performed within an organization. Each process walks a user step-by-step through the activities that are required for that process. As users navigate through the activities for each process and perform the tasks described in each activity, they become acquainted with the way in which a particular process should be performed.

You can create your own custom process guidance packages and distribute them to the QuickTest users in your organization. QuickTest users can then display the processes from your package in QuickTest while they work, to assist them in following your organization's processes and standards according to your preferred methodology.

A process guidance package is comprised of two entities: the package configuration file and the data files.

- **Package Configuration file.** This XML file defines the **Processes** included in the package and the structure of the **Groups** and **Activities** in each process.
- **Data Files.** A set of HTML files. Each HTML data file contains the content for a single activity.

For an overview of process guidance and how it is used in QuickTest, see Chapter 41, "Process Guidance Panes."

Data Files for Process Guidance Packages

Each data file contains the HTML content for a single process guidance activity. When an activity link is clicked in the **Process Guidance Activities** pane, the HTML content is displayed in a browser control in the **QuickTest Process Guidance Description** pane.

The package data files can include reference to a **.css** file to display content in your organization's standard style, and can contain any content that can be displayed by a browser.

You can also add special code to your HTML data files to activate QuickTest dialog boxes or jump to other process guidance processes or activities using the QuickTest **UI** automation object. For details, see the Automation Object Model Reference (**Help > QuickTest Professional Help > HP QuickTest Professional Advanced References > HP QuickTest Professional Automation Object Model Reference**).

The HTML data files, and any folders or files that the HTML data files reference can be stored on the user's local hard drive in a network location on the file system or on a Web server. The package configuration file (specifically, the **Address** attribute of each **Activity** element in this file) provides HTML links for each activity.

Write the HTML data file for each activity such that there will be minimum scrolling when the content is displayed in the Process Guidance Description pane at its default size.

If you find that your HTML data files are too long, you may want to break them up into multiple process guidance activities to make it easier for your QuickTest users to reference while they work.

Tasks

How to Create a Custom Package Configuration File

The following steps describe how to create a custom package configuration file. To view an example of a custom package configuration file, see "Example of Custom Package Configuration File" on page 1818.

1 Create an XML file

This XML file describes the processes included in the package and sets the structure of the groups and activities in each process. The structure you define is displayed as the table of contents for a selected process in the QuickTest **Process Guidance Activities** pane.

For a list of the elements and attributes you can use in your package configuration file, see "XML Details for Custom Process Guidance Packages" on page 1816.

2 Save the XML configuration file

Save the file with the name: **Configuration.xml**

3 Create the HTML data files

For details, see "Data Files for Process Guidance Packages" on page 1813.

4 Install the custom package configuration file together with the relevant HTML data files (and any files or folders referenced from the HTML data files)

For details, see "How to Install Custom Process Guidance Packages in QuickTest" on page 1815.

How to Install Custom Process Guidance Packages in QuickTest

The following steps describe how to distribute and install custom process guidance packages. You can:

- Install the process guidance package from a zip file
- Install the process guidance package via registry key

Install the process guidance package from a zip file

- 1** Create a folder that contains the **Configuration.xml** file and all the HTML data files (as well as any files or folders referenced from the HTML data files).
- 2** Zip the folder and then send the **.zip** file to all relevant QuickTest users or store it in a location that they can access.
- 3** In QuickTest, select **File > Process Guidance Management**. The Process Guidance Management dialog box opens.
- 4** Click the **Add** button and browse to the **.zip** file. The package is added and its processes are displayed in the dialog box.

Install the process guidance package via registry key

- 1** Prepare the **Configuration.xml** file and the data files.
- 2** Place the data files in a local or shared network folder or on a Web server. Ensure that the **Address** attribute of the **Activity** elements in the **Configuration.xml** file point to this location.
- 3** Copy the **Configuration.xml** to a local drive on the QuickTest computer.
- 4** Open the Registry Editor and find the key:
HKEY_LOCAL_MACHINE\SOFTWARE\Mercury Interactive\QuickTest Professional\MicTest\ProcessGuidance\ConfFiles
- 5** Add a value to this key with the path to the **Configuration.xml** file. The next time QuickTest is opened, it will include the new package.

Reference



XML Details for Custom Process Guidance Packages

You can use the following elements and attributes in your package configuration file.

<Process> Elements	
Name	The name of the process as you want it to appear in the QuickTest Process Guidance pane.
ID	A unique identification name. This name is used to distinguish between two processes with the same name.
DocType	<p>Indicates the QuickTest document types for which this process is applicable. If specified, the process is available only when the relevant document type is open.</p> <p>In the example shown in "Example of Custom Package Configuration File" on page 1818, if a QuickTest user opens a test document, both processes will be available, but if an application area document is opened, only the second process will be available.</p> <p>Possible values:</p> <ul style="list-style-type: none"> ➤ test. A test document. ➤ AA. An application area document. ➤ BC. A business component document. ➤ SBC. A scripted component document.
Addin	<p>Indicates the QuickTest add-ins for which this process is applicable. If specified, the process is available only when the relevant add-in is loaded. In the example above, the first process will be available only if the Web Add-in is loaded. The second process will always be visible.</p> <p>Specify the add-in value using the add-in name as displayed in the Add-in Manager.</p>
SortLevel	Determines the location of the process within the process list. This list is displayed in the Process Guidance Management dialog box and in the QuickTest Automation > Process Guidance List menu.

<Group> Elements	
Name	Same as the Name attribute for the <Process> element, as described above.
ID	Same as the ID attribute for the <Process> element, as described above.
Addin	Same as the Addin attribute for the <Process> element, as described above.
<Activity> Elements	
Name	Same as the Name attribute for the <Process> element, as described above.
ID	Same as the ID attribute for the <Process> element, as described above.
Addin	Same as the Addin attribute for the <Process> element, as described above.
Address	The path where the relevant HTML data file is located. This can be a local or network path on the file system or an HTTP address. If you specify a relative path, the location is resolved relative to the configuration file location.

For more information, see "Example of Custom Package Configuration File" on page 1818 for a usage example.



Example of Custom Package Configuration File

The following is an example of a package configuration file that contains two processes:

```
<?xml version="1.0" encoding="UTF-8"?>
<ProcessGuidance Name="MyCustomPackage">
    <Process Name="My Process" ID="Process1" DocType="test" Addin="web"
SortLevel="4" >
        <Group Name="New User Overview">
            <Activity Name="Step 1" Address="Step1.html" />
            <Activity Name="Step 2" Address="Step2.html" />
        </Group>
    </Process>
    <Process Name="Important Processes" ID="Process2" DocType="test|AA"
SortLevel="3">
        <Group Name="Getting Started">
            <Activity Name="Open" Address="F:\ProcessData\open.html" />
            <Activity Name="Create" Address="F:\ProcessData\create.html" />
            <Activity Name="Test" Address="F:\ProcessData\test.html" />
            <Activity Name="Debug" Address="F:\ProcessData\debug.html" />
        </Group>
        <Group Name="Finish">
            <Activity Name="Save" Address="F:\ProcessData\save.html" />
            <Activity Name="Close" Address="F:\ProcessData\close.html" />
            <Activity Name="Exit" Address="F:\ProcessData\exit.html" />
        </Group>
    </Process>
</ProcessGuidance>
```

E

Bitmap Checkpoint Customization

This appendix is intended for COM programmers who want to customize the algorithm used to compare bitmaps in bitmap checkpoints.

This chapter includes:

Concepts

- About Bitmap Checkpoint Customization on page 1820
- Custom Comparer for Images Whose Location Changes in the Application - Use-Case Scenario on page 1821
- Custom Bitmap Comparer Development on page 1823

Tasks

- How to Develop a Custom Comparer on page 1824
- How to Implement the Bitmap Comparer Interfaces on page 1828
- How to Install Your Custom Comparer and Register it to QuickTest on page 1832
- How to Use the Bitmap Checkpoint Customization Samples on page 1836
- How to Develop a Custom Comparer - Tutorial on page 1839

Reference

- The Bitmap Checkpoint Comparer Interfaces on page 1851

Concepts

About Bitmap Checkpoint Customization

By default, a bitmap checkpoint compares the actual and expected bitmaps pixel by pixel and fails if there are any differences. QuickTest enables its users to define tolerance levels for bitmap checkpoints to refine the bitmap comparison and make it more flexible. For more information, see "Fine-Tuning the Bitmap Comparison" on page 621.

If you need to further customize the way bitmaps are compared in checkpoints, you can develop custom comparers that compare bitmaps according to your requirements. You develop a custom comparer as a COM object and install and register it on the QuickTest computer. A QuickTest user can then choose to use a custom comparer to perform the comparison in a bitmap checkpoint (on a per checkpoint basis).

You implement bitmap checkpoint customization by developing custom comparers. A custom comparer is a COM object that you develop to run the bitmap comparison in a bitmap checkpoint according to a specific algorithm. The COM object that you develop must implement interfaces that QuickTest provides in a type library, and register to the component category that QuickTest defines for bitmap comparers. The type library (**BitmapComparer.tlb**) and the category ID (defined in **ComponentCategory.h**) are available in **<QuickTest installation folder>\dat\BitmapCPCustomization**.

When a QuickTest user creates or edits a bitmap checkpoint, QuickTest displays any registered custom comparers in the Bitmap Checkpoint Properties dialog box (in addition to the QuickTest default comparer). The user can then select a comparer according to the testing requirements of the specific application or bitmap being tested. For more information about using custom comparers in QuickTest, see "Custom Comparer Area" on page 632.

You can find an example of a situation where bitmap checkpoint customization enhanced the use of bitmap checkpoints, in "Custom Comparer for Images Whose Location Changes in the Application - Use-Case Scenario" on page 1821.

Custom Comparer for Images Whose Location Changes in the Application - Use-Case Scenario

Ben is a quality assurance engineer who is experienced in using QuickTest, and often uses bitmap checkpoints to test the appearance of different icons or pictures in the user interface he is testing. He does not have a programming background.

Joanne is a software engineer who is experienced in image processing and is familiar with COM programming.

When Ben began testing the user interface of a furniture purchasing application, he created a bitmap checkpoint to test that the pictures of the items on sale were displayed properly. In the checkpoint, he captured an image of the pane in the application that contained the pictures he wanted to test. Ben found that the bitmap checkpoint often failed, even though the graphic images displayed in the application during the run seemed identical to the ones he had captured when creating the checkpoint.

Ben reviewed the actual, expected, and difference bitmaps displayed in the run results. He also took a closer look at the application's user interface. The application contained three panes. The left pane displayed general information, the middle pane displayed the pictures of the items on sale, and the right pane displayed the corresponding list of items and relevant details. Ben found that depending on the information displayed in the left pane, the images in the middle pane sometimes shifted slightly one way or the other within the pane. While the images themselves were still identical, their changed location was causing the bitmap checkpoint to fail.

Ben did not want to use pixel tolerance to address this issue because he wanted the checkpoint to fail when the pixels within the images themselves were not identical.

When Ben mentioned his problem to a co-worker, she suggested that bitmap checkpoint customization could solve the problem, and referred him to Joanne. Joanne developed a custom comparer that would accept as input the number of pixels that the images should be allowed to shift without failing the checkpoint. The bitmap comparison that Joanne designed would pass the checkpoint only if the images were identical and they had all shifted by the same number of pixels. This way, Ben knew that his checkpoint would still catch incorrect images and cases where the application's interface looked bad because the images were no longer aligned.

Ben installed and registered the custom comparer on his QuickTest computer, and then selected the new custom comparer for his bitmap checkpoint. After some experimenting, he found the optimal number of pixels to enter in the configuration string, so that significant changes in the application's interface were detected, but insignificant shifting of the images did not cause the checkpoint to fail.

After Ben successfully used this custom comparer for a while, his company decided to install and register it on all of the QuickTest computers. The custom comparer would now be available to everyone in the quality assurance team to use for similar situations.

Custom Bitmap Comparer Development

To develop a custom comparer, you create a COM object that implements the QuickTest bitmap checkpoint comparer interfaces (described on page 1851) to perform the following:

- Accept input from QuickTest and perform the bitmap comparison.
- Provide comparison results to QuickTest.
- (Optional) Provide information that QuickTest can display in the Bitmap Checkpoint Properties dialog box when a user creates or edits a bitmap checkpoint.

Custom comparers run within the QuickTest context. You must therefore exercise care when developing your custom comparer, as its behavior and performance will affect the behavior and performance of QuickTest.

For QuickTest to recognize the custom comparer, it must be registered to the component category that QuickTest defines for bitmap comparers.

Depending on how you implement your custom comparer, you can design the comparer to register itself when it is installed, or you can provide an additional program that needs to be run at the time of installation. For details, see "How to Install Your Custom Comparer and Register it to QuickTest" on page 1832.

Perform the tutorial in "How to Develop a Custom Comparer - Tutorial" on page 1839 to learn how to create and use a custom comparer. You can then create your own custom comparers in much the same way. For task details, see "How to Develop a Custom Comparer" on page 1824.

In addition to the tutorial, QuickTest provides source files that implement a sample custom comparer in different languages. The source files are provided in C++ and in Visual Basic. Both projects generate a similar custom comparer.

You can study the samples to help you learn about QuickTest bitmap checkpoint customization, or use them as a reference or template when you develop your own custom comparers. For details, see "How to Use the Bitmap Checkpoint Customization Samples" on page 1836.

Tasks

How to Develop a Custom Comparer

This task describes the process for developing a custom bitmap comparer.

Tip: To practice performing this task, see "How to Develop a Custom Comparer - Tutorial" on page 1839.

This task includes the following steps:

- "Prerequisites" on page 1836
- "Develop the custom comparer COM object" on page 1825
- "Prepare the custom comparer installation - optional" on page 1825
- "Install the custom comparer" on page 1826
- "Test the custom comparer" on page 1827

1 Prerequisites

- Knowledge of image processing
- Experience in developing COM objects

2 Develop the custom comparer COM object

- a Create the custom comparer COM object. You can use any language and development environment that supports creating COM objects.
-

Note: Depending on the language that you use for development, you might be able to specify the custom comparer name when you create the COM object. Otherwise the name can be specified on the QuickTest computer after registering the object. For details, see "Set the Custom Comparer Name - Optional" on page 1834.

- b Program your COM object to implement the custom comparer interfaces. For details, see "How to Implement the Bitmap Comparer Interfaces" on page 1828.

3 Prepare the custom comparer installation - optional

Your custom comparer might need to be installed on more than one computer. You can create a program that automatically performs the steps necessary to install and register the comparer and its documentation on those computers.

For details on the steps that such a program needs to perform, see "How to Install Your Custom Comparer and Register it to QuickTest" on page 1832.

For example, when you design your custom comparer installation, you must ensure that when it is installed on the QuickTest computer, it is also registered to the component category for QuickTest bitmap comparers. This can be achieved in different ways, such as:

- If you develop your custom comparer in C++ using Microsoft Visual Studio, you can modify the **DllRegisterServer** and **DllUnregisterServer** methods to handle this registration. These methods are called when you run a DLL using the **regsvr32.exe** program. You can see an example of this type of implementation in step 6 of "How to Develop a Custom Comparer - Tutorial", on page 1848.
- If you develop your custom comparer in an environment that does not enable you to modify the registration methods, you can add an additional program that handles this registration and instruct users who install the custom comparer to run this program as well. You can see an example of this type of implementation in the Visual Basic sample custom comparer that QuickTest provides. For more information, see "How to Use the Bitmap Checkpoint Customization Samples" on page 1836.

4 Install the custom comparer

On the computer where you want to use the custom comparer, do one of the following:

- Run the installation program that automatically installs and registers the comparer.
- Manually install and register the custom comparer. For details, see "How to Install Your Custom Comparer and Register it to QuickTest" on page 1832.

5 Test the custom comparer

Create bitmap checkpoint steps in a test in QuickTest. In the Bitmap Checkpoint Properties dialog box, select your custom comparer and use it to perform bitmap checkpoints and check that your customizations perform correctly.

Tip: By default, QuickTest displays expected, actual, and difference bitmaps in the Run Results only for checkpoints that fail. When you test your custom comparer on QuickTest, you might want to see the expected, actual, and difference bitmaps in the run results even for bitmap checkpoints that pass. To configure this, select **Tools > Options > Run > Screen Capture** in QuickTest and set the **Save still image captures to results** option to **Always**. For more information on using this option, see "Screen Capture Pane (Options Dialog Box)" on page 1450.

How to Implement the Bitmap Comparer Interfaces

This task describes how to implement the bitmap comparer interfaces so that your custom comparer COM object performs the following:

- Accepts bitmaps and compares them
- Provides comparison results to QuickTest
- Provides information for the Bitmap Checkpoint Properties dialog box

Note: This task is part of a higher-level task. For details, see "How to Develop a Custom Comparer" on page 1824.

This task includes the following steps:

- "Prerequisite - Reference the type library" on page 1828
- "Implement the CompareBitmaps method to accept input and compare bitmaps" on page 1829
- "Implement the CompareBitmaps method to return the comparison results to QuickTest" on page 1830
- "Implement IBitmapCompareConfiguration to provide information for the Bitmap Checkpoint Properties dialog box" on page 1830

Prerequisite - Reference the type library

In the COM object that you develop, reference the type library that QuickTest provides (located in <QuickTest installation folder>\dat\BitmapCPCustomization\BitmapComparer.tlb)

Implement the **CompareBitmaps** method to accept input and compare bitmaps

QuickTest calls the **CompareBitmaps** method in the **IVerifyBitmap** interface (described on page 1851) to pass the expected and actual bitmaps to the custom comparer for comparison.

Method syntax:

```
HRESULT CompareBitmaps ([in] IPictureDisp* pExpected,  
                      [in] IPictureDisp* pActual,  
                      [in] BSTR bstrConfiguration,  
                      [out] BSTR* pbstrLog,  
                      [out] IPictureDisp** ppDiff,  
                      [out, retval] VARIANT_BOOL* pbMatch);
```

Implement the **CompareBitmaps** method to perform the following:

- Accept and compare two bitmaps according to a predefined algorithm that you define based on the testing requirements.
- Accept a text string that can contain configuration information provided by the QuickTest user (in the Bitmap Checkpoint Properties dialog box), and use it in the comparison. For example, the string could contain tolerance specifications, acceptable deviations in size or location of the image, or any other information that you want to affect the comparison.

The string can have any format you choose (XML, comma separated, INI file style, and so on). Make sure that the documentation you provide for the custom comparer describes the format. The configuration input that the QuickTest user enters in the Bitmap Checkpoint Properties dialog box must conform to this format.

Implement the CompareBitmaps method to return the comparison results to QuickTest

QuickTest displays the results of bitmap checkpoints in the Run Results Viewer.

When you implement the **CompareBitmaps** method in the **IVerifyBitmap** interface (described on page 1851) to compare the bitmaps, you must also return the following information:

- Whether the bitmaps match and the checkpoint should pass.
- A text string that QuickTest displays in the run results.

The purpose of this string is to provide information about the comparison to the QuickTest user, but while you develop and test your comparer, you can use this string for debugging purposes as well.

- A bitmap that visually represents the difference between the actual and expected bitmaps.

The purpose of this bitmap is to help the QuickTest user understand why the checkpoint failed. The custom comparer can create this bitmap using any visualization approach you choose. For example, the default QuickTest comparer creates a black-and-white bitmap containing a black pixel for every pixel that is different in the two images.

Implement IBitmapCompareConfiguration to provide information for the Bitmap Checkpoint Properties dialog box

When a QuickTest user selects a custom comparer in the Bitmap Checkpoint Properties dialog box, QuickTest displays a **Configuration options** text box, and, optionally, a link to documentation provided for the custom comparer. For more information, see "Custom Comparer Area" on page 632.

To support these options, you can implement the **IBitmapCompareConfiguration** interface (described on page 1853) to provide the necessary information for the dialog box.

- Implement the **GetDefaultConfigurationString** method to return the default configuration string for your custom comparer.

Method syntax:

```
HRESULT GetDefaultConfigurationString ([out, retval] BSTR* pbstrConfiguration);
```

QuickTest displays this string in the **Configuration options** box in the Bitmap Checkpoint Properties dialog box.

The format of this string must be the same as the format of the configuration string that the comparer expects as input.

- Implement the **GetHelpFilename** method to return a path to the documentation about your custom comparer. A QuickTest user can then access the documentation from the Bitmap Checkpoint Properties dialog box.

Method syntax:

```
HRESULT GetHelpFilename ([out, retval] BSTR* pbstrFilename);
```

The documentation can be in any format that you choose. QuickTest opens the documentation using the program associated with the provided file type on the user's computer. Therefore, you should provide the documentation in a format for which you expect the QuickTest user to have the necessary program.

The documentation should provide the QuickTest user with the following information:

- The type of comparison the custom comparer performs (to enable the user to determine when to use it to run a bitmap checkpoint).
- The required format for the configuration string and the possible values it can contain.
- An explanation of the comparison result information that is displayed in the run results (text string and difference bitmap).

How to Install Your Custom Comparer and Register it to QuickTest

The custom comparer must be installed and registered on any computer that runs a test with a bitmap checkpoint using the custom comparer.

This task describes how to install the custom comparer on a QuickTest computer and register it to QuickTest.

- ▶ When you develop a custom comparer, you can create a program that automatically performs the steps in this task. If you choose not to create an installation program, review these steps and make sure to provide your users with all of the required files and information.
- ▶ When you install a custom comparer that you did not develop, you may need additional information from the developer to perform the steps in this task. Alternatively, you may receive an installation program from the developer that performs this task automatically.

Note: This task is part of a higher-level task. For details, see "How to Develop a Custom Comparer" on page 1824.

This task includes the following steps:

- ▶ "Prerequisites" on page 1833
- ▶ "Install the custom comparer COM object on the QuickTest computer" on page 1833
- ▶ "Register the custom comparer to the component category for QuickTest bitmap comparers" on page 1833
- ▶ "Place the custom comparer documentation in the correct location" on page 1834
- ▶ "Set the Custom Comparer Name - Optional" on page 1834
- ▶ "Results" on page 1835

Prerequisites

- 1 More than one custom comparer can be installed and registered on the same QuickTest computer.

However, before installing and registering a new version of a specific custom comparer, unregister the existing comparer.

- 2 A custom comparer DLL is created using a specific development environment version. Make sure that the computer on which this DLL runs has the corresponding runtime environment installed.

Install the custom comparer COM object on the QuickTest computer

For example, you can do this by double-clicking the DLL or running the DLL using the **regsvr32.exe** program.

Register the custom comparer to the component category for QuickTest bitmap comparers

Register the component category ID for QuickTest bitmap comparers, **CATID_QTPBitmapComparers**, as a registry key under the COM object's **HKEY_CLASSES_ROOT\CLSID\<Object's CLSID>\Implemented Categories** key.

Note: When QuickTest is installed, it adds this component category ID as a registry key under the **HKEY_CLASSES_ROOT\Component Categories** key. The component category ID is defined in **<QuickTest installation folder>\dat\BitmapCPCustomization\ComponentCategory.h**.

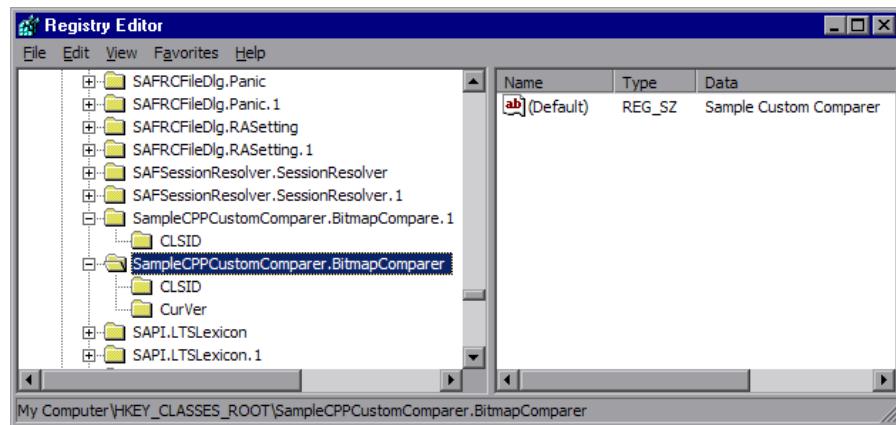
Place the custom comparer documentation in the correct location

The Bitmap Checkpoint Properties dialog box in QuickTest can display documentation about the custom comparer, if such documentation is provided.

Place the custom comparer documentation in the location specified in the custom comparer object's **GetHelpFilename** method.

Set the Custom Comparer Name - Optional

QuickTest displays the name of the custom comparer in the Bitmap Checkpoint Properties dialog box and in the Run Results Viewer. The name that QuickTest uses is the value (in the registry) of the default property of the custom comparer ProgID key under the HKEY_CLASSES_ROOT key. For example, in the image below, the name of the custom comparer is **Sample Custom Comparer**.



- In some environments you set the name while developing the object. For example if the custom comparer is developed in C++ using Microsoft Visual Studio, this name can be specified during development in the **Type** box in the ATL Simple Object Wizard.
- In other environments, you can set or customize the name as part of the installation or registration process on each computer. For example, if the custom comparer is developed in Visual Basic, this registry value is automatically set to the COM object's ProgID. If you want to modify the custom comparer name, you can edit it manually in the registry after the comparer is installed, or design the program that performs the installation and registration to edit this value as well.

Results

In the Bitmap Checkpoint Properties dialog box, QuickTest displays the comparer you installed and registered, along with all of the available custom comparers, and the QuickTest default comparer. You can then select the appropriate comparer to use for each bitmap checkpoint.

How to Use the Bitmap Checkpoint Customization Samples

This task describes how to generate the sample custom comparer, and then register it and work with it.

This task includes the following steps:

- "Prerequisites" on page 1836
- "Generate the sample comparer" on page 1836
- "Install the custom comparer on a QuickTest computer" on page 1836
- "Register the custom comparer to the component category for QuickTest bitmap comparers" on page 1837
- "Study the custom comparer functionality" on page 1838

1 Prerequisites

Decide whether you want to use the sample C++ project or the Visual Basic one.

The samples are located under <QuickTest installation folder>\samples\BitmapCPSample.

2 Generate the sample comparer

- a** To open the sample project, do one of the following:
 - To open the C++ project, use Microsoft Visual Studio 2003 or later.
 - To open the Visual Basic project, use Microsoft Visual Studio 6.0.
- b** Compile the custom comparer, to build the DLL.

3 Install the custom comparer on a QuickTest computer

Run the custom comparer using the **regsvr32.exe** program to install it on the computer.

4 Register the custom comparer to the component category for QuickTest bitmap comparers

► If you are using the C++ sample project:

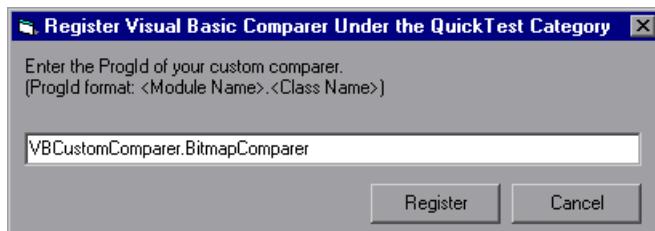
The C++ sample sources implement registering the custom comparer to QuickTest in the **DllRegisterServer** and **DllUnregisterServer** methods. Therefore, if you used the C++ project to create the DLL, running the DLL (in the previous step) will also register the custom comparer.

Note: The name displayed for the custom comparer in QuickTest will be **Custom QTP Bitmap Comparer**.

► If you are using the Visual Basic sample project:

The Visual Basic sample project does not implement this registration. Therefore, the Visual Basic sample also includes an additional tool that you must run after installing the custom comparer, to register the custom comparer to the component category for QuickTest bitmap comparers. For more information, see "How to Install Your Custom Comparer and Register it to QuickTest" on page 1832.

You can run the Visual Basic Comparer Registration Tool from **<QuickTest installation folder>\samples\BitmapCPSample\VBCustomComparer\RegisterCategory.exe** (if you copy and paste this path from the PDF, make sure to remove the line break).



In the dialog box that opens, enter the ProgID of the custom comparer and click **Register**.

Note: The name displayed for the custom comparer in QuickTest will be its ProgId—**VBCustomComparer.BitmapComparer**. For information on modifying this name, see "Set the Custom Comparer Name - Optional" on page 1834.

5 Study the custom comparer functionality

Create bitmap checkpoint steps in a test in QuickTest. In the Bitmap Checkpoint Properties dialog box, you can select the sample custom comparer and use it to perform bitmap checkpoints.

- a** Note the customized information that is displayed in the dialog box when you select the custom comparer.
 - The default configuration string that the sample comparer returns (and QuickTest displays in the Bitmap Checkpoint Properties dialog box) is MaxSurfAreaDiff=140000.
 - The documentation provided with this sample comparer (and opened from the Bitmap Checkpoint Properties dialog box) is the **SampleComparerDetails.txt** text file located in **<QuickTest installation folder>\samples\BitmapCPSample\CPPCustomComparer**.
- b** Run bitmap checkpoints to test the behavior of the sample comparer. For example, you can run checkpoints on the Windows Calculator application, alternately setting the Calculator view to **Standard** or **Scientific**, to obtain different size bitmaps for the same object.

The sample custom comparer does not compare the content of the actual and expected bitmaps. It compares the total number of pixels they contain. For configuration input, this comparer expects a string that defines the MaxSurfAreaDiff parameter. The comparer fails the checkpoint if the difference in total number of pixels is greater than the number defined for MaxSurfAreaDiff.

- View the results of your checkpoint in the run results.

This sample bitmap custom comparer returns the actual bitmap as the difference bitmap. In addition, it provides a text string that specifies the difference in total number of pixels. QuickTest displays this string in the run results.



How to Develop a Custom Comparer - Tutorial

This tutorial walks you step-by-step through the process of creating a custom comparer in C++ using Microsoft Visual Studio. The custom comparer you create is similar to the sample custom comparer provided with QuickTest. You can create your own custom comparers in a similar way. For more information about the sample custom comparer, see "How to Use the Bitmap Checkpoint Customization Samples" on page 1836.

Note: For a task related to this tutorial, see "How to Develop a Custom Comparer" on page 1824.

By following the instructions in this section, you create a COM object that:

- Implements the **CompareBitmaps** method to receive two bitmaps to compare and a configuration string, compare the (size of) the two bitmaps, and return the necessary results.
- Implements the **GetDefaultConfigurationString** method and the **GetHelpFilename** method, to return the information that QuickTest displays in the Bitmap Checkpoint Properties dialog box.
- Registers to the component category for QuickTest bitmap comparers.

When the design of your custom comparer is complete, you can install and register it and use it in QuickTest to run a bitmap checkpoint.

Note: Depending on the version of Microsoft Visual Studio that you use to perform the tutorial, the command names may be different.

This tutorial includes the following steps:

- "Create a new ATL project—SampleCPPCustomComparer" on page 1841
- "Create a new class—CBitmapComparer" on page 1841
- "Define that the CBitmapComparer class implements the bitmap checkpoint comparer interfaces" on page 1842
- "Move the function bodies for the bitmap checkpoint comparer interface methods from BitmapComparer.h to BitmapComparer.cpp" on page 1843
- "Implement the bitmap checkpoint comparer interface methods to customize the bitmap checkpoint as required" on page 1845
- "Design your custom comparer to register to the component category for QuickTest bitmap comparers" on page 1848
- "Compile your DLL and run it using the regsvr32.exe program" on page 1849
- "Test your custom comparer by using it for bitmap checkpoints in QuickTest" on page 1849

1 Create a new ATL project—**SampleCPPCustomComparer**

- a In Microsoft Visual Studio, select **New > Project**. The New Project dialog box opens.
- b Select the **ATL Project** template, enter **SampleCPPCustomComparer** in the **Name** box for the project, and click **OK**. The New ATL Project wizard opens.
- c In **Application Settings**, make sure that the **Attributed** option is not selected, and click **Finish**.

2 Create a new class—**CBitmapComparer**

- a In the class view, select the **SampleCPPCustomComparer** project, right-click, and select **Add > Class**. The Add Class dialog box opens.
- b Select **ATL Simple Object** and click **Add**. The ATL Simple Object Wizard opens.
- c In the **Short name** box, enter **BitmapComparer**. The wizard uses this name to create the names of the class, the interface, and the files that it creates.
- d In the **Type** box, enter **Sample Custom Comparer**. This is the custom comparer name that QuickTest will display in the Bitmap Checkpoint Properties dialog box and in the run results. For more information, see "Set the Custom Comparer Name - Optional" on page 1834.
- e Click **Finish**. The wizard creates the necessary files for the class that you added, including **.cpp** and **.h** files with implementation of **CBitmapComparer** class.

3 Define that the **CBitmapComparer** class implements the bitmap checkpoint comparer interfaces

- a In the class view, select **CBitmapComparer**, right-click, and select **Add > Implement Interface**. The Implement Interface wizard opens.
- b In the **Implement interface from** option, select **File**. Browse to or enter the location of the QuickTest bitmap checkpoint comparer type library. The type library is located in: **<QuickTest installation folder>\dat\BitmapCPCustomization\BitmapComparer.tlb**.

The wizard displays the interfaces available in the selected type library, **IBitmapCompareConfiguration** and **IVerifyBitmap**.

- c Add both interfaces to the list of interfaces to implement, and click **Finish**.

In the **BitmapComparer.h** file, the wizard adds the declarations, classes, and method stubs that are necessary to implement the interfaces. In subsequent steps you will need to add implementation to these method stubs.

Note: In Microsoft Visual Studio 2005, the wizard generates the signature for the **CompareBitmaps** method in the **IVerifyBitmap** interface incorrectly. To enable your project to compile correctly, manually change the type of the last argument (*pbMatch*) from **BOOL*** to **VARIANT_BOOL***.

4 Move the function bodies for the bitmap checkpoint comparer interface methods from **BitmapComparer.h** to **BitmapComparer.cpp**

- a Open the **BitmapComparer.h** and **BitmapComparer.cpp** files.
- b In **BitmapComparer.h**, create declarations for the bitmap checkpoint comparer interface methods (based on the function bodies that the wizard created): **CompareBitmaps**, **GetDefaultConfigurationString**, and **GetHelpFilename**.
- c Move the function bodies that the wizard created for the bitmap checkpoint comparer interface methods from the **BitmapComparer.h** file to the **BitmapComparer.cpp** file.

At the end of this step, **BitmapComparer.cpp** and **BitmapComparer.h** should contain the following code:

```
// BitmapComparer.cpp : Implementation of CBitmapComparer
#include "stdafx.h"
#include "BitmapComparer.h"

// CBitmapComparer
// IBitmapCompareConfiguration Methods
STDMETHODIMP CBitmapComparer::GetDefaultConfigurationString
    (BSTR * pbstrConfiguration)
{
    return E_NOTIMPL;
}
STDMETHODIMP CBitmapComparer::GetHelpFilename(BSTR * pbstrFilename)
{
    return E_NOTIMPL;
}

// IVerifyBitmap Methods
STDMETHODIMP CBitmapComparer::CompareBitmaps
    (IPictureDisp * pExpected, IPictureDisp * pActual,
     BSTR bstrConfiguration, BSTR * pbstrLog,
     IPictureDisp ** ppDiff, VARIANT_BOOL * pbMatch)
{
    return E_NOTIMPL;
}
```

```

// BitmapComparer.h : Declaration of the CBitmapComparer
#pragma once
#include "resource.h"      // main symbols
#include "SampleCPPCustomComparer.h"
// CBitmapComparer
class ATL_NO_VTABLE CBitmapComparer :
    public CComObjectRootEx<CComSingleThreadModel>,
    public CComCoClass<CBitmapComparer, &CLSID_BitmapComparer>,
    public IDispatchImpl<IBitmapComparer, &IID_IBitmapComparer,
        &LIBID_SampleCustomComparerLib, /*wMajor =*/ 1, /*wMinor =*/
        0>,
    public IDispatchImpl<IBitmapCompareConfiguration,
        &__uuidof(IBitmapCompareConfiguration),
        &LIBID_BitmapComparerLib, /* wMajor = */ 1, /*wMinor = */ 0>,
    public IDispatchImpl<IVerifyBitmap, &__uuidof(IVerifyBitmap),
        &LIBID_BitmapComparerLib, /* wMajor = */ 1, /*wMinor = */ 0>
{
public:
    CBitmapComparer()
    {
    }
    DECLARE_REGISTRY_RESOURCEID(IDR_BITMAPCOMPARER)
    BEGIN_COM_MAP(CBitmapComparer)
        COM_INTERFACE_ENTRY(IBitmapComparer)
        COM_INTERFACE_ENTRY2(IDispatch, IBitmapCompareConfiguration)
        COM_INTERFACE_ENTRY(IBitmapCompareConfiguration)
        COM_INTERFACE_ENTRY(IVerifyBitmap)
    END_COM_MAP()
    DECLARE_PROTECT_FINAL_CONSTRUCT()
    HRESULT FinalConstruct()
    {
        return S_OK;
    }
    void FinalRelease()
    {}
    // IBitmapCompareConfiguration Methods
public:
    STDMETHOD(GetDefaultConfigurationString)(BSTR * pbstrConfiguration);
    STDMETHOD(GetHelpFilename)(BSTR * pbstrFilename);
    // IVerifyBitmap Methods
public:
    STDMETHOD(CompareBitmaps)(IPictureDisp * pExpected,
        IPictureDisp * pActual, BSTR bstrConfiguration, BSTR * pbstrLog,
        IPictureDisp ** ppDiff, VARIANT_BOOL * pbMatch);
};

OBJECT_ENTRY_AUTO(__uuidof(BitmapComparer), CBitmapComparer)

```

5 Implement the bitmap checkpoint comparer interface methods to customize the bitmap checkpoint as required

In this tutorial, you implement a custom comparer similar to the sample custom comparer provided with QuickTest. For more information about the sample custom comparer, see "How to Use the Bitmap Checkpoint Customization Samples" on page 1836.

When you create your own custom comparers, this is the step during which you design the custom comparer logic. You define the configuration input that it can receive, the algorithm that it uses to compare the bitmaps, and the output that it provides.

In the **BitmapComparer.cpp** file, add `#include <atlstr.h>`, and implement the bitmap checkpoint comparer interface methods as follows:

- The **GetDefaultConfigurationString** method:

```
STDMETHODIMP CBitmapComparer::GetDefaultConfigurationString
(BSTR * pbstrConfiguration)
{
    CComBSTR bsConfig("MaxSurfAreaDiff=140000");
    *pbstrConfiguration = bsConfig.Detach();
    return S_OK;
}
```

- The **GetHelpFilename** method: (If you copy and paste the code from this PDF, make sure to remove the line break and tabs from the filename string.)

```
STDMETHODIMP CBitmapComparer::GetHelpFilename(BSTR * pbstrFilename)
{
    CComBSTR bsFilename
    ("..\samples\BitmapCPSample\CPPCustomComparer\
        SampleComparerDetails.txt");
    *pbstrFilename = bsFilename.Detach();
    return S_OK;
}
```

Note: When the **GetHelpFilename** method returns a relative path, QuickTest searches for this path relative to **<QuickTest installation folder>\bin**. The implementation above instructs QuickTest to use the documentation file provided with the CPP sample custom comparer.

► The **CompareBitmaps** method:

```

STDMETHODIMP CBitmapComparer::CompareBitmaps
    (IPictureDisp * pExpected, IPictureDisp *
pActual,
     BSTR bstrConfiguration, BSTR * pbstrLog,
     IPictureDisp ** ppDiff, VARIANT_BOOL *
pbMatch)
{
    HRESULT hr = S_OK;
    if (!pExpected || !pActual)
        return S_FALSE;
    CComQIPtr<IPicture> picExp(pExpected);
    CComQIPtr<IPicture> picAct(pActual);

    // Try to get HBITMAP from IPicture
    HBITMAP HbmpExp, HbmpAct;
    hr = picExp->get_Handle((OLE_HANDLE*)&HbmpExp);
    if (FAILED(hr))
        return hr;
    hr = picAct->get_Handle((OLE_HANDLE*)&HbmpAct);
    if (FAILED(hr))
        return hr;
    BITMAP ExpBmp = {0};
    if( !GetObject(HbmpExp, sizeof(ExpBmp), &ExpBmp) )
        return E_FAIL;
    BITMAP ActBmp = {0};
    if( !GetObject(HbmpAct, sizeof(ActBmp), &ActBmp) )
        return E_FAIL;

    CString s, tol;
    tol = bstrConfiguration;
    int EPos = tol.ReverseFind('=');
    tol = tol.Right(tol.GetLength() - EPos - 1);
    int maxSurfaceAreaDiff = _toi(tol);
    // Set output parameters
    CComPtr<IPictureDisp> Diff(pActual);
    *ppDiff = Diff;
    int DiffPixelsNumber = abs (ExpBmp.bmHeight * ExpBmp.bmWidth -
                                ActBmp.bmHeight * ActBmp.bmWidth);
    *pbMatch = DiffPixelsNumber <= maxSurfaceAreaDiff;
    s.Format(_T("The number of different pixels is: %d."), DiffPixelsNumber);
    CComBSTR bs (s);
    *pbstrLog = bs.Detach();
    return hr;
}

```

6 Design your custom comparer to register to the component category for QuickTest bitmap comparers

For QuickTest to recognize the COM object that you create as a custom comparer, you must register it to the component category for QuickTest bitmap comparers. The component category ID is defined in **<QuickTest installation folder>\dat\BitmapCPCustomization\ComponentCategory.h**.

You can implement this registration in the **DllRegisterServer** and **DllUnregisterServer** methods in the **SampleCPPCustomComparer.cpp** file that the wizard created as part of your project. These methods are called when you run a DLL using the **regsvr32.exe** program.

- a** Add the **<QuickTest installation folder>\dat\BitmapCPCustomization** folder to your project's include path.
- b** Open the **SampleCPPCustomComparer.cpp** file and add the following line: `#include "ComponentCategory.h"`
- c** In the **SampleCPPCustomComparer.cpp** file, modify the **DllRegisterServer** and **DllUnregisterServer** methods created by the wizard, to contain the following code:

```
STDAPI DllRegisterServer(void)
{
    // registers object, typelib and all interfaces in typelib
    HRESULT hr = _AtModule.DllRegisterServer();

    CComPtr<ICatRegister> spReg;
    hr = spReg.CoCreateInstance(
        CLSID_StdComponentCategoriesMgr, 0, CLSCTX_INPROC);
    if (FAILED(hr))
        return hr;

    // register comparer to the QuickTest bitmap comparers category
    CATID catid = CATID_QTPBitmapComparers;
    hr = spReg->RegisterClassImplCategories(CLSID_BitmapComparer, 1,
    &catid);

    return hr;
}
```

```

STDAPI DllUnregisterServer(void)
{
    HRESULT hr = _AtlModule.DllUnregisterServer();
    CComPtr<ICatRegister> spReg;
    hr = spReg.CoCreateInstance(
        (CLSID_StdComponentCategoriesMgr, 0, CLSCTX_INPROC);
    if (FAILED(hr))
        return hr;

    // unregister comparer from the QuickTest bitmap comparers category
    CATID catid = CATID_QTPBitmapComparers;
    hr = spReg->UnRegisterClassImplCategories(CLSID_BitmapComparer, 1,
    &catid);

    return hr;
}

```

Note the second section in these methods, that handles registration to the component category for QuickTest bitmap comparers—**CATID_QTPBitmapComparers**.

7 Compile your DLL and run it using the **regsvr32.exe** program

Your custom comparer can now be used in QuickTest for bitmap checkpoints.

8 Test your custom comparer by using it for bitmap checkpoints in QuickTest

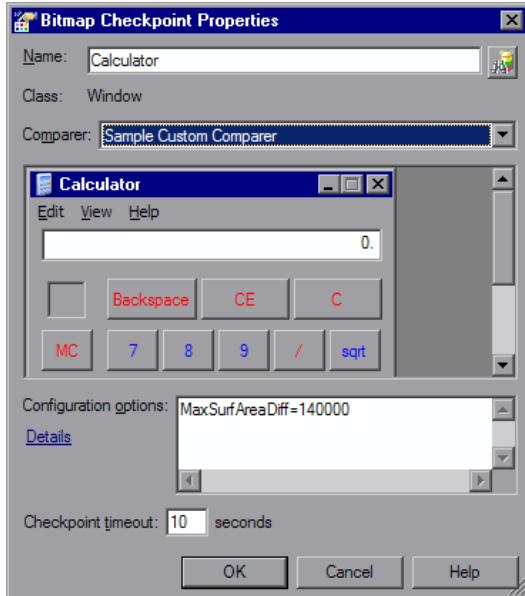
For information on how to work with bitmap checkpoints, see Chapter 17, "Bitmap Checkpoints."

- a** Open QuickTest and create a bitmap checkpoint on the Windows Calculator application (Standard view).

The Bitmap Checkpoint Properties dialog box includes the **Comparer** option, in which you can select the QuickTest default comparer or your sample custom comparer.

- b** Change the Calculator view to **Scientific**. The size of the calculator object is now larger. Run the checkpoint using the default QuickTest comparer. The checkpoint fails.

- c Edit the checkpoint and select **Sample Custom Comparer** in the **Comparer** box.



In the Bitmap Checkpoint Properties dialog box, in the **Configuration options** box, you can see the default configuration string returned by the **GetDefaultConfigurationString** method: MaxSurfAreaDiff=140000

If you click **Details**, the text file containing documentation for the sample custom comparer opens.

The comparer you designed in this exercise checks how different the expected and actual bitmaps are in size, and fails the checkpoint if the difference is greater than the number of pixels defined in the configuration string.

If you run the checkpoint using default MaxSurfAreaDiff value, the checkpoint passes, because the difference in the total size of the calculator object when it is set to different views is less than 140000 pixels (the difference is approximately 80000 pixels). If you set MaxSurfAreaDiff to 70000, the checkpoint fails.

View the run results to see the text string and difference bitmap that your custom comparer provides to QuickTest after the comparison.

Reference

The Bitmap Checkpoint Comparer Interfaces

Your custom comparer must implement the interfaces described in this section. QuickTest calls these interfaces' methods when creating or running a bitmap checkpoint that uses your custom comparer.

This section includes:

- "The IVerifyBitmap Interface" on page 1851
- "The IBitmapCompareConfiguration Interface" on page 1853

The IVerifyBitmap Interface

This interface contains the **CompareBitmaps** method that you need to implement to perform the bitmap comparison for the checkpoint.

The CompareBitmaps Method

The **CompareBitmaps** method receives the actual and expected bitmaps that need to be compared for the bitmap checkpoint, and a string that can contain configuration input for the custom comparer.

The method must compare the bitmaps according to the comparison algorithm for which this custom comparer is designed, and return the results to QuickTest.

The results include:

- An indication whether the bitmaps match and the checkpoint should pass.
- A text string that contains information about the results of the bitmap comparison.
- A bitmap that reflects the differences between the actual and expected bitmaps.

QuickTest displays the results that this method returns in the Run Results Viewer. For more information, see "Bitmap Checkpoint Results" on page 1180.

Method syntax:

```
HRESULT CompareBitmaps ([in] IPictureDisp* pExpected,  
                      [in] IPictureDisp* pActual,  
                      [in] BSTR bstrConfiguration,  
                      [out] BSTR* pbstrLog,  
                      [out] IPictureDisp** ppDiff,  
                      [out, retval] VARIANT_BOOL* pbMatch);
```

Method Parameters:

- *pExpected*. A picture object (input).
The expected bitmap stored in the checkpoint.
- *pActual*. A picture object (input).
The actual bitmap captured from the application being tested.
- *bstrConfiguration*. A text string (input).
A string that contains configuration input for the custom comparer. This is the string displayed in the **Configuration options** box in the Bitmap Checkpoint Properties dialog box.
The string can be the default configuration string that the custom comparer provides to QuickTest in the **GetDefaultConfigurationString** method described below, or an input string entered by the QuickTest user.
The **bstrConfiguration** string can have any format you choose (XML, comma separated, ini file style, and so on). Make sure that the default configuration string returned by the **GetDefaultConfigurationString** method matches the format expected in the **CompareBitmaps** method. Additionally, make sure that the documentation you provide for your custom comparer explains the format that the QuickTest user must use when editing this string in the **Configuration options** box.
- *pbstrLog*. A text string (output).
A string that contains information about the results of the bitmap comparison. QuickTest displays this string in the Run Results Viewer.

- *ppDiff*. A picture object (output).

A bitmap (created by the custom comparer) that reflects the difference between the actual and expected bitmaps. QuickTest displays this bitmap in the Run Results Viewer along with the actual and expected bitmaps.

- *pbMatch*. A boolean value (output).

A value that indicates whether the bitmaps match and the checkpoint should pass.

Possible values:

VARIANT_TRUE. Actual and expected bitmaps match, checkpoint passes.

VARIANT_FALSE. Actual and expected bitmaps do not match, checkpoint fails.

Return Value

The HRESULT that this method returns indicates whether the comparison ran successfully (and not whether the bitmaps match).



The IBitmapCompareConfiguration Interface

This interface contains the methods that you need to implement to support the custom comparer options that QuickTest displays in the Bitmap Checkpoint Properties dialog box. For more information, see "Bitmap Checkpoint Properties Dialog Box" on page 626.

The GetDefaultConfigurationString Method

The **GetDefaultConfigurationString** method must return the default configuration string for your custom comparer. For more information on configuration strings, see "Implement the CompareBitmaps method to accept input and compare bitmaps" on page 1829.

QuickTest displays this string in the **Configuration options** box in the Bitmap Checkpoint Properties dialog box when a user creating a new bitmap checkpoint selects your custom comparer.

If the QuickTest user does not modify the configuration string in the dialog box, the string provided by **GetDefaultConfigurationString** is passed to the custom comparer's **CompareBitmaps** method. You must therefore make sure that the default configuration string matches the format that your custom comparer expects to receive in the **CompareBitmaps** method.

Method syntax:

```
HRESULT GetDefaultConfigurationString ([out, retval] BSTR* pbstrConfiguration);
```

The **GetHelpFilename Method**

The **GetHelpFilename** method must return a path to the documentation that contains information about your custom comparer for QuickTest users.

QuickTest displays the documentation when a user selects your custom comparer in the Bitmap Checkpoint Properties dialog box and clicks **Details**. Make sure that when your custom comparer is installed, the documentation that you provide is installed in the location specified by the **GetHelpFilename** method.

The path can be one of the following:

- A full path to a file.
- A relative path to a file (QuickTest searches for this path relative to <QuickTest installation folder>\bin).
- A URL.

If you do not provide documentation for your custom comparer, this method should return the HRESULT **E_NOTIMPL**. For more information on the type of information you should provide, see "Implement IBitmapCompareConfiguration to provide information for the Bitmap Checkpoint Properties dialog box" on page 1830.

Method syntax:

```
HRESULT GetHelpFilename ([out, retval] BSTR* pbstrFilename);
```