

HP QuickTest Professional

Software Version: 11.00

Add-ins 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	
Add-ins Guide	15
HP QuickTest Professional Add-ins Guide Overview	15
Documentation Library Contents	17
Additional Online Resources	20

PART I: WORKING WITH QUICKTEST PROFESSIONAL ADD-INS

Chapter 1: Working with QuickTest Add-ins	23
About Working with QuickTest Add-Ins	24
Loading QuickTest Add-ins	28
The Record and Run Settings Dialog Box: Overview	37
Tips for Working with QuickTest Add-ins	42
QuickTest Add-in Extensibility	43
Chapter 2: Testing Web-Based Applications	45
About Testing Web-Based Applications	46
Setting Web Record and Run Options	46
Defining Record and Run Variables for a Web Environment	51
Setting Web Testing Options	52
Defining Web Settings for Your Test	69
Defining Web Settings for Your Application Area	71
Viewing Web Settings for Your Business Component	73
Web Event Recording Configurations	74
Understanding Web Object Identifiers	78
Accessing Custom Properties of Web-Based Objects	85
Chapter 3: Testing Windows-Based Applications	87
About Testing Windows-Based Applications	87
Setting Windows Applications Record and Run Options	89
Defining Record and Run Variables for a Windows-Based Environment	101
Setting Windows Application Testing Options	103
Setting Advanced Windows Applications Options	107

PART II: THE .NET ADD-IN

Chapter 4: Using the Silverlight Add-in..... 119
Silverlight Add-in Extensibility 121
Troubleshooting and Limitations - Silverlight..... 123

Chapter 5: Testing .NET Web Forms Applications 125
Considerations for Testing .NET Web Forms Applications 127
Checking .NET Web Forms Objects and Outputting Values 128
Troubleshooting and Limitations - .NET Web Forms 129

Chapter 6: Testing .NET Windows Forms Applications..... 133
Considerations for Testing .NET Windows Forms Applications 135
Checking .NET Windows Forms Objects and Outputting Values 136
Using the .NET Windows Forms Spy 139
.NET Add-in Extensibility..... 151
Troubleshooting and Limitations - .NET Windows Forms 152

Chapter 7: Using the Windows Presentation Foundation Add-in.... 155
Considerations for Working with the WPF Add-in 157
About WPF User Interface Automation 158
Checking WPF Objects and Outputting Values..... 159
Using WPF Objects, Methods, and Properties to Enhance Your
Test or Component 160
WPF Add-in Extensibility 162
Troubleshooting and Limitations -
Windows Presentation Foundation 163

PART III: THE ACTIVEX ADD-IN

Chapter 8: Using the ActiveX Add-in..... 167
Considerations for Working with the ActiveX Add-in 170
Troubleshooting and Limitations - ActiveX Add-in 171

PART IV: THE DELPHI ADD-IN

Chapter 9: Using the Delphi Add-in 177
Enabling Communications Between QuickTest Professional and
Your Delphi Application..... 180
Delphi Add-in Extensibility 182

PART V: THE JAVA ADD-IN

Chapter 10: Using the Java Add-in.....	187
Considerations for Working with the Java Add-in	189
Understanding Java Add-in Dependencies and Conflicts	191
Java Add-in Extensibility	191
Chapter 11: Creating and Running Tests on Java Objects	193
Defining Java Testing Options	194
Defining Java Settings for Individual Tests and Components.....	203
Defining Java Record and Run Options for Tests	209
Defining Application Details Environment Variables for Tests.....	214
Optimizing Settings for Other Record and Run Settings	
Dialog Box Tabs	215
Recording Tests and Components on Java Objects	216
Chapter 12: Using Advanced Java Test Object Methods.....	225
Creating Objects in Your Applet or Application (Advanced)	225
Working with Static Members.....	226
Firing Java Events	227
Chapter 13: Troubleshooting Testing Java Applets and	
Applications	229
Identifying and Solving Common Problems	230
Checking Java Environment Variables Settings.....	233
Locating the Java Console.....	235
Running an Application or Applet with the Same Settings.....	237
Running the Java Add-in on Multiple Environments	238
Disabling Dynamic Transformation Support (Advanced)	239
Additional Notes and Limitations.....	241

PART VI: THE ORACLE ADD-IN

Chapter 14: Using the Oracle Add-in.....249
Considerations for Working with the Oracle Add-in251
Verifying Whether the Oracle Server Supplies
 Unique Name Attributes252
Enabling the Oracle Name Attribute.....253

Chapter 15: Troubleshooting Testing Oracle Applications255
Identifying and Solving Common Problems256
Checking Oracle Environment Settings.....257
Locating the Java Console.....258
Understanding Dynamic Transformation Support.....259
Disabling Dynamic Transformation Support (Advanced)260
General Notes & Limitations262

**Chapter 16: Creating and Running Steps on
Oracle Applications265**
About Creating and Running Steps on Oracle Applications265
Defining Record and Run Settings for Oracle Tests266
Creating Steps on Oracle Applications.....272

PART VII: THE PEOPLESOFT ADD-IN

Chapter 17: Using the PeopleSoft Add-in277
Considerations for Working with the PeopleSoft Add-in.....279
Troubleshooting and Limitations - PeopleSoft Add-in280

PART VIII: THE POWERBUILDER ADD-IN

Chapter 18: Using the PowerBuilder Add-in283
Considerations for Working with the PowerBuilder Add-In285
Troubleshooting and Limitations - PowerBuilder Add-in286

PART IX: THE ADD-IN FOR SAP SOLUTIONS

Chapter 19: Using the Add-in for SAP Solutions on Web-based SAP Applications	289
Recording Tests on Web-based SAP Applications	292
Troubleshooting and Limitations - Web-based SAP Support.....	296
Chapter 20: Enhancing Your SAP Web Test	299
Checking SAP Web Objects and Outputting Values.....	299
Chapter 21: Adding SAP Web Statements to Your Test or Component	303
Working with SAP Web Test Objects	303
Chapter 22: Setting Up Your SAP GUI for Windows Environment	313
About Setting Up Your SAP Windows Environment	314
Installing SAP GUI Scripting Support.....	315
Checking Package and Patch Versions Installed on the SAP Application Server.....	316
Checking the Patch Version Installed on your SAP GUI for Windows Application.....	321
Enabling Scripting on the SAP Application (Server-Side)	322
Enabling Scripting on the SAP Application (Client-Side)	326
Setting F4 Help to Use Dialog Display Mode	330
Setting F1 Help to Use Modal Dialog Box Mode.....	332
Checking the Connection Speed on the SAP Server.....	333
Chapter 23: Using the Add-in for SAP Solutions on SAP GUI for Windows Applications	335
Considerations for Working with the Add-in for SAP Solutions	338
Understanding QuickTest and the SAP GUI Scripting API	339
Setting Record and Run Settings for SAP GUI for Windows Tests	342
Configuring Testing Options for SAP GUI for Windows Applications.....	347
Understanding Low-Level or Analog Mode Recording on SAP GUI for Windows.....	359
Using Standard Windows Recording Capabilities	359
Understanding QuickTest-eCATT Integration	360
Configuring eCATT to Work with QuickTest	363
Working with eCATT in Standalone Mode.....	365
Working with eCATT in Integrated Mode	390
Troubleshooting and Limitations - SAP Windows.....	408

Chapter 24: Enhancing Your SAP Windows Test	413
Considerations for Enhancing SAP Windows Tests.....	413
Checking SAP Windows Objects and Outputting Values.....	414
Outputting SAP Windows Property and Table Cell Values	420
Chapter 25: Adding SAP Windows Statements to Your Test or Component	431
Working with SAP Windows Test Objects	432
Accessing Native Operations and Properties in Your SAP GUI for Windows Application.....	443

PART X: THE SIEBEL ADD-IN

Chapter 26: Using the Siebel Add-in	447
Considerations for Working with the Siebel Add-in	450
Setting Up Your Siebel 7.7.x or Later Environment.....	452
Chapter 27: Creating and Running Tests and Components on Siebel Objects	455
Understanding the Siebel Test Object Model	456
Setting Siebel Record and Run Options	458
Setting Siebel Application Options for Components	464
Using Environment Variables to Specify Record and Run or Applications Settings	464
Recording Steps on Siebel Objects	466
Information for Users of Earlier Versions of the QuickTest Professional Siebel Add-in	467
Troubleshooting & Limitations - Siebel Add-in	468
Chapter 28: Enhancing Your Siebel Test or Component	473
Considerations for Checking Siebel Objects.....	473
Accessing Native Operations and Properties in Siebel 7.0.x and 7.5.x Applications.....	475
Spooling Data from a Siebel Table	476
Chapter 29: Generating an Object Repository Using Siebel Test Express	479
About Generating an Object Repository Using Siebel Test Express.....	480
Siebel Test Express System Requirements and Supported Environments.....	481
Using Siebel Test Express to Create an Object Repository	481
Using Siebel Test Express to Update an Existing Object Repository	490

PART XI: STANDARD WINDOWS TESTING SUPPORT

Chapter 30: Using Standard Windows Testing Support	497
--	-----

PART XII: THE STINGRAY ADD-IN

Chapter 31: Using the Stingray Add-in.....	503
Considerations for Working with the Stingray Add-in	506
Setting Up Stingray Object Support	507
Configuring Stingray Options.....	521
Troubleshooting and Limitations - Stingray Add-in.....	527

PART XIII: THE TERMINAL EMULATOR ADD-IN

Chapter 32: Using the Terminal Emulator Add-in	533
Using the Terminal Emulator Configuration Wizard.....	536
Copying Existing Configurations.....	550
Setting Your HLLAPI Terminal Emulator to Work with QuickTest	552
Chapter 33: Testing Terminal Emulator Applications	559
About Testing Terminal Emulator Applications	561
Modifying Your Terminal Emulator Settings.....	562
Validating your Terminal Emulator Configuration	564
Understanding the Test Object Model.....	568
Identifying Test Object Classes for Terminal Emulators.....	569
Understanding Terminal Emulator Recovery Scenarios	573
Recording Tests and Components on Terminal Emulator Applications.....	574
Troubleshooting and Limitations - Terminal Emulator	576
Chapter 34: Enhancing Your Terminal Emulator Tests and Components.....	585
Working with Checkpoints and Output Values	586
Synchronizing the Run Session.....	587
Identifying Test Object Classes and Icons	591
Chapter 35: Adjusting Your Terminal Emulator Configuration Settings	593
Using the Terminal Emulator Configuration Adjustments Dialog Box.....	594
Understanding the Configuration Adjustment Options	597

PART XIV: THE VISUAL BASIC ADD-IN

Chapter 36: Using the Visual Basic Add-in609
Troubleshooting and Limitations - Visual Basic Add-in..... 612

PART XV: THE VISUALAGE SMALLTALK ADD-IN

Chapter 37: Using the VisualAge Smalltalk Add-in615
Configuring the VisualAge Smalltalk Add-in..... 618

PART XVI: THE WEB ADD-IN

Chapter 38: Using the Web Add-in.....623
Considerations for Working with the Web Add-in 626
Working with Web Browsers..... 627
Checking Web Pages 633
Accessibility Checkpoints -
 Checking Web Content Accessibility 646
Accessing Password-Protected Resources in the Active Screen 650
Activating Methods Associated with a Web Object..... 656
Using Programmatic Descriptions for the WebElement Object 657
Registering Browser Controls 658
Web 2.0 Toolkit Support 659
Web Add-in Extensibility 665
Extensibility Accelerator for HP Functional Testing 666

**Chapter 39: Configuring Web Event Recording for
 Web Objects.....675**
About Configuring Web Event Recording 676
Selecting a Predefined Event Recording Configuration..... 678
Customizing the Web Event Recording Configuration..... 680
Recording Right Mouse Button Clicks 690
Saving and Loading Custom Event Configuration Files..... 694
Resetting Event Recording Configuration Settings..... 696

PART XVII: THE WEB SERVICES ADD-IN

Chapter 40: Using the Web Services Add-in	701
About the Web Services Add-in.....	703
Considerations for Working with the Web Services Add-in.....	704
Understanding the Web Service Testing Wizard	706
Checking that Your WSDL Meets WS-I Standards.....	723
Using the Web Service Add Object Wizard.....	727
Specifying the Web Services Toolkit	731
Setting Web Services Test Options	733
Defining Web Service Test or Component Settings.....	736
Working with Web Service Operations.....	737
Working with Business Process Testing	742
Analyzing the Results of a Web Service Test.....	743
Introduction to HP Service Test and HP Service Test Management.....	746
Troubleshooting and Limitations - Web Services	747
Chapter 41: Working with XML Data	749
About Working with XML Data	749
Checking XML.....	750
Outputting XML Values	752
Working with XML Structures	753
Parameterizing XML Values	763
Working with XML Data Operations.....	765

PART XVIII: APPENDIX

Appendix A: Supported Checkpoints and Output Values Per Add-in	769
Supported Checkpoints.....	770
Supported Output Values.....	772

Table of Contents

Welcome to the HP QuickTest Professional Add-ins Guide

This chapter includes:

- "HP QuickTest Professional Add-ins Guide Overview" on page 15
- "Documentation Library Contents" on page 17
- "Additional Online Resources" on page 20

HP QuickTest Professional Add-ins Guide Overview

Welcome to the *HP QuickTest Professional Add-ins Guide*.

This guide explains how to set up support for, and work with, the QuickTest Professional add-ins and standard Windows testing support, enabling you to test any supported environment using QuickTest Professional tests and components. The guide begins with an introductory section that describes working with QuickTest Professional add-ins, and specific aspects of working with Windows-based and Web-based add-ins. After this overview section, and the section on standard Windows testing support, the add-ins are presented alphabetically.

This guide assumes that you are familiar with QuickTest features and options. It describes the functionality that is added or changes in QuickTest when you work with specific QuickTest add-ins as well as other add-in-specific considerations and best practices.

This guide should be used in conjunction with the *HP QuickTest Professional User Guide* or *HP QuickTest Professional for Business Process Testing User Guide*, and the *HP QuickTest Professional Object Model Reference*.

The information, examples, and screen captures in this guide often focus specifically on working with QuickTest tests. However, much of the information applies equally to business components and scripted components. Information that is unique to using a specific QuickTest Professional add-in with Business Process Testing is indicated as such.

Note: Business components and scripted components are part of HP Business Process Testing, which utilizes a keyword-driven methodology for testing applications. For more information, see the section on working with Business Process Testing in the *HP QuickTest Professional User Guide*, and the *HP QuickTest Professional for Business Process Testing User Guide*.

For users that work with QuickTest add-in extensibility, QuickTest also provides developer guides that describe how to extend QuickTest support for third-party and custom controls for supported environments, such as Delphi, Java, .NET, or Web. For more information, see the relevant Add-in Extensibility Help, available from the QuickTest Professional Extensibility Documentation program group (**Start > Programs > HP QuickTest Professional > Extensibility > Documentation**). Printer-friendly (PDF) versions of the developer guides are available in the **<QuickTest Professional installation folder>\help\Extensibility** folder.

Prerequisite Background

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

In addition, because each QuickTest add-in takes advantage of commonly used QuickTest features such as the object repository, Keyword View, and checkpoints and output value steps, you should also have at least a basic understanding of these concepts before you begin working with a QuickTest add-in.

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
<p>Reference Documentation</p>	<ul style="list-style-type: none"> ▶ HP Mercury Product Advanced References contains documentation for the following QuickTest COM and XML references: <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

Working with QuickTest Professional Add-ins

1

Working with QuickTest Add-ins

QuickTest Professional includes built-in support for testing standard Windows applications. You can install and load add-ins from the QuickTest Professional setup, enabling QuickTest to recognize objects in the corresponding development environments and to provide functionality appropriate for that environment. Standard Windows testing support is automatically loaded when QuickTest opens.

When you work with these add-ins, you can use special methods, properties, and various special options to create the best possible test or component for your application.

This chapter includes:

- About Working with QuickTest Add-Ins on page 24
- Loading QuickTest Add-ins on page 28
- The Record and Run Settings Dialog Box: Overview on page 37
- Tips for Working with QuickTest Add-ins on page 42
- QuickTest Add-in Extensibility on page 43

About Working with QuickTest Add-Ins

QuickTest add-ins help you to create and run tests and components on applications in a variety of development environments. After you load an add-in, you can record and run tests or components on applications in the corresponding development environment, similar to the way you do with any other application.

You can install QuickTest add-ins when you install QuickTest Professional, or you can install the add-ins at a later time by running the installation again in **Modify** mode.

Your QuickTest Professional license enables all QuickTest features, including the use of all QuickTest add-ins. You can use the latest released version of all QuickTest add-ins with QuickTest Professional. If upgrading from a version earlier than 9.5, only licensed add-ins will be available. Additional non-licensed add-ins that you install will be disabled in the Add-in Manager dialog box.

- ▶ For more information about installing and loading add-ins, see "Considerations When Working with Add-ins" on page 26.
- ▶ For more information on installing add-ins and licenses, see the *HP QuickTest Professional Installation Guide*.

When QuickTest opens, you can choose which of the installed add-ins you want to load using the QuickTest Professional Add-In Manager dialog box, but to maximize performance, you should load only the add-ins you need for that testing session.

You can check whether a specific add-in is installed by choosing **Help > About QuickTest Professional**. Loaded add-ins are indicated by a check mark in the add-ins list.

When you load an add-in, QuickTest recognizes the objects you work with on the corresponding environment. In many cases, loading the add-in also adds new user interface options and capabilities to QuickTest, as well as adding support for the add-in's **object model**—the set of test objects, methods, and properties specially designed for working with the objects in your development environment. Details of these objects, methods, and properties can be found in the relevant section of the *HP QuickTest Professional Object Model Reference* (Select **Help > QuickTest Professional Help**).

You can use the Keyword View and Expert View to activate environment-specific test object and native (run-time object) operations, retrieve and set the values of properties, and check that objects exist.

You can customize the Active Screen capture settings for some of the QuickTest add-ins. When you apply custom Active Screen settings, you override your previous capture-level settings with all of the settings in the Custom Active Screen Capture Settings 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. For more information, see the section describing Active Screen capture setting options in the *HP QuickTest Professional User Guide*.

Several QuickTest Add-ins are designed to support special objects that are generally available in Web applications, such as standard Web (HTML), Siebel, .NET Web forms, and Web-based SAP objects. These add-ins are known as Web-based Add-ins. The interface options, capabilities, and other functionality that is available for the Web-based add-ins are often identical or similar. These Web-specific features are described in Chapter 2, "Testing Web-Based Applications."

Similarly, QuickTest provides a set of add-ins designed to support special objects that are generally part of Windows applications, such as .NET Windows Forms, Windows Presentation Foundation, PowerBuilder, SAP GUI for Windows, VisualAge, Stingray, and others. These add-ins are known as Windows-based Add-ins. The interface options, capabilities, and other functionality that is available for the Windows-based add-ins are often identical or similar. These Windows-specific features are described in Chapter 3, "Testing Windows-Based Applications."

Considerations When Working with Add-ins

- ▶ You must install and load an add-in to enable QuickTest to recognize objects from the corresponding environment. To load an add-in, select the add-in from the Add-in Manager dialog box that opens when you start QuickTest. If the Add-in Manager dialog box does not open when you start QuickTest, see the tip in "Loading QuickTest Add-ins" on page 28.
- ▶ For optimal performance when testing your applications, it is strongly recommended that you load *only* the required add-in or add-ins. For example, if you want to test a process that spans a Web application and a .NET application, load only the Web and .NET Add-ins. Do not load all add-ins unless you need to work with all of them. As a reminder, the tip at the bottom of the Add-in Manager changes to red text if more than three add-ins are selected.
- ▶ Some QuickTest add-ins require additional configuration after the installation is complete. Similarly, some environments may require configuration to enable QuickTest to interact with them. Configuration requirements, if any, are described in the introductory section of each relevant environment.
- ▶ Some applications must be opened prior to opening QuickTest, while some must be opened after QuickTest is opened. These requirements are described in the introductory section of each relevant environment.
- ▶ If you are testing Java, .NET Web Forms, Oracle, PeopleSoft, or Web-based SAP applications, make sure that you also load the Web Add-in. The Web Add-in is required whenever you are testing an application in a Web browser.
- ▶ When testing applications that do not contain .NET objects, it is strongly recommended that you do not load the .NET Add-in.

- ▶ If an add-in license has not yet been installed for a specific add-in, the add-in is displayed as **Not Licensed** in the **License** column of the Add-in Manager dialog box. An add-in may also be displayed as **Not Licensed** if no concurrent license server within your subnet has a registered license for the specific add-in, or if all concurrent licenses are in use (and are, therefore, unavailable). In this case, you can use the LSFORCEHOST or LSHOST variable to connect to a concurrent license server outside of the subnet that has the relevant add-in license installed on it, if one is available. For more information on connecting to concurrent license servers, see the *HP QuickTest Professional Installation Guide*.
- ▶ You can view license details for all currently loaded licensed add-ins by clicking **License** in the About QuickTest Professional dialog box (**Help > About QuickTest Professional**).
 - ▶ For seat licenses, the category for each license is displayed. The license category may be **Demo**, **Permanent**, **Commuter**, or **Time-Limited**. For **Demo**, **Commuter** (used with concurrent licenses), and **Time-Limited** QuickTest seat licenses, the number of days and hours remaining until the license expires is also displayed.
 - ▶ For concurrent licenses, the URL or host name of the concurrent license server used for each license is displayed.

To switch between a seat and a concurrent license, click **Modify License**. Note that you can use only one license type per session for QuickTest Professional and all loaded add-ins—either seat or concurrent. For more information on license types, installing licenses, and modifying licenses, see the *HP QuickTest Professional Installation Guide*.

Loading QuickTest Add-ins

To test applications developed in various environments, you must ensure that the relevant QuickTest add-in is installed and loaded on the computer on which you create and run your tests and components. Loading the relevant add-in enables QuickTest to work with the corresponding environment.

When you start QuickTest, the Add-in Manager dialog box opens. It displays a list of all installed add-ins and the license used for each add-in. If you are using a seat add-in license, it also displays the time remaining for time-limited licenses. For information on the details shown in the Add-in Manager dialog box, see "The Add-in Manager Dialog Box" on page 31.



Tips:

- ▶ 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.
 - ▶ If the Web Services Add-in is loaded, a message opens when you open QuickTest that provides a link to more information on the SOA testing capabilities available with HP Service Test and HP Service Test Management. The Add-in Manager dialog box is displayed when you click **OK** in the message box. Select the check box if you do not want the message to open each time you open QuickTest with the Web Services Add-in loaded.
-

If you have QuickTest add-ins installed, you can specify which add-ins to load at the beginning of each QuickTest session. It is recommended to load only the QuickTest add-ins you need for a particular QuickTest session, as this improves performance and object identification reliability. You can also load QuickTest without add-in support if you want to test only standard Windows-based objects.

This section includes:

- ▶ "Loading QuickTest With Add-in Support" on page 30
- ▶ "The Add-in Manager Dialog Box" on page 31
- ▶ "Loading QuickTest Without Add-in Support" on page 36
- ▶ "Matching Loaded Add-ins with Associated Add-ins" on page 36

Loading QuickTest With Add-in Support

You use the Add-in Manager to load support for testing your applications.

To start QuickTest with add-in support:

- 1** Select **Start > Programs > QuickTest Professional > QuickTest Professional**. The QuickTest Professional Add-in Manager dialog box opens.

(If the Add-in Manager dialog box does not open, see the tip in "Loading QuickTest Add-ins" on page 28.)

- 2** In the add-in list, select the required add-in. For more information about the Add-in Manager, see "The Add-in Manager Dialog Box" on page 31.

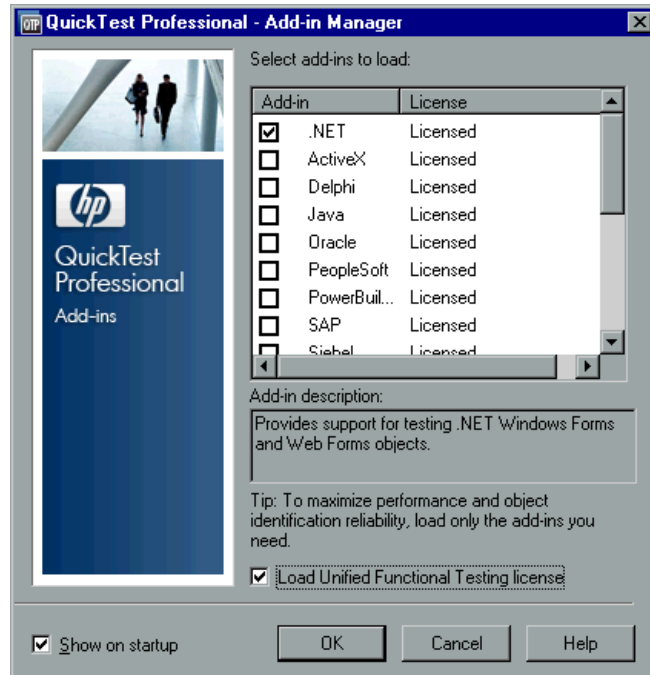
Notes:

- ▶ If you plan to test your application in a Web browser, select **Web** as well as your required add-in.
- ▶ If you want to test .NET Windows Forms, select **.NET** and click **OK**. A message is displayed stating that for full operation of the .NET Add-in you must also load the Web Add-in. If you want to test only .NET Windows Forms (and not .NET Web Forms), you can click **Yes**.
- ▶ If you load or unload an add-in that is displayed as a child of the Java add-in in the Add-in Manager, only applications that are opened after loading or unloading the add-in are affected.

-
- 3** Click **OK**.

The 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	<p>By default, this dialog box opens when you start QuickTest.</p> <p>To display the Add-in Manager if it does not open when you start QuickTest, select Tools > Options > General node and select Display Add-in Manager on startup.</p>
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
See also	Product Information Window

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>

UI Element	Description
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
<p>Load Unified Functional Testing license</p>	<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>

Additional References

- "About Working with QuickTest Add-Ins" on page 24
- "Loading QuickTest Add-ins" on page 28
- "Loading QuickTest Without Add-in Support" on page 36
- "Matching Loaded Add-ins with Associated Add-ins" on page 36
- "Tips for Working with QuickTest Add-ins" on page 42
- "Considerations When Working with Add-ins" on page 26
- "Matching Loaded Add-ins with Associated Add-ins" on page 36

Loading QuickTest Without Add-in Support

If you want to work with QuickTest without support for a specific environment, you can load QuickTest without that add-in.

To load QuickTest without add-in support:

- 1 Select **Start > Programs > HP QuickTest Professional > QuickTest Professional**. The QuickTest Professional Add-in Manager dialog box opens. (If the Add-in Manager dialog box does not open, see the tip in "Loading QuickTest Add-ins" above.)
- 2 Clear the check box for the relevant add-in and click **OK**. QuickTest opens without add-in support for that add-in environment.

Matching Loaded Add-ins with Associated Add-ins

When you open a test or component, QuickTest compares the add-ins that are currently loaded with the add-ins associated with your test or with your component's application area. If they do not match, QuickTest issues a warning message.

If there are add-ins associated with your test or with your component's application area that are not currently loaded, you can:

- ▶ Close and reopen QuickTest, and select the required add-ins in the Add-in Manager dialog box.
- ▶ Remove the add-ins from the list of associated add-ins for your test or component. To change the list of add-ins associated with your test or component, select **File > Settings** and click **Modify** in the Properties pane.

If add-ins are loaded but not associated with your test or with your component's application area, you can:

- ▶ Close and reopen QuickTest, and clear the check boxes for the add-ins in the Add-in Manager dialog box, if they are not required.
- ▶ Add the add-ins to the list of associated add-ins for your test or for your component's application area. To change the list of add-ins associated with your test or component, select **File > Settings** and click **Modify** in the Properties pane.

For more information on associating add-ins with your test or component, see the *HP QuickTest Professional User Guide* (for tests) or the *HP QuickTest Professional for Business Process Testing User Guide* (for components).

The Record and Run Settings Dialog Box: Overview

You can control how QuickTest starts record and run sessions in specific environments by setting the options in the relevant tabs of the Record and Run Settings dialog box. In some cases, the settings in this dialog box may also affect the applications that QuickTest recognizes for other QuickTest operations, such as learning objects or using the Object Spy.

For example, you can choose to have QuickTest open a specific application when you start a record or run session. You can set your record and run options in the Record and Run Settings dialog box, or you can set the options using environment variables.

- ▶ For general information on the Record and Run Settings dialog box, see "Using the Record and Run Settings Dialog Box" on page 38.
- ▶ For more information on setting preferences for recording and running tests on Windows-based applications, see "Setting Windows Applications Record and Run Options" on page 89.
- ▶ For more information on setting browser preferences for recording and running tests on Web-based environments, see "Setting Web Record and Run Options" on page 46.
- ▶ For more information on setting preferences for recording and running tests for other environments, see the relevant add-in chapter.
- ▶ For more information on setting preferences for recording and running tests using environment variables, see "Using Environment Variables to Specify the Record and Run Details for Your Test" on page 40.

Using the Record and Run Settings Dialog Box

Before you record or run a test on an application, you can use the Record and Run Settings dialog box to instruct QuickTest which applications to open when you begin to record or run your test. For some Windows-based applications, you also use the dialog box to specify the specific applications you want QuickTest to recognize during record, run, and Object Spy sessions.

You can instruct QuickTest to open and record on applications from more than one environment.

The Record and Run Settings dialog box opens automatically each time you begin recording a new test and saves your settings with the test unless you open the dialog box (**Automation > Record and Run Settings**) and set your preferences manually before you begin recording the first step in your test.

The Record and Run Settings dialog box does not open when you perform additional record sessions on an existing test or when you run the test. QuickTest automatically applies the settings already in the Record and Run Settings dialog box for that test.

The Record and Run Settings dialog box always contains the Windows Applications tab. It may contain other tabs corresponding to add-ins that are loaded. For more information on which tab of the Record and Run Settings dialog box you should use with an add-in, see the relevant add-in chapter.

To set record and run options:

- 1** Click the **Record** button or select **Automation > Record**. If you are recording for the first time in a test and have not yet set your recording preferences (by opening the dialog box manually), the Record and Run Settings dialog box opens. It is divided by environment into several tabbed pages.
- 2** To choose an environment, click a tab.
- 3** Set the required options, as described in the following sections.
- 4** To apply your changes and keep the Record and Run Settings dialog box open, click **Apply**.

5 When you are finished, click **OK** to save your changes and start recording.

Guidelines for Working with Record and Run Settings

- ▶ You can set the record and run settings for some add-in environments using the corresponding tab (displayed only when the add-in is installed and loaded).
 - ▶ For most Windows-based add-in environments, you use the Windows Applications tab. For more information, see "Setting Windows Applications Record and Run Options" on page 89.
 - ▶ For most Web-based add-in environments, you use the Web tab. For more information, see "Setting Web Record and Run Options" on page 46.

For more information on setting preferences for recording and running tests for other environments, see the relevant add-in chapter.

- ▶ The setting of the Active Screen capture level (**Tools > Options > Active Screen** pane) can significantly affect the recording time for your test and the functionality of the Active Screen while editing your test. Confirm that the level selected answers your testing needs. For more information, see the section on setting active screen options in the *HP QuickTest Professional User Guide*.
- ▶ You can set record and run options such that no applications open at the beginning of record and run sessions. In this case, you may need to open the application after you open QuickTest to ensure that QuickTest recognizes the application. For more information, see the relevant add-in chapter.
- ▶ If you define environment variables to specify the record and run details, those values override the values in the Record and Run Settings dialog box. For more information, see "Using Environment Variables to Specify the Record and Run Details for Your Test" on page 40.
- ▶ After you set the record and run settings for a test, the Record and Run settings dialog box will not open the next time you record operations in that test. If needed, you open the Record and Run Settings dialog box by choosing **Automation > Record and Run Settings**.

You should set or modify your record and run preferences in the following scenarios:

- ▶ You have already recorded one or more steps in the test and you want to modify the settings before you continue recording.
- ▶ You want to run the test on a different application than the one you previously set in the Record and Run Settings dialog box.

If you change the record and run settings for additional recording sessions, confirm that you return the settings to match the needs of the first step in your test before you run it.

Using Environment Variables to Specify the Record and Run Details for Your Test

You can use special, predefined environment variables to specify the applications or browsers you want to use for your test. This can be useful if you want to test how your application works in different environments. For example, you may want to test that your Web application works properly on identical or similar Web sites with different Web addresses.

When you define an environment variable for one (or more) of the application details, the environment variable values override any values that were added using these areas of the Record and Run Settings dialog box.

Note: If you select the option to Record and Run on any application (the upper radio button in each tab of the Record and Run Settings dialog box), QuickTest ignores any defined record and run environment variables.

You can define the environment variables as internal user-defined variables, or you can add them to an external environment variable file and set your test to load environment variables from that file.

You can set your Record and Run settings manually while recording your test and then define the environment variables or load the environment variable file only when you are ready to run the test (as described in the procedure below).

Alternatively, you can define environment variables before you record your test. In this case, QuickTest uses these values to determine which applications or browsers to open when you begin recording—assuming that the option to open an application when starting record and run sessions for the particular environment is selected. (This option corresponds to the lower radio button in each tab of the Record and Run Settings dialog box, and the third check box in the Windows Applications tab.)

To use record and run environment variables for your test:

- 1** Set your Record and Run Setting preferences normally to record your test.

Note: If you already have environment variables set for one or more application details, and you select the option to open an application when the record session begins (the lower radio button in each tab of the Record and Run Settings dialog box), QuickTest ignores the record settings you enter in the dialog box.

- 2** Record and edit your test normally.
- 3** If you did not define environment variables prior to recording your test, define an environment variable for each application detail you want to set using the appropriate variable name.

For more information on environment variables for record and run settings, see "Defining Record and Run Environment Variables" on page 42.

For more information on how to define a user-defined environment variable and how to create environment variable files, see the section on using environment variable parameters in the *HP QuickTest Professional User Guide*.

- 4** Before running the test, confirm that the lower radio button is selected in the tabs corresponding to the environments for which you want to use environment variables. If you are running a Windows application, also select the third check box.

- 5 Run the test. QuickTest uses the environment values to determine which applications to open at the beginning of the run session, and on which processes to record.

Defining Record and Run Environment Variables

To use environment variables to specify the applications or browsers you want to use for your test run, you must use appropriate variable names.

- For the variable names required to define the Web browser and URL to open, see "Defining Record and Run Variables for a Web Environment" on page 51.
- For the variable names required to define the details for Windows applications on which you want to record and run tests, see "Defining Record and Run Variables for a Windows-Based Environment" on page 101.
- For the variable names required to define the details corresponding to the options in other tabs in the Record and Run Settings dialog box, see the relevant add-in chapter in this guide.

Tips for Working with QuickTest Add-ins

To take full advantage of QuickTest add-in capabilities, keep the following in mind when designing tests or components using QuickTest add-ins:



- If the Add-in Manager does not open when you start QuickTest, click the **Options** button or select **Tools > Options** and click the **General** node. Select the **Display Add-in Manager on startup** check box and click **OK**. Restart QuickTest.
- You can view the list of add-ins that are currently installed or loaded by choosing **Help > About QuickTest Professional**. The dialog box displays a list of all add-ins installed on your computer. A check mark indicates that the add-in is currently loaded.

- ▶ You can enhance your tests and components using environment-specific checkpoints and output values. For more information, see the sections describing checkpoints and output values in the *HP QuickTest Professional User Guide* (for tests), and the *HP QuickTest Professional for Business Process Testing User Guide* (for components).
- ▶ You can add additional steps to your test using the Step Generator, or you can add them manually from the Expert View. For more information on the objects, methods, and properties available for your application's environment, see the *HP QuickTest Professional Object Model Reference*.
- ▶ When you run a QuickTest test from Quality Center, Quality Center instructs QuickTest to load the add-ins that are associated with the test. If you created the test in Quality Center (and not in QuickTest), the test contains the settings specified in the template test you chose when creating the test. If you need to modify the associated add-ins, you can do so by opening the test in QuickTest. For more information, see the template test section in the *HP QuickTest Professional User Guide*.
- ▶ Before you run a QuickTest test from Quality Center, make sure that the required QuickTest add-ins are installed on the computer on which you want to run the QuickTest test.

QuickTest Add-in Extensibility

QuickTest add-in extensibility, available for some environments, enables you to extend the relevant QuickTest add-in to support third-party and custom controls that are not supported out-of-the-box.

When QuickTest learns an object in an application, it recognizes the object as belonging to a specific test object class. This type of test object might not have certain characteristics that are specific to the control you are testing. Therefore, when you try to create test steps with this test object, the available identification properties and test object operations might not be sufficient.

By developing support for a control using Add-in Extensibility, you can direct QuickTest to recognize the control as belonging to a specific test object class, and you can specify the behavior of the test object.

You can also teach QuickTest to treat a control that contains a set of lower-level controls as a single functional control, instead of relating to each lower-level control separately. For example, a calendar control may consist of buttons and text boxes. If you teach QuickTest to recognize the control as a calendar, ignoring the individual buttons and text boxes, you can create more meaningful tests on the calendar control.

In most environments, you can also extend the list of available test object classes that QuickTest is able to recognize. This enables you to create tests that fully support the specific behavior of your controls.

QuickTest add-in extensibility is currently supported for the Delphi, Java, .NET, Silverlight, Web, and WPF add-ins.

If you cannot develop support for your controls using the extensibility options provided for these environments, you might be able to take advantage of the QuickTest Professional Testing Extensibility program. Testing Extensibility is intended for customers who want to extend QuickTest testing capabilities for technologies or applications not supported by existing QuickTest add-ins. Participation in the program requires a separate license agreement with HP.

For more information on Testing Extensibility, please contact HP Software support.

For more information on QuickTest Add-in Extensibility, see:

- "Delphi Add-in Extensibility" on page 182
- "Java Add-in Extensibility" on page 191
- ".NET Add-in Extensibility" on page 151
- "Silverlight Add-in Extensibility" on page 121
- "Web Add-in Extensibility" on page 665
- "WPF Add-in Extensibility" on page 162

2

Testing Web-Based Applications

You can use QuickTest Professional to test Web-based applications.

This chapter includes:

- ▶ About Testing Web-Based Applications on page 46
- ▶ Setting Web Record and Run Options on page 46
- ▶ Defining Record and Run Variables for a Web Environment on page 51
- ▶ Setting Web Testing Options on page 52
- ▶ Defining Web Settings for Your Test on page 69
- ▶ Defining Web Settings for Your Application Area on page 71
- ▶ Viewing Web Settings for Your Business Component on page 73
- ▶ Web Event Recording Configurations on page 74
- ▶ Understanding Web Object Identifiers on page 78
- ▶ Accessing Custom Properties of Web-Based Objects on page 85

About Testing Web-Based Applications

QuickTest provides a number of add-ins for testing Web-based applications. The way you configure many of your QuickTest settings is the same or similar for most QuickTest Web-based add-ins. These common configuration options are described in the remainder of this chapter.

For additional details on how to work with Web-based add-ins, see the specific sections describing these add-ins in the guide:

- ▶ "Using the Web Add-in" on page 623
- ▶ "Testing .NET Web Forms Applications" on page 125
- ▶ "Using the Silverlight Add-in" on page 119
- ▶ "The PeopleSoft Add-in" on page 275
- ▶ "Using the Add-in for SAP Solutions on Web-based SAP Applications" on page 289
- ▶ "The Siebel Add-in" on page 445
- ▶ "Web 2.0 Toolkit Support" on page 659

In addition to using the add-ins described above, you can also use the Extensibility Accelerator to develop your own Web-based add-in support for third-party and custom Web controls that are not supported by any of the above QuickTest Web-based add-ins. For details, see "Extensibility Accelerator for HP Functional Testing" on page 666.

Setting Web Record and Run Options

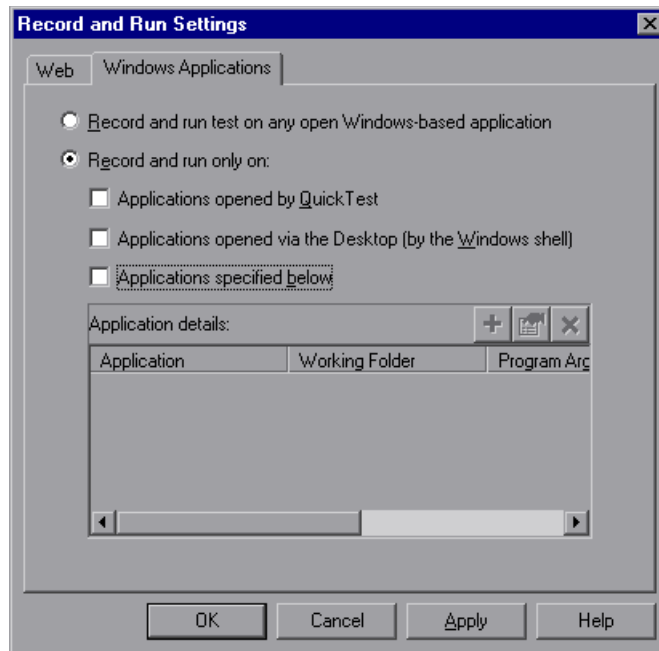
You use the Web tab in the Record and Run Settings dialog box to set options that affect how you start creating and running tests for Web-based applications.

The options in the Web tab instruct QuickTest which applications to open when you begin to record or run your test. You can instruct QuickTest to open and record on applications from more than one environment. You can create steps on more than one browser tab, if your browser supports tabbed browsing.

In addition to setting the appropriate settings in the Web tab, you should confirm that the other tabs in the dialog box have the appropriate settings.

The following settings are recommended:

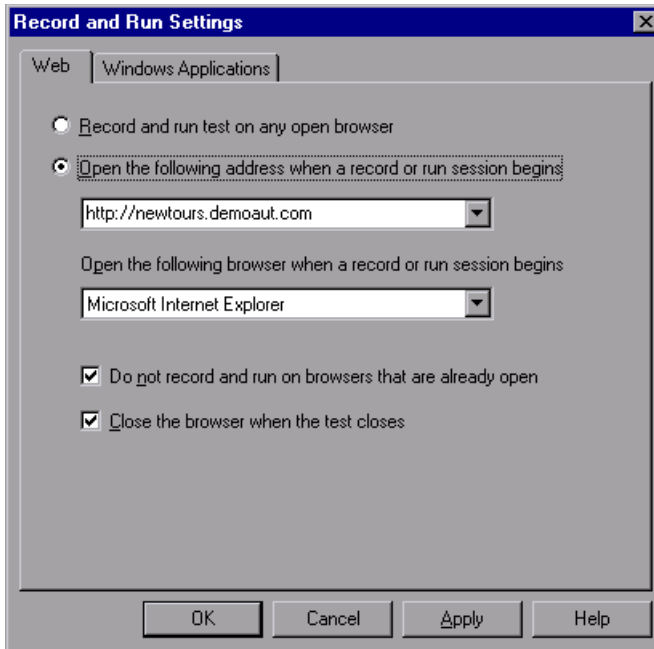
- ▶ **Windows Applications tab.** Select **Record and run only on:** and confirm that all three check boxes are cleared, as shown below.
- ▶ **Other tabs.** (If displayed.) Select the option to record and run on any open application (upper radio button of each tab).



While these settings do not directly affect your record or run sessions when working with Web-based applications, they prevent you from inadvertently recording operations performed on Windows applications (such as e-mail) during your recording session. These settings also prevent QuickTest from opening unnecessary applications when you record or run tests on Web-based applications.

Note: You can also use special, predefined environment variables to specify the applications or browsers you want to use for your test. For more information, see "Using Environment Variables to Specify the Record and Run Details for Your Test" on page 40.

The Web tab is available only when the Web add-in is installed and loaded. QuickTest uses the settings in this tab when recording and running tests or components on Web, .NET Web Forms, PeopleSoft, and Web-based SAP objects. (For Siebel objects, QuickTest uses the settings in the Siebel tab, available when the Siebel add-in is installed and loaded.) For more information, see "Setting Siebel Record and Run Options" on page 458.



The Web tab includes the following options:

Option	Description
<p>Record and run test on any open browser</p>	<p>Instructs QuickTest to record and run on any open (supported) Web browser. (For information on supported browsers, 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> <p>Note: You must open the Web browser after you open QuickTest.</p> <p>Tip: You can instruct QuickTest to ignore Quality Center browsers or other browsers that are open to a specified URL or have a specific title. For more information, see "Setting Web Testing Options" on page 52.</p>
<p>Open the following address when a record or run session begins</p>	<p>Instructs QuickTest to open a new browser session to record and run the test using the specified URL address.</p> <p>Note: If you define a value for the URL_ENV environment variable, that value overrides the value specified here during a run session. For more information, see "Using Environment Variables to Specify the Record and Run Details for Your Test" on page 40.</p>

Option	Description
<p>Open the following browser when a record or run session begins</p>	<p>Instructs QuickTest to open the specified browser type when recording or running a test:</p> <p>Notes:</p> <ul style="list-style-type: none"> ▶ Only those browsers currently installed on your computer are available in the list. ▶ If you define a value for the BROWSER_ENV environment variable, that value overrides the value specified here during a run session. For more information, see "Defining Record and Run Variables for a Web Environment" on page 51 and "Using Environment Variables to Specify the Record and Run Details for Your Test" on page 40.
<p>Do not record and run on browsers that are already open</p>	<p>Instructs QuickTest not to record or run tests on any browsers that are already open prior to the start of the record or run session (and prior to opening QuickTest). Selecting this option also prevents you from viewing the properties of these browsers using the Object Spy.</p>
<p>Close the browser when the test closes</p>	<p>Instructs QuickTest to close the browser window specified in the Address box when the test closes.</p>

Notes to users of applications with embedded Web browser controls:

- ▶ To record and run tests on an application with embedded Web browser controls, select **Record and run tests on any open Web browser** in the Record and Run Settings dialog box.
 - ▶ You must also register your browser control application (using the Register Browser Control Utility) so that QuickTest Professional recognizes your Web object when recording or running tests. For more information, see "Registering Browser Controls" on page 658.
 - ▶ Ensure that the application is opened after QuickTest, and start recording.
-

For more information on the Record and Run Settings dialog box, see "Using the Record and Run Settings Dialog Box" on page 38.

Defining Record and Run Variables for a Web Environment

You can use predefined environment variables to specify the applications or browsers you want to use for your test. This can be useful if you want to test how your application works in different environments.

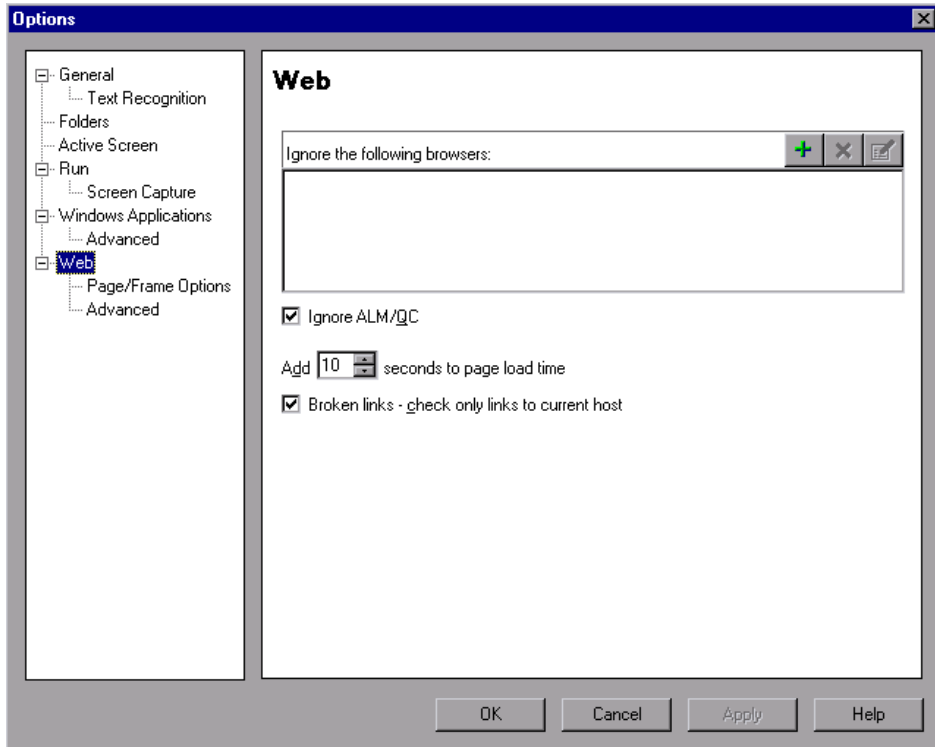
Note: For more information on environment variables and how to use them in tests, see "Using Environment Variables to Specify the Record and Run Details for Your Test" on page 40.

To use environment variables to define the Web browser and URL to open, you must use the appropriate variable names as specified below:

Option	Variable Name	Description
Type	BROWSER_ENV	The browser program to open. For example, Microsoft Internet Explorer, or Mozilla Firefox. Possible values: IE, FF30, FF35 Note: If the specified browser program is not installed, the default browser is used.
Address	URL_ENV	The Web address to display in the browser.

Setting Web Testing Options

The Web pane options in the Options dialog box (**Tools > Options > Web** node) determine how QuickTest behaves when recording and running tests or components on Web sites.



This section includes:

- ▶ "Understanding the Web Pane" on page 53
- ▶ "Managing the List of Browsers to Ignore" on page 54
- ▶ "Page and Frame Options" on page 58
- ▶ "Advanced Web Options" on page 62

Understanding the Web Pane

The Web pane includes the following options:

Option	Description
Ignore the following browsers	Instructs QuickTest to ignore any specified browsers that may be open while QuickTest is recording or running a test or component. For more information, see "Managing the List of Browsers to Ignore" on page 54.
Ignore ALM/QC	Instructs QuickTest to ignore all instances of HP ALM or Quality Center that are opened while recording or running a test or component. By default, this option is selected.
Add __ seconds to page load time	Instructs QuickTest to add a specified number of seconds to the page load time property specified in each Page checkpoint (Page checkpoints are not relevant for business components). Note: This option is a safeguard that prevents page checkpoints from failing in the event that the amount of time it takes for a page to load during the run is longer than the amount of time it took during the record session.
Broken links - check only links to current host	Instructs QuickTest to check only for broken links that are targeted to your current host.

You can also modify how QuickTest displays captured Web pages in the Active Screen. You do this in the Active Screen pane of the Options dialog box (**Tools > Options > Active Screen** node). For more information, see the section describing Active Screen options in the *HP QuickTest Professional User Guide*.

Managing the List of Browsers to Ignore

You can instruct QuickTest to ignore specific browsers that are open while you are recording or running a test or component. This enables you to keep browsers that are not related to your testing environment open, without having them affect the record or run session. For example, you may want to check your company's share price or the news headlines during the record and run session. If you instruct QuickTest to ignore these specific browsers, they do not affect the session.

Notes: QuickTest ignores browsers that match the defined criteria at the start of a record or run session. However, browsers that do not match the defined criteria at the start of a record or run session, but do match them during the session, are not ignored.

Changes made to these settings apply to new tests or components and new steps in existing tests or components only, but not to any other existing steps.

You can also modify the properties that QuickTest uses to identify the browsers to ignore, or delete them from the list of ignored browsers.

Tip: By default, QuickTest ignores all instances of Quality Center that were opened during a record or run session, if the **Ignore Quality Center** check box in the Web pane of the Options dialog box is selected. There is no need to specify Quality Center in the list of browsers to ignore.

Adding a Browser to the List

You can specify the browsers that you want QuickTest to ignore during a record or run session.

Note: When working with tests, QuickTest ignores these browsers only if you selected **Record and run test on any open Web browser** in the Web tab of the Record and Run Settings dialog box. For more information, see "Setting Web Record and Run Options" on page 46.

To add a browser to the list:



- 1 To add a browser to the list, click the **Add Browser** button. The Browser Details dialog box opens.

- 2 Enter a name for the browser definition in the **Name** field. By default, the name of the browser is **Browser<number of browser in list>**. The name you specify is used only to identify the browser in the list, and is not used by QuickTest.

- 3 Select one or both of the following properties to identify the browser to be ignored, and then enter the following details:
 - **Title.** The name of the Web page as it appears in the title bar of the browser, for example, MyBank - Finance.*
 - **URL.** The URL of the Web page, for example, http://www.finance.mybank.com
Any descendants of this Web page are automatically included in the list of browsers to ignore.
-

Tip: You can use regular expressions when specifying the values of these properties. For example, you can use `.*finance.mybank.com` to specify all `finance.yahoo.com` domains and Web sites starting with `www.`, `http://`, or `https://`. Note that you do not have to use a regular expression to include child pages of a site, as QuickTest automatically ignores the entire domain or site. For information on supported regular expressions, see the *HP QuickTest Professional User Guide*.

Note: The **Title** and **URL** properties have an AND relationship, meaning that a browser must match both property values (if defined) to be ignored by QuickTest.

- 4 Click **OK**. The browser is added to the list of ignored browsers.
- 5 Repeat steps 1 to 4 for each browser to be added to the list.

Modifying a Browser in the List

You can modify the definitions of browsers that you want QuickTest to ignore during a record or run session.

To modify a browser in the list:

- 1 Highlight the browser you want to modify.
- 2 Click the **Modify Browser Details** button. The Browser Details dialog box opens.
- 3 Make the required changes in the Browser Details dialog box and click **OK**.



Removing a Browser from the List

You can remove a browser from the list if you no longer want QuickTest to ignore it during a record or run session.

Tip: If a browser in the list is required for running a specific test, you can temporarily remove it from the list by clearing the check mark next to its name in the list of browsers.

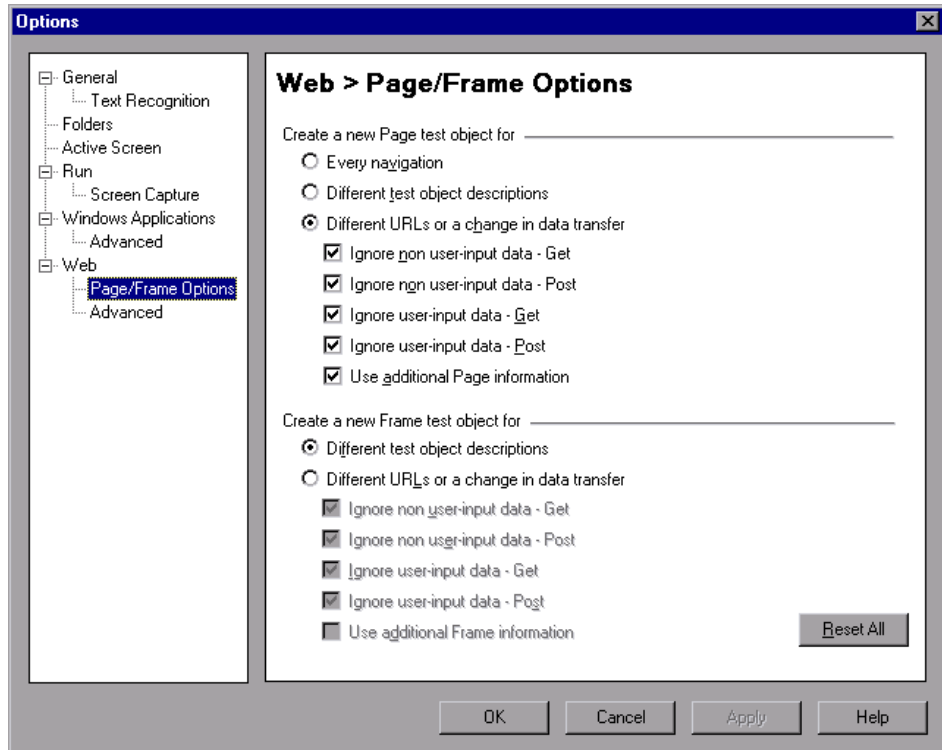
To remove a browser from the list:

- 1 Highlight the browser you want to remove from the list.
- 2 Click the **Remove Browser** button.



Page and Frame Options

The Web > Page/Frame Options pane enables you to modify how QuickTest records Page and Frame objects.



Note: You can click the **Reset** button at any time to reset all options to their default core settings. Some Web-based add-ins modify the default settings to optimize page and frame recording. If you are using an add-in, it is recommended that you keep the default add-in settings and do not use the **Reset** button.

Page Options

The **Create a new Page test object for** options instruct QuickTest when to create a new Page object in the object repository while recording.

Note: These options only affect how Page test objects are created; Frame test objects are created according to the Frame options you select. The Frame options are similar to the Page options (except that the **Every navigation** option is not available).

The following Page options are available:

- ▶ **Every navigation.** Creates a new Page object every time a navigation is performed in a Web page.
 - ▶ **Different test object descriptions.** Creates a new Page object for pages with different test object descriptions, according to the properties defined for the Page test object.
-

Note: The default test object description for Page objects includes only the test object class. If you select this option, it is highly recommended that you define object identification properties that uniquely identify different Page objects. You should also ensure that the properties you define remain constant over time, otherwise future runs may fail.

- ▶ **Different URLs or a change in data transfer.** Creates a new Page object only when the page URL changes, or if the URL stays the same and data that is transferred to the server changes, according to the data types and transfer methods you select (below).
- ▶ **Ignore non user-input data - Get.** Instructs QuickTest to ignore non user-input data if the Get method is used to transfer data to the server.

For example, suppose a user enters data on a Web page, and the data is then inserted as a hidden field using the Get method. The user clicks **Submit** (to send the data to the server). The new Web page is different, according to the hidden field data. However, QuickTest does not create a new Page test object.
- ▶ **Ignore non user-input data - Post.** Instructs QuickTest to ignore non user-input data if the Post method is used to transfer data to the server.

For example, suppose a user enters data on a Web page, and the data is then inserted as a hidden field using the Post method. The user clicks **Submit** (to send the data to the server). The new Web page is different, according to the hidden field data. However, QuickTest does not create a new Page test object.
- ▶ **Ignore user-input data - Get.** Instructs QuickTest to ignore user-input data if the Get method is used to transfer data to the server.

For example, suppose a user enters data in a form on a Web page and clicks **Submit** (to send the data to the server) using the Get method. The new Web page is different according to the data filled in by the user. However, QuickTest does not create a new Page test object.
- ▶ **Ignore user-input data - Post.** Instructs QuickTest to ignore user-input data if the Post method is used to transfer data to the server.

For example, suppose a user enters data in a form on a Web page and clicks **Submit** (to send the data to the server) using the Post method. The new Web page is different according to the data filled in by the user. However, QuickTest does not create a new Page test object.
- ▶ **Use additional Page information.** Instructs QuickTest to use additional properties of the test object to identify an existing Page test object.

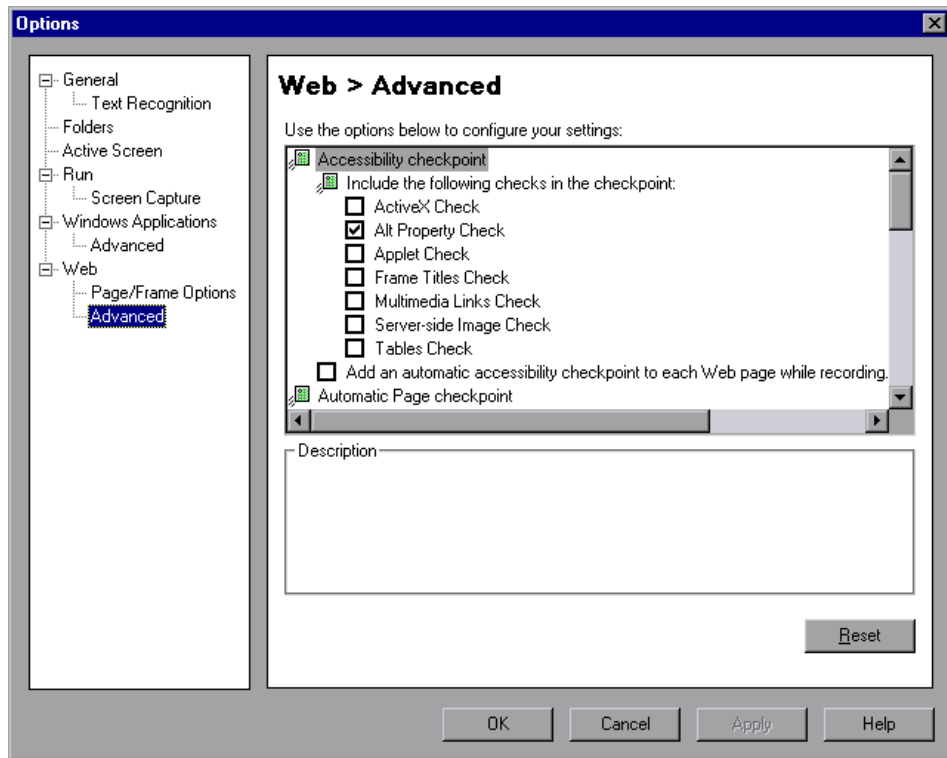
Tip: Select this option to instruct QuickTest to recognize existing pages when the **Back** and **Forward** navigation buttons are used.

Frame Options

The **Create a new Frame test object for** options instruct QuickTest when to create a new Frame object in the object repository while recording. The Frame options are similar to the Page options (except that the **Every navigation** option is not available). For more information, see "Page Options" on page 59.

Advanced Web Options

The Web > Advanced pane enables you to modify how QuickTest records and runs tests and components on Web sites. You can click the **Reset** button at any time to reset all options to their default settings.



Note: The **Accessibility checkpoint** and **Automatic Page checkpoint** options are not relevant for business components.

Accessibility Checkpoint Options in Tests

When working with tests, you can add accessibility checkpoints to check that Web pages and frames conform to the W3C Web Content Accessibility Guidelines. All accessibility checkpoints in a test use the options that are selected in this dialog box during the run session.

The Web > Advanced pane includes the following **Accessibility checkpoint** options:

- ▶ **Include the following checks in the checkpoint.** Instructs QuickTest to check the selected accessibility elements for all accessibility checkpoints. Choose from the following:
 - ▶ **ActiveX Check.** Checks whether the page or frame contains ActiveX objects. If so, QuickTest sends a warning and displays a list of the objects in the Run Results.
 - ▶ **Alt Property Check.** Checks that the <alt> attribute exists for all relevant objects (such as images). If one or more objects lack the required attribute, the test fails and QuickTest displays a list of the objects with the missing attribute in the Run Results. (Selected by default.)
 - ▶ **Applet Check.** Checks whether the page or frame contains Java objects. If so, QuickTest sends a warning and displays a list of the objects in the Run Results.
 - ▶ **Frame Titles Check.** Checks that the page and all frames in the page have titles. If one or more frames (or the page) lack the required title, the test fails and QuickTest displays a list of the frames that lack titles in the Run Results.
 - ▶ **Multimedia Links Check.** Checks whether the page or frame contains links to multimedia objects. If so, QuickTest sends a warning and displays a list of the links in the Run Results.
 - ▶ **Server-side Image Check.** Checks whether the page or frame contains Server-side images. If so, QuickTest sends a warning and displays a list of the images in the Run Results.
 - ▶ **Tables Check.** Checks whether the page or frame contains tables. If so, QuickTest sends a warning and displays the table format and the tags used in each cell in the Run Results.

For more information, see "Accessibility Checkpoints - Checking Web Content Accessibility" on page 646.

- ▶ **Add an automatic accessibility checkpoint to each Web page while recording.** Instructs QuickTest to automatically add an accessibility checkpoint to each Web page while recording, using the checks selected in the option above.

Automatic Web Page Checkpoint Options in Tests

When working with tests, you can check that expected and actual page properties are identical. The Web > Advanced pane includes the following automatic Page checkpoint options:

- ▶ **Create a checkpoint for each Web page while recording.** Instructs QuickTest to automatically add a Page checkpoint for each Web page navigated during the recording process.

Note: If you are testing a Web page with dynamic content, using automatic Page checkpoints may cause the test to fail as these checkpoints assume that the page content is static between record and run sessions.

All automatic Page checkpoints include the checks that you select from among the following options:

- ▶ **Broken links.** Displays the number of broken links contained in the page during the run session.

Note: If the **Broken links - check only links to current host** option is selected (see "Setting Web Testing Options" on page 52), this number includes only those broken links that are targeted to the current host.

- ▶ **HTML source.** Checks that the expected source code is identical to the source code during the run session.

- ▶ **HTML tags.** Checks that the expected HTML tags in the source code are identical to those in the run session.
- ▶ **Image source.** Checks that the expected source paths of the images are identical to the sources in the run session.
- ▶ **Links URL.** Checks that the expected URL addresses for the links are identical to the URL addresses in the source code during the run session.
- ▶ **Load time.** Checks that the expected time it takes for the page to load during the run session is less than or equal to the amount of time it took during the record session PLUS the amount of time specified in the **Add seconds to page load time** option (see "Setting Web Testing Options" on page 52).
- ▶ **Number of images.** Checks that the expected number of images is identical to the number displayed in the run session.
- ▶ **Number of links.** Checks that the expected number of links is identical to the number displayed in the run session.
- ▶ **Ignore automatic checkpoints while running tests.** Instructs QuickTest to ignore the automatically added Page checkpoints while running your test.

Record Settings

You can set preferences for recording Web objects. The Web > Advanced pane includes the following **Record** settings:

- ▶ **Enable Web support for Microsoft Windows Explorer.** When selected, QuickTest treats relevant objects in Microsoft Windows Explorer as Web objects. When cleared, QuickTest does not record events on Web pages displayed in Microsoft Windows Explorer.

Note: After modifying this setting, for the change to take effect, you must close all instances of Microsoft Windows Explorer (confirm that all **explorer.exe** processes are closed in the Windows Task Manager or restart the computer) and then restart QuickTest.

- ▶ **Record coordinates.** Records the actual coordinates relative to the object for each operation.
 - ▶ **Record MouseDown and MouseUp as Click.** Records a **Click** method for MouseUp and MouseDown events.
-

Note: For Web, QuickTest records **RightClick** and **MiddleClick** methods for most Web objects. Therefore, this option is relevant only for clicks made using the left mouse button.

- ▶ **Record Navigate for all navigation operations.** Records a Navigate statement each time a Frame URL changes.
 - ▶ **Use standard Windows mouse events.** Instructs QuickTest to use standard Windows mouse events instead of browser events for the following events:
 - ▶ **OnClick**
 - ▶ **OnMouseDown**
 - ▶ **OnMouseUp**
-

Note:

- ▶ Use this option only if the events are not properly recorded using browser events.
 - ▶ For Web, QuickTest records **RightClick** and **MiddleClick** methods for most Web objects. Therefore, this option is relevant only for clicks made using the left mouse button.
 - ▶ This option is available only for Internet Explorer.
-

If QuickTest does not record Web events in a way that matches your needs, you can also configure the events you want to record for each type of Web object. For example, if you want to record events, such as a mouseover that opens a sub-menu, you may need to modify your Web event configuration to recognize such events. For more information, see Chapter 39, "Configuring Web Event Recording for Web Objects."

Run Settings

You can set preferences for working with Web objects during a run session. The Web > Advanced pane includes the following **Run** settings:

- ▶ **Browser cleanup.** Closes all open browsers after the current run or iteration ends.

When this option is selected, all currently open browsers are closed when the current run or iteration ends, regardless of whether the browsers were opened before or after QuickTest was opened.
- ▶ **Run only click.** Determines whether a Click operation is run on the application by sending MouseDown, MouseUp, and Click events, or by sending only a Click event. This option is relevant only for Click operations when 'Mouse' is selected for the 'Replay type' option. It is relevant only for left-button mouse clicks.
- ▶ **Replay type.** Configures how to run mouse operations according to the selected option:
 - ▶ **Event.** Runs mouse operations using browser events.
 - ▶ **Mouse.** Runs mouse operations using the mouse, and keyboard operations using the keyboard.
- ▶ **Run using source index.** Learns and stores the source index value when learning Web test objects, and uses that value during a run session to increase performance capabilities. During the run session, QuickTest uses the learned source index value to return the DOM element from the application and verifies that this object matches the test object description. If it does not, the source index is ignored.

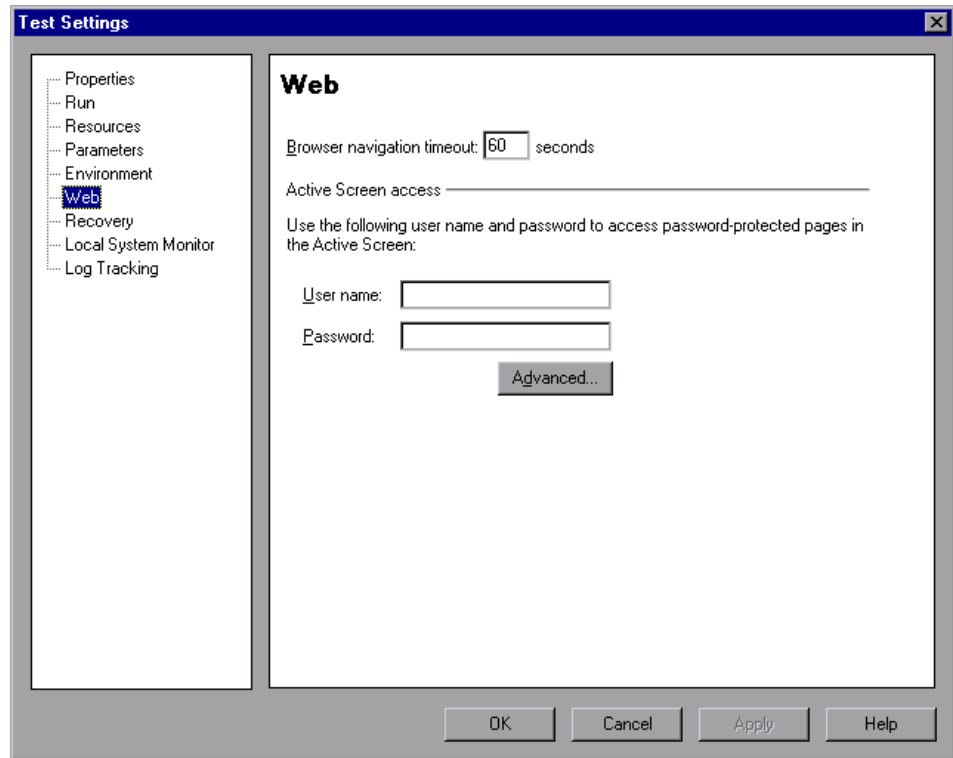
- ▶ **Resize browser on run if resized during a recording session.** If this option is selected and you resize the browser during a recording session, QuickTest resizes the browser to this size when a subsequent run session begins. At the end of a run session, the browser returns to its default size. It is recommended that you select this option if your test performs drag and drop operations.

Note: To use this option, select the **Open the following browser** option in the Record and Run Settings dialog box before recording. When this option is cleared, QuickTest does not change the browser size when a run session begins.

- ▶ **Learn and run using automatic XPath identifiers.** Generates and stores an XPath value when learning Web test objects, and uses that value during a run session to improve object identification reliability. During the run session, QuickTest uses the learned XPath value to return the DOM element from the application and verifies that this object matches the test object description. If it does not, the learned XPath is ignored. Additionally, if the description for a test object includes the XPath or CSS identifier, this option is ignored for that object. For details, see "Understanding Web Object Identifiers" on page 78.

Defining Web Settings for Your Test

The Web pane of the Test Settings dialog box (**File > Settings > Web** node, when a test is open) provides options for recording and running tests on Web sites. You can set how long to wait for browser navigations and you can specify the Active Screen access information to use with password-protected resources in the captured Active Screen page.



Note: The Web pane is available only if the Web Add-in is installed and loaded.

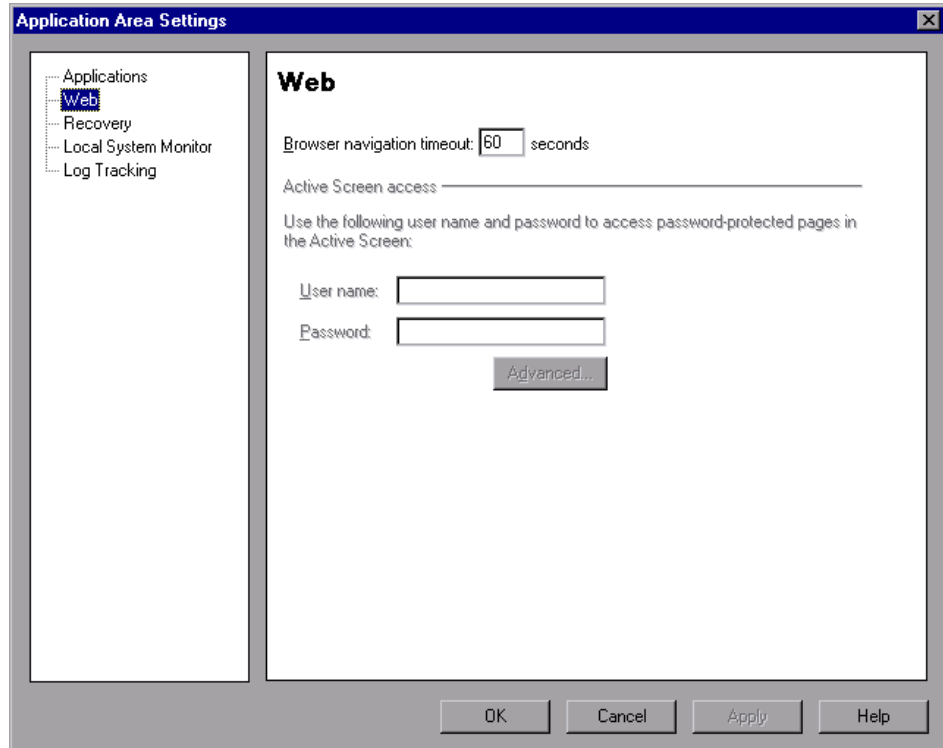
The Web pane includes the following options:

Option	Description
Browser navigation timeout	Sets the maximum time (in seconds) that QuickTest waits for a Web page to load before running a step in the test.
User name	The user name for password-protected resources that use a standard authentication mechanism. For more information, see "Using the Advanced Authentication Mechanism" on page 654.
Password	The password for password-protected resources that use a standard authentication mechanism. For more information, see "Using the Advanced Authentication Mechanism" on page 654.
Advanced	Opens the Advanced Authentication dialog box, which enables you to manually log in to your Web site to enable access to password-protected resources that use an advanced authentication mechanism. For more information, see "Using the Advanced Authentication Mechanism" on page 654.

Tip: In addition to the options in this pane, you can also configure the events you want to record for each type of Web object. For example, if you want to record events, such as moving the pointer over an object to open a sub-menu, you may need to modify your Web event configuration to recognize such events. For more information, see "Web Event Recording Configurations" on page 74.

Defining Web Settings for Your Application Area

The Web pane of the Application Area Settings dialog box (**File > Settings > Web** node, when an application area is open) provides an option that defines how long to wait for a Web page to load.



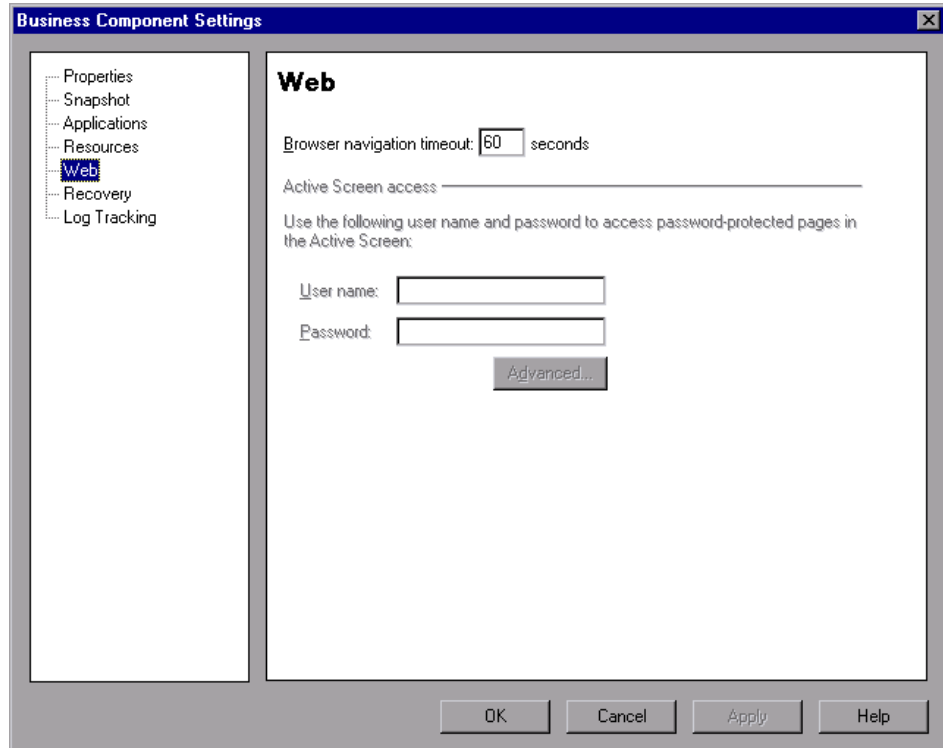
Note: The Web pane is available only if the Web Add-in is installed and loaded.

The Web pane includes the following options:

Option	Description
Browser navigation timeout	Sets the maximum time (in seconds) that QuickTest waits for a Web page to load before running a step in the test or component.
User name	This option is not relevant for components.
Password	This option is not relevant for components.
Advanced	This option is not relevant for components.

Viewing Web Settings for Your Business Component

The Web pane of the Business Component Settings dialog box (**File > Settings > Web** node, when a business component is open) displays the setting used when running components on an application.



You define the Web pane settings for a component in its associated application area settings. For more information, see "Defining Web Settings for Your Application Area" on page 71.

Note: The Web pane is available only if the Web Add-in is installed and loaded.

The Web pane includes the following items:

Setting	Description
Browser navigation timeout	Displays the maximum time (in seconds) that QuickTest waits for a Web page to load before running a step in the component.
User name	This option is not relevant for components.
Password	This option is not relevant for components.
Advanced	This option is not relevant for components.

Tip: In addition to the options in this tab, you can also configure the events you want to record for each type of Web object. For example, if you want to record events, such as moving the pointer over an object to open a sub-menu, you may need to modify your Web event configuration to recognize such events. For more information, see "Web Event Recording Configurations" on page 74.

Web Event Recording Configurations

When you record on a Web application, QuickTest generates steps by recording the events you perform on the Web objects in your application. An **event** is a notification that occurs in response to an operation, such as a change in state, or as a result of the user clicking the mouse or pressing a key while viewing the document.

QuickTest includes event recording configurations that have been optimized for each Web-based add-in, so that in most cases QuickTest records steps for relevant events on each object and avoids recording steps for events that usually do not impact the application. For example, by default, QuickTest records a step when a click event occurs on a link object, but does not record a step when a mouseover event occurs on a link.

Each Web-based add-in has its own XML file that defines the Web-event recording configuration for objects in that environment.

When you perform an operation on a Web-based object during a recording session (and the appropriate add-in is installed and loaded), QuickTest uses the recording configuration defined for that environment.

If your application contains several types of Web-based controls, the appropriate Web event recording configuration is used for each object and the configuration for one environment does not override another.

Customizing Web Event Recording Configurations

You can view and customize the configuration settings for the Web Add-in in the Web Event Recording Configuration dialog box. The settings in that dialog box affect the recording behavior only for objects that QuickTest recognizes as Web test objects.

Note: For the purposes of Web event recording, QuickTest treats Web test objects that are child objects of a PSFrame test object as PeopleSoft objects and thus applies the settings in the PeopleSoft event configuration XML file when recording those objects.

For more information, see Chapter 39, "Configuring Web Event Recording for Web Objects."

In most cases, it is not necessary to customize the Web event recording configuration of other add-ins. If you do need to customize these settings, you can do so either by editing the XML for the relevant add-in manually, or you can import the XML into the Web Event Recording Configuration dialog box to make the necessary changes and then export the modified file.

To modify the Web event recording configuration XML file manually:

- 1** In a text or XML editor, open the appropriate **MyEnvEventConfiguration.xml** file from the **<QuickTest installation folder>\dat** folder, according to the following table:

Object Type:	XML File Name
.NET Web Forms	WebFormsEventConfiguration.xml
Siebel 7.5 or earlier	SiebelEventConfiguration.xml
Siebel 7.7 or later	CASEventConfiguration.xml
PeopleSoft Frame objects and all Web objects that are children of a PeopleSoft frame object	PSEventConfiguration.xml

- 2** Edit the file as necessary.
- 3** Save the file.

To modify the Web event recording configuration in the Web Event Recording Configuration dialog box:

- 1** Backup the event recording configuration for the Web environment:
 - a** Select **Tools > Web Event Recording Configuration**. The Web Event Recording Configuration dialog box opens.
 - b** Click **Custom Settings**.
 - c** Select **File > Save Configuration As** and specify an XML filename for the backup file.
- 2** Back up the event recording configuration for the environment you want to modify:

Create a copy of the relevant **<MyEnv>EventConfiguration.xml** file from the **<QuickTest installation folder>\dat** folder.

- 3** Modify the `<MyEnv>EventConfiguration.xml` file in the Web Event Recording Configuration dialog box:
 - a** In the Web Event Recording Configuration dialog box, select **File > Load Configuration** and browse to the relevant `<QuickTest installation folder>\dat\<MyEnv>EventConfiguration.xml` file. The event configuration for the selected environment is displayed in the dialog box.
 - b** Modify the configuration using the Web Event Recording Configuration dialog box options, as described in Chapter 39, "Configuring Web Event Recording for Web Objects."
 - c** Select **File > Save Configuration As** and overwrite the previous `<QuickTest installation folder>\dat\<MyEnv>EventConfiguration.xml` file.
- 4** Restore the configuration file for the Web environment:

Select **File > Load Configuration** and browse to the backup copy of the Web configuration file that you saved in step 1.

Caution: QuickTest always applies the configuration that is loaded in the Web Event Recording Configuration dialog box to all Web objects. If you do not restore the Web configuration file as described in step 4, then QuickTest will apply the configuration for the file you loaded in step 3, and as a result, QuickTest may not record Web events properly.

Understanding Web Object Identifiers

During a run session, QuickTest attempts to identify each object in your application by matching the description properties stored for the corresponding test object with the properties of the DOM element in the application. For complex Web applications that contain many objects, using only the standard identification methods may have unreliable results. For more information on the standard methods QuickTest uses to identify objects, see the section on how QuickTest identifies objects in the *HP QuickTest Professional User Guide*.

You can instruct QuickTest to use Web object identifiers before the regular object identification process to help limit the number of candidate objects to identify. QuickTest accesses the application's DOM and returns objects that match the object identifier property values. QuickTest then continues to identify this smaller set of returned objects using the normal object identification process. Therefore, using Web object identifiers can lead to a more reliable and accurate object identification, and a quicker object identification process.

For more information about the general workflow of the object identification process, see the section on object identification in the *HP QuickTest Professional User Guide*.

This section includes:

- "Web Object Identifier Types" on page 79
- "Considerations for Working with Web Object Identifiers" on page 80
- "How to Use Web Object Identifiers - Exercise" on page 81

Web Object Identifier Types

The following Web object identifiers are available:

CSS

CSS (Cascading Style Sheet) is a language used to define formatting of elements in HTML pages. You can define a CSS identification property value for a test object to help identify a Web object in your application based on its CSS definition.

QuickTest uses CSS identifiers only when identifying objects and not when learning objects. Therefore, they are not available from the Object Spy or the Object Identification dialog box.

For usage examples, see "How to Use Web Object Identifiers - Exercise" on page 81.

User-defined XPath

XPath (XML Path) is a language used to define the structure of elements in XML documents. You can define an XPath identification property to help identify a Web object in your application based on its location in the hierarchy of elements in the Web page. Because of the flexible nature of the language, you can define the XPath according to the unique way your Web page is structured.

QuickTest uses XPath identifiers only when identifying objects and not when learning objects. Therefore, they are not available from the Object Spy or the Object Identification dialog box.

For usage examples, see "How to Use Web Object Identifiers - Exercise" on page 81.

Automatic XPath

You can instruct QuickTest to automatically generate and store an XPath value when learning Web test objects. During the run session, if the automatically learned XPath for a particular object results in multiple matches or no matches, the learned XPath is ignored. Additionally, if you have added a user-defined XPath or CSS identification property to a test object description, then the automatically learned XPath is ignored.

Automatic XPath is a QuickTest-generated property, and therefore it is not available from the Object Spy, the Add/Remove Properties dialog box, or the Object Identification dialog box.

You enable this option in the Web section of the Options dialog box. For details, see "Setting Web Testing Options" on page 52.

Attribute/* Notation

You can use the **attribute/*** notation to access custom native properties of Web-based objects or events associated with Web-based objects. For details defining and using this option, see "Accessing Custom Properties of Web-Based Objects" on page 85.

For considerations on working with Web object identifiers, see "Considerations for Working with Web Object Identifiers" on page 80.

Considerations for Working with Web Object Identifiers

Consider the following when using Web object identifiers:

General

- ▶ Defining **xpath** and **css** properties using Frame HTML tags is not supported. This may cause incorrect identification when identifying **Frame** objects or retrieving **Frame** objects using the **ChildObjects** method.
- ▶ **xpath** and **css** properties are not supported for .NET Web Forms test objects or for other Web-based test objects that have .NET Web Forms parent test objects.
- ▶ 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.

Differences Between User-defined XPath and Automatic XPath Behavior During Run Sessions

Behavior in case of...	User-defined XPath	Automatic XPath
Multiple objects match the XPath value	QuickTest continues to identify the matching objects.	QuickTest ignores the learned XPath and continues with the regular object identification process.
No objects match the XPath value	Object identification fails, and QuickTest continues to identify the object using Smart Identification	QuickTest ignores the learned XPath and continues with the regular object identification process.

How to Use Web Object Identifiers - Exercise

In this exercise, you use XPath and CSS identifiers in a test object description to help locate the correct button in an HTML table.

This exercise includes the following steps:

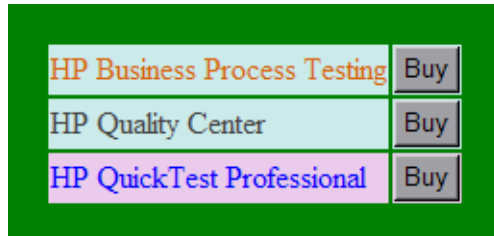
- "Prerequisites" on page 82
- "Create a sample Web application" on page 82
- "Learn the button objects in the Web application" on page 82
- "Remove the ordinal identifiers from the button objects" on page 83
- "Add a CSS identifier based on the object's parent-container" on page 83
- "Add an XPath identifier based on the object's parent-container" on page 84
- "Add an XPath identifier based on the object's sibling-element" on page 84
- "Results" on page 84

1 Prerequisites

- a Open QuickTest and create a new test.
- b Disable Smart Identification for the **Button** test object class by selecting **Tools > Object Identification**, selecting the Web environment in the Object Identification dialog box, and then selecting the **Button** test object class from the **Test Object classes** list.
- c Disable automatic XPath by selecting **Tools > Options > Web > Advanced**, and then making sure that the **Learn and run using automatic XPath identifiers** checkbox is not selected.

2 Create a sample Web application

- a Open the Help version of this exercise, copy the syntax content into a text document, and save the document with an **.html** extension. The document is saved as an HTML page.
- b Review the appearance and content of your newly created HTML page in any browser. Make sure that it matches the following image.



3 Learn the button objects in the Web application

- a In QuickTest, open the Object Repository Manager, and select **Object > Navigate and Learn**. QuickTest is hidden, and the cursor changes to a pointing hand.
- b To verify that QuickTest learned the objects correctly, in the object repository, select each **Button** object and select **View > Highlight in Application**. QuickTest highlights each button object in the HTML page.

- c** Rename the **Button** objects to make them more clear:
 - Rename **Buy** to **Buy_BPT**.
 - Rename **Buy_2** to **Buy_QC**.
 - Rename **Buy_3** to **Buy_QTP**.

4 Remove the ordinal identifiers from the button objects

Because all of the **Button** objects have identical property values, when QuickTest learned the objects it assigned an ordinal identifier to each test object based on the location of each object in the application. This may cause QuickTest to identify the objects incorrectly if the sorting order of the buttons in the application changes.

- a** Select the first button object to display its object properties on the right side of the object repository window.
- b** In the **Ordinal Identifier** section, select the **Browse** button. The Ordinal Identifier dialog box opens.
- c** In the **Identifier type** drop-down list, select **None** and close the dialog box. The ordinal identifier is removed from the test object's identification properties.
- d** Repeat steps a- c above for each of the buttons.
- e** Verify that the test object descriptions are no longer unique by selecting each test object and selecting **View > Highlight in Application**. QuickTest cannot identify the objects.

5 Add a CSS identifier based on the object's parent-container

- a** Select the **Buy_BPT** button. The test object details are displayed on the right side of the object repository window.
- b** In the **Object Description** section, click the **Add** button, and add the **css** property to the test object description.
- c** Copy and paste the following syntax into the Value edit box:

```
tr.BPTRow input
```

6 Add an XPath identifier based on the object's parent-container

- a Select the **Buy_QTP** button. The test object details are displayed on the right side of the object repository window.
- b In the **Object Description** section, click the **Add** button, and add the **xpath** property to the test object description.
- c Copy and paste the following syntax into the Value edit box:

```
//TR[@id='QTP']/*/INPUT
```

7 Add an XPath identifier based on the object's sibling-element

- a Select the **Buy_QC** button. The test object details are displayed on the right side of the object repository window.
- b In the **Object Description** section, click the **Add** button, and add the **xpath** property to the test object description.
- c Copy and paste the following syntax into the **Value** edit box:

```
//td[contains(concat(' ',text(),''),'Quality')]/../INPUT
```

8 Results

Select each object and select **View > Highlight in Application**. QuickTest can now identify each button based on the Web object identifiers you added.

Accessing Custom Properties of Web-Based Objects

You can use the **attribute/*** notation to access custom native properties of Web-based objects or events associated with Web-based objects. You can then use these properties or events to identify such objects by adding the notation to the object's description properties using the Object Identification dialog box, or by using programmatic descriptions.

Example of using **attribute/<property>** to identify a Web object

Suppose a Web page has the same company logo image in two places on the page:

```
<IMG src="logo.gif" LogoID="122">
<IMG src="logo.gif" LogoID="123">
```

You could identify the image that you want to click by adding the **attribute/LogoID** notation to the object's description properties and using a programmatic description to identify the object:

```
Browser("Mercury Tours").Page("Find Flights").Image("src:=logo.gif",
    "attribute/LogoID:=123").Click 68, 12
```

Example of using **attribute/<event>** to identify a Web object

Suppose a Web page has an object with an **onclick** event attached to it:

```
"alert('OnClick event for edit.');"

```

You can identify the object by adding the **attribute/onclick** notation to the object's description properties and using a programmatic description to identify the object:

```
Browser("Simple controls").Page("Simple controls").WebEdit("attribute/onclick:=
alert('OnClick event for edit.\.');"").Set "EditText"
```

For more information on the Object Identification dialog box and programmatic descriptions, see the *HP QuickTest Professional User Guide*.

3

Testing Windows-Based Applications

This chapter describes how to set QuickTest record and run options for testing Windows-based applications, and how to define record and run variables for a Windows Environment. It also describes how you can specify the Windows-based applications on which the components associated with this application area can record and run.

This chapter includes:

- About Testing Windows-Based Applications on page 87
- Setting Windows Applications Record and Run Options on page 89
- Defining Record and Run Variables for a Windows-Based Environment on page 101
- Setting Windows Application Testing Options on page 103
- Setting Advanced Windows Applications Options on page 107

About Testing Windows-Based Applications

QuickTest provides a number of add-ins for testing Windows-based applications. The way you configure many of your QuickTest settings is the same or similar for most QuickTest Windows-based add-ins (as well as for the built-in standard Windows testing support). These common configuration options are described in the remainder of this chapter.

For more information about standard Windows testing support, see Chapter 30, "Using Standard Windows Testing Support."

For additional details on how to work with Windows-based add-ins, see the specific sections describing these add-ins in the guide:

- "The ActiveX Add-in" on page 165
- "The Delphi Add-in" on page 175
- "The .NET Add-in" on page 117
- "The PowerBuilder Add-in" on page 281
- "The Add-in for SAP Solutions" on page 287
- "The Stingray Add-in" on page 501
- "The Terminal Emulator Add-in" on page 531
- "The Visual Basic Add-in" on page 607
- "The VisualAge Smalltalk Add-in" on page 613

Tip: When recording tests or scripted components on 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.

This option is not relevant for business components.

For more information, see the section describing the Active Screen pane in the *HP QuickTest Professional User Guide*.

Setting Windows Applications Record and Run Options

You use the Windows Applications tab of the Record and Run Settings dialog box to set options that affect how you start creating and running tests for Windows-based applications. These options instruct QuickTest which applications to open when you begin to record or run your test. For Windows applications, you also specify the applications on which you want to record. Note that you can instruct QuickTest to open and record on applications from more than one environment.

This section includes:

- "The Record and Run Settings Dialog Box: Windows Applications Tab" on page 90
- "The Application Details Dialog Box" on page 95
- "Record and Run Setting Guidelines for Windows-Based Add-ins" on page 99

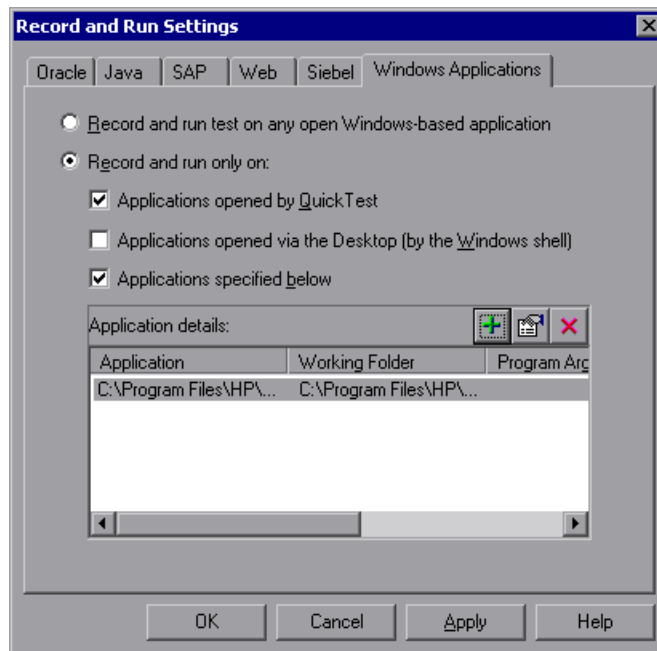
Note: The Record and Run Settings dialog box applies only to tests. Record settings for components are specified in the Applications pane or Applications dialog box of the relevant application area. However, specific record and run settings do not need to be defined for components. For more information on the Applications pane and the Applications dialog box, see the *HP QuickTest Professional for Business Process Testing User Guide*.

The Record and Run Settings Dialog Box: Windows Applications Tab

<p>Description</p>	<p>Enables you to define preferences for recording and running tests on Windows-based applications.</p> <p>For some Windows-based add-ins, the settings in this dialog box may also affect the applications that QuickTest recognizes for other QuickTest operations, such as learning objects or using the Object Spy. For more information, see "Record and Run Setting Guidelines for Windows-Based Add-ins" on page 99.</p>
<p>How to Access</p>	<p>Do one of the following:</p> <ul style="list-style-type: none"> ▶ Automation menu > Record and Run Settings dialog box > Windows Applications tab ▶ The Record and Run Settings dialog box also opens automatically each time you begin recording a new test (unless you open the dialog box and set your preferences manually before you begin recording).
<p>Important Information</p>	<p>For performance reasons, the default setting in the Windows Applications tab is set 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.</p> <p>If you loaded other add-ins when you opened QuickTest, there may be additional tabs in the Record and Run Settings dialog box. If this is the case, confirm that the option to record and run on any open application (upper radio button of each tab) is selected.</p> <p>While this setting does not directly affect your record or run sessions, it prevents QuickTest from opening unnecessary applications when you begin record or run sessions.</p>

<p>Learn More</p>	<p>Conceptual overview:</p> <ul style="list-style-type: none"> ▶ "The Record and Run Settings Dialog Box: Overview" on page 37 ▶ "About Testing Windows-Based Applications" on page 87 ▶ "Setting Windows Applications Record and Run Options" on page 89 <p>Additional related topics: "Additional References" on page 94</p>
--------------------------	---




Below is an image of the Windows Applications tab:



Windows Applications Tab Options

Option	Description
Record and run test on any open Windows-based application	<p>Instructs QuickTest to record all operations performed on any Windows-based application that is opened while recording your test (including e-mail applications, file management applications, and so on). QuickTest records and runs only on applications that have a user interface, and it does not matter how the applications are opened (as child processes of Windows Explorer, child processes of QuickTest, and so on).</p> <p>When selecting this option, make sure that all the applications on which you want to record are currently closed. For some environments, QuickTest can recognize and/or record on the applications that you open manually only after you select this option and click OK. Instances of these applications that are already open when the Record and Run Settings dialog box opens may be ignored or may not be recognized or recorded correctly.</p>

Option	Description
<p>Record and run only on</p>	<p>Restricts record and run (and in some cases pointing hand) operations to selected applications. Additionally, you can configure whether QuickTest should open these applications for you at the beginning of a record or run session. The following options are available:</p> <ul style="list-style-type: none"> ▶ Applications opened by QuickTest. This option records, recognizes, and runs only on applications invoked by QuickTest (as child processes of QuickTest). For example, applications opened during a record or run session using a SystemUtil.Run statement, or using a statement such as <code>Set shell = createobject("wscript.shell")</code> <code>shell.run "notepad".</code> ▶ Applications opened via the Desktop (by the Windows shell). This option records, recognizes, and runs only on applications that are opened via the Windows Desktop. For example, applications opened from the Windows Start menu, by double-clicking executable files in the Windows Explorer, by double-clicking a shortcut on the Windows Desktop, or by clicking icons on the Quick Launch bar. ▶ Applications specified below. This option records, recognizes, and runs only on applications listed in the Application details area. <p>Tips:</p> <ul style="list-style-type: none"> ▶ If you do not want to record on any Windows-based applications, select only the Applications specified below check box, and ensure that there are no applications listed in the Application details area. ▶ Make sure that all the applications listed in the Application details area are currently closed. For some environments, QuickTest can record only on the instances of the specified applications that are opened after you select this option and click OK. Instances of these applications that are already open when the Record and Run Settings dialog box opens may be ignored or may not be recognized or recorded correctly.

Option	Description
Application details	Lists the details of the applications on which to record and run the test. For more information on the details displayed, see "The Application Details Dialog Box" on page 95.
	Opens the Application Details dialog box to enable you to add an application to the application list. You can add up to ten applications. For more information, see "The Application Details Dialog Box" on page 95.
	Opens the Application Details dialog box to enable you to edit the application details for the selected application. For more information, see "The Application Details Dialog Box" on page 95.
	Removes the selected application from the application list.

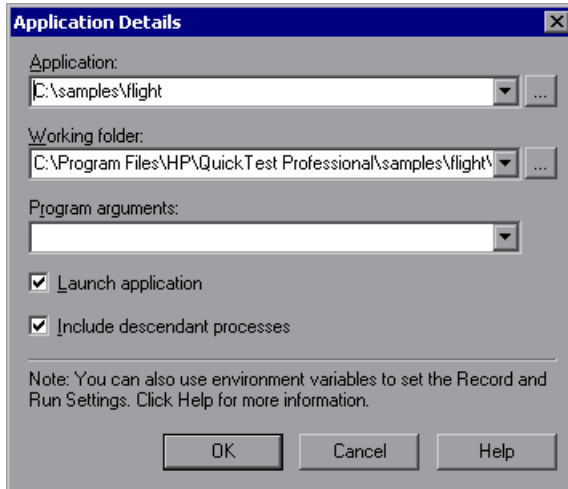
Additional References

Related User Interface Topics	<ul style="list-style-type: none"> ➤ "The Application Details Dialog Box" on page 95 ➤ "Setting Windows Application Testing Options" on page 103 ➤ "Setting Advanced Windows Applications Options" on page 107
Related Tasks	<ul style="list-style-type: none"> ➤ "Using the Record and Run Settings Dialog Box" on page 38 ➤ "Defining Record and Run Variables for a Windows-Based Environment" on page 101
Related Concepts	<ul style="list-style-type: none"> ➤ "Record and Run Setting Guidelines for Windows-Based Add-ins" on page 99 ➤ "Using Environment Variables to Specify the Record and Run Details for Your Test" on page 40

The Application Details Dialog Box

Description	Enables you to edit the application details for the selected application.
How to Access	<ul style="list-style-type: none"> ▶ Automation menu > Record and Run Settings dialog box > Windows Applications tab. Click the Add or Edit button. <p>Note: The Record and Run Settings dialog box also opens automatically each time you begin recording a new test (unless you open the dialog box and set your preferences manually before you begin recording).</p>
Important Information	<ul style="list-style-type: none"> ▶ You can add up to ten applications to the application list displayed in the Windows Applications tab, and you can edit an existing application in the list. You can also select whether to launch the selected applications when the session starts, and whether to record and run on the application's descendant processes. ▶ The details entered in the Application Details dialog box are displayed as a single list item for each application in the Application details area of the Windows Applications tab.
Learn More	<p>Conceptual overview:</p> <ul style="list-style-type: none"> ▶ "The Record and Run Settings Dialog Box: Overview" on page 37 ▶ "About Testing Windows-Based Applications" on page 87 <p>Additional related topics: "Additional References" on page 98</p>

Below is an image of the Application Details dialog box:



Application Details Dialog Box Options

Option	Description
Application	<p>Instructs QuickTest to record and run on the specified executable file.</p> <p>You can enter the executable file as a relative path. During the run session, QuickTest searches for the file in the folder for the current test, and then in the folders listed in the Folders pane of the Options dialog box. For more information, see the sections on setting folder testing options, and on using relative paths in the <i>HP QuickTest Professional User Guide</i>.</p> <p>Notes:</p> <ul style="list-style-type: none"> ▶ The Application box should contain only the file name and path for the application. If you want to add command line arguments, use the Program arguments box. ▶ The full path name is used to launch an application only when Launch application is selected. QuickTest records and runs on any application with the specified executable file name. For example, if you specify C:\Windows\notepad.exe, QuickTest records on a Notepad application invoked from any folder. <p>Tip: You can specify a document or other file associated in the file system with an application, for example, c:\tmp\1.txt. In this case, QuickTest automatically opens the specified file in the associated application (Notepad in this example). If you use this option, QuickTest ignores any defined program arguments.</p>
Working folder	<p>Optional. Specifies the current working folder for the application. The current working folder is used by the application to search for related files. If a working folder is not specified, the executable folder is used as the working folder.</p> <p>Note: This parameter is used only when Launch application is selected. If Launch application is not selected, its value has no effect.</p>
Program arguments	<p>Optional. Instructs QuickTest to open the application using the specified command line arguments.</p> <p>Note: This parameter is used only when Launch application is selected. If Launch application is not selected, its value has no effect.</p>

Option	Description
Launch application	Instructs QuickTest whether to launch the selected application when the record and run session begins. By default, this option is selected.
Include descendant processes	Instructs QuickTest whether to record and run on processes created by the specified application during the record and run session. For example, a process that is used only as a launcher may create another process that actually provides the application functionality. This descendant process must therefore be included when recording or running tests on this application, otherwise the functionality will not be recorded, or the run session will fail. By default, this option is selected.

Additional References

Related User Interface Topics	<ul style="list-style-type: none"> ▶ "The Record and Run Settings Dialog Box: Windows Applications Tab" on page 90 ▶ "The Options Dialog Box: Windows Applications Pane" on page 103 ▶ "The Windows Applications > Advanced Pane" on page 107
Related Tasks	"Defining Record and Run Variables for a Windows-Based Environment" on page 101
Related Concepts	"Setting Windows Applications Record and Run Options" on page 89

Record and Run Setting Guidelines for Windows-Based Add-ins

Many QuickTest add-ins rely on the settings in the Windows Applications tab of the Record and Run Settings dialog box to determine on which applications QuickTest records and runs. For some add-ins, the settings in the Windows Applications tab of the Record and Run Settings dialog box may also affect the applications that QuickTest recognizes for certain operations while in edit mode, such as using the Object Spy or other pointing hand operations.

There may also be additional issues that you need to address to ensure that QuickTest recognizes your objects properly during record, run, and/or pointing hand operations.

These special considerations are detailed below for each QuickTest add-in that is affected by the settings in the Windows Applications tab of the Record and Run Settings dialog box.

Add-in Environment	Considerations and Guidelines
ActiveX	<ul style="list-style-type: none"> ▶ If you select the Record and Run only on radio button, the settings also define and limit which applications are recognized by the Object Spy and other pointing hand operations. ▶ QuickTest recognizes ActiveX objects only in applications that are opened after changing the settings in the Windows Applications tab of the Record and Run Settings dialog box.
Delphi	<ul style="list-style-type: none"> ▶ QuickTest recognizes only Delphi applications that have been precompiled with the Delphi agent module (MicDelphiAgent.pas). For more information, see "Enabling Communications Between QuickTest Professional and Your Delphi Application" on page 180. ▶ In some cases, if you select the Record and Run only on radio button, the settings may also define and limit which applications are recognized by the Object Spy and other pointing hand operations.

Add-in Environment	Considerations and Guidelines
.NET Windows Forms	If you select the Record and Run only on radio button, the settings also define and limit the applications that are recognized by the .NET Windows Forms Spy, the Object Spy, and other pointing hand operations.
PowerBuilder	If you select the Record and Run only on radio button, the settings also define and limit the applications that are recognized by the Object Spy and other pointing hand operations.
Standard Windows	<ul style="list-style-type: none"> ▶ The Record and Run only on radio button applies only to record and run sessions. QuickTest recognizes all standard Windows objects for Object Spy and pointing hand operations, regardless of the settings in the Record and Run Settings dialog box. ▶ It is recommended that applications are opened after changing the settings in the Windows Applications tab of the Record and Run Settings dialog box.
Stingray	<ul style="list-style-type: none"> ▶ In addition to the settings in the Record and Run Settings dialog box, you must also configure QuickTest to recognize your Stingray applications in the Stingray pane of the Options dialog box. For more information, see "Configuring Stingray Options" on page 521. ▶ If you select the Record and Run only on radio button, the settings also define and limit the applications that are recognized by the Object Spy and other pointing hand operations.
Terminal Emulators	<ul style="list-style-type: none"> ▶ QuickTest recognizes only the terminal emulator set in the Terminal Emulator pane of the Options dialog box. For more information, see "Modifying Your Terminal Emulator Settings" on page 562. ▶ The Record and Run only on radio button does not affect the applications on which QuickTest records, recognizes, and runs.

Add-in Environment	Considerations and Guidelines
Visual Basic	<ul style="list-style-type: none"> ▶ If you select the Record and Run only on radio button, the settings may also define and limit the applications that are recognized by the Object Spy and other pointing hand operations. ▶ QuickTest recognizes Visual Basic objects only in applications that are opened after changing the settings in the Windows Applications tab of the Record and Run Settings dialog box.
VisualAge	<ul style="list-style-type: none"> ▶ QuickTest can recognize only VisualAge applications that have been precompiled with the VisualAge Smalltalk agent (qt-adapter). For more information, see "Configuring the VisualAge Smalltalk Add-in" on page 618. ▶ The Record and Run only on radio button applies only to record and run sessions. QuickTest recognizes all VisualAge objects for Object Spy and pointing hand operations, regardless of the settings in the Record and Run Settings dialog box.

Defining Record and Run Variables for a Windows-Based Environment

You can use predefined environment variables to specify the applications or browsers you want to use for your test. This can be useful if you want to test how your application works in different environments.

Note: For more information on environment variables and how to use them in tests, see "Using Environment Variables to Specify the Record and Run Details for Your Test" on page 40.

To use environment variables to define the details for Windows applications on which you want to record and run tests, you must use the appropriate variable names as specified below:

Option	Variable Names	Description
Application	EXE_ENV_1 ... EXE_ENV_10	The executable files on which QuickTest records operations when record and run sessions begin. You can specify up to ten executable files.
Working folder	DIR_ENV_1 ... DIR_ENV_10	The folder to which the corresponding executable file refers (for each corresponding application).
Program arguments	ARGS_ENV_1 ... ARGS_ENV_10	The command line arguments to be used for the specified application (for each corresponding application).
Launch application	LNCH_ENV_1 ... LNCH_ENV_10	Whether to open the application when starting the record and run session (for each corresponding application). Possible values: 0 (do not launch the application) 1 (launch the application)
Include descendant processes	CHLD_ENV_1 ... CHLD_ENV_10	Whether to record and run on processes created by the application during the record and run session (for each corresponding application). Possible values: 0 (do not record on descendant processes) 1 (record descendant processes)

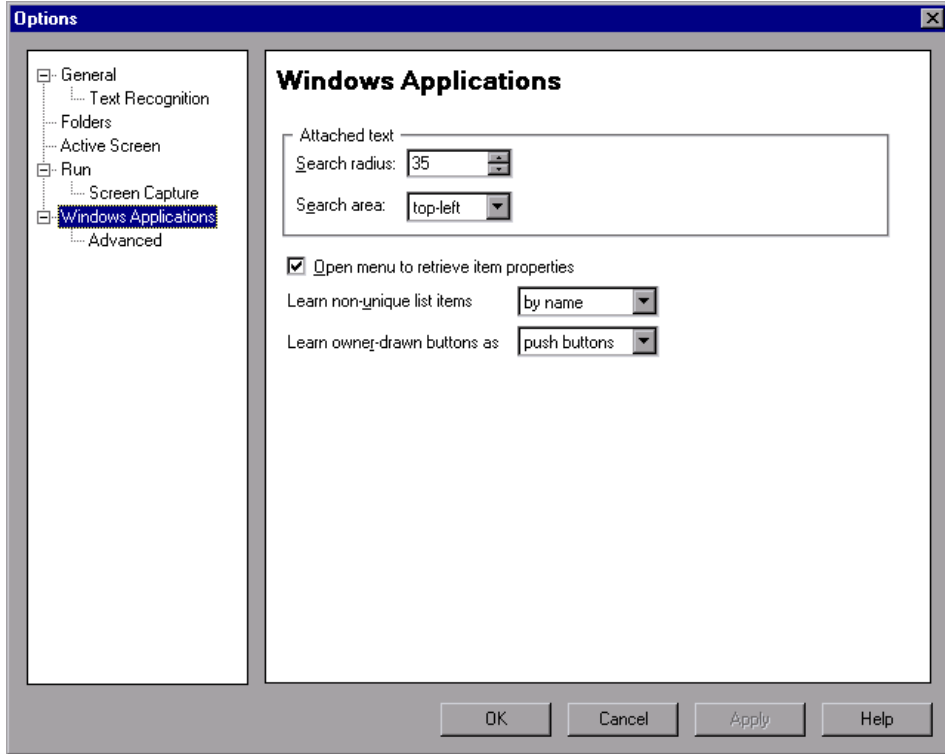
Setting Windows Application Testing Options

The Windows Applications pane options enable you to configure how QuickTest records and runs tests and components for Windows-based applications, such as standard Windows, ActiveX, .NET Windows Forms, WPF, SAP GUI for Windows, and Visual Basic applications.

The Options Dialog Box: Windows Applications Pane

Description	Enables you to configure how QuickTest records and runs tests and components for Windows-based applications.
How to Access	Tools menu > Options dialog box > Windows Applications node
Learn More	Conceptual overview: "About Testing Windows-Based Applications" on page 87 Additional related topics: "Additional References" on page 106

Below is an image of the Windows Applications pane:



Options Dialog Box: Windows Applications Pane Options

Option	Description
Attached text	<p>Enables you to specify the search criteria that QuickTest uses to retrieve an object's attached text. An object's attached text is the closest static text within a specified radius from a specified point. The retrieved attached text is saved in the object's corresponding text or attached text identification property.</p> <p>Note: Sometimes the static text that you believe to be closest to an object is not actually the closest static text. You may need to use trial and error to make sure that the attached text is the static text object of your choice.</p> <p>Search radius. Indicates the maximum distance, in pixels, that QuickTest searches for attached text.</p> <p>Search area. Indicates the point on an object from which QuickTest searches for the object's attached text.</p> <p>Select an option from the Search area list:</p> <ul style="list-style-type: none"> ➤ Top-Left. Top-left corner ➤ Top. Midpoint between the two top corners ➤ Top-Right. Top-right corner ➤ Right. Midpoint between the two right corners ➤ Bottom-Right. Bottom-right corner ➤ Bottom. Midpoint between the two bottom corners ➤ Bottom-Left. Bottom-left corner ➤ Left. Midpoint between the two left corners
Open menu to retrieve item properties	<p>Instructs QuickTest to open menu objects before retrieving menu item properties during a run session.</p> <p>Notes:</p> <ul style="list-style-type: none"> ➤ Selecting this option may slow the run, but it can be useful if menu item properties change upon opening the menu. ➤ This option, which is selected by default, sets the default behavior for all menu objects. You can use the <code>ExpandMenu</code> property in a test or function library to set this behavior for a specified menu object. For more information, see the <i>HP QuickTest Professional Object Model Reference</i>.

Option	Description
Record non-unique list items	<p>Determines what QuickTest records when more than one item (in a list or tree) has an identical name.</p> <ul style="list-style-type: none"> ▶ by name. Records the item's name. During a run session, QuickTest finds and selects the first instance of the name, regardless of the item chosen during recording. Select this option if the all items with the same name have the identical behavior. ▶ by index. Records the item's index number. Select this option if items with the same name do not necessarily have the identical behavior.
Record owner-drawn buttons as	<p>Instructs QuickTest how to identify and learn custom-made buttons in the application.</p> <p>Select an option from the list:</p> <ul style="list-style-type: none"> ▶ push buttons ▶ check boxes ▶ radio buttons ▶ objects <p>Note: If you select objects, QuickTest learns each owner-drawn button as a WinObject. (When working with tests, QuickTest can also learn an owner-drawn button as a virtual object, if you define the virtual object. For more information, see the section on learning virtual objects in the <i>HP QuickTest Professional User Guide</i>.)</p>

Additional References

Related User Interface Topics	<ul style="list-style-type: none"> ▶ "The Windows Applications > Advanced Pane" on page 107 ▶ "The Record and Run Settings Dialog Box: Windows Applications Tab" on page 90
--------------------------------------	--

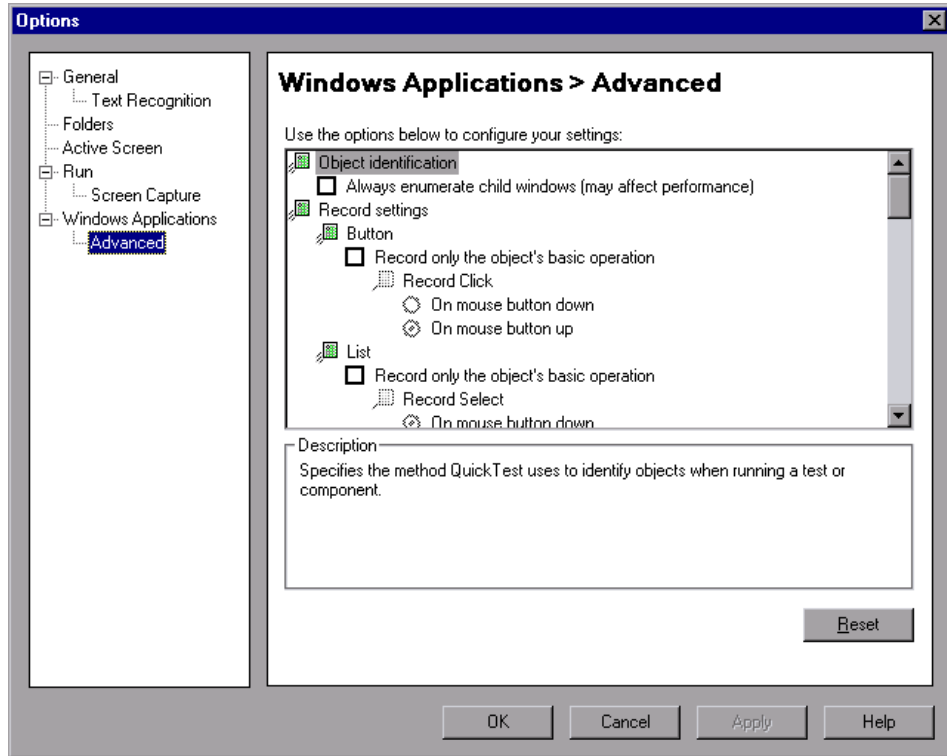
Setting Advanced Windows Applications Options

The Windows Applications > Advanced pane enables you to modify how QuickTest records and runs tests or components on Windows-based applications, such as ActiveX or Visual Basic. You can click the **Reset** button at any time to reset all options to their default settings.

The Windows Applications > Advanced Pane

Description	Enables you to modify how QuickTest records and runs tests or components on Windows-based applications.
How to Access	Tools menu > Options item > Windows Applications node > Advanced node
Learn More	Conceptual overview: "About Testing Windows-Based Applications" on page 87 Additional related topics: "Advanced Information" on page 114

Below is an image of the Windows Applications > Advanced pane in the Options dialog box:



Windows Applications > Advanced Pane Options

The Windows Applications > Advanced pane contains the following groups of options:

- ▶ Object Identification Options (described on page 109)
- ▶ Record Settings Options (described on page 109)
- ▶ Run Settings Options (described on page 114)

For additional information on some of the options, see "Advanced Information" on page 114.

Object Identification Options

You can specify the method QuickTest uses to identify objects when running a test or component. The Windows Applications > Advanced pane includes the following **Object identification** options:

Option	Description
Always enumerate child windows (may affect performance)	Instructs QuickTest to enumerate all child windows when recording and running a test or component. This option is cleared by default and should be used only when an object cannot otherwise be identified, because it may significantly affect performance. For more information, see "Advanced Information" on page 114.

Record Settings Options

You can specify how QuickTest treats certain objects when recording a test or component. The Windows Applications > Advanced pane includes the following **Record settings** options:

Category	Option
Button	<p>Defines record settings for button objects:</p> <ul style="list-style-type: none"> ▶ Record only the object's basic operation. Enables simplified recording on the button. Using this mode may improve recognition of user operations in non-standard cases. This option is cleared by default and should be used only when the default recording method does not meet your needs. For more information, see "Advanced Information" on page 114. ▶ Record Click. Specifies whether the Click operation should be recorded when the mouse button is pressed (On mouse button down) or when the mouse button is released (On mouse button up). This option is enabled only when Record only the object's basic operation is selected. Default = On mouse button up.

Category	Option
<p>List</p>	<p>Defines record settings for Windows-based list objects (for example, WinList, WinListView, and VbList):</p> <ul style="list-style-type: none"> ➤ Record only the object's basic operation. Enables simplified recording on the list. Using this mode may improve recognition of user operations in non-standard cases. This option is cleared by default and should be used only when the default recording method does not meet your needs. For more information, see "Advanced Information" on page 114. ➤ Record Select. Specifies whether the Select operation should be recorded when the mouse button is pressed (On mouse button down) or when the mouse button is released (On mouse button up). This option is enabled only when Record only the object's basic operation is selected. Default = On mouse button up.
<p>Menu</p>	<p>Defines record settings for menu objects:</p> <ul style="list-style-type: none"> ➤ Enable recording. Specifies whether QuickTest records operations on menu controls. For example, you may want QuickTest to ignore the actual process of selecting a menu to open another window. This option is selected by default. ➤ Menu recording mode. Specifies whether QuickTest verifies or ignores menu initialization events before recording operations on menu controls. This option is enabled only when Enable recording is selected. Default = Verify menu initialization event. <p>For more information, see "Advanced Information" on page 114.</p>

Category	Option
Object	<p>Defines record settings for objects recognized as WinObject test objects:</p> <ul style="list-style-type: none"> ➤ Record only the object's basic operation. Enables simplified recording on the WinObject test object. Using this mode may improve recognition of user operations in non-standard cases. This option is cleared by default and should be used only when the default recording method does not meet your needs. For more information, see "Advanced Information" on page 114. ➤ Record Click. Specifies whether the Click operation should be recorded when the mouse button is pressed (On mouse button down) or when the mouse button is released (On mouse button up). This option is enabled only when Record only the object's basic operation is selected. Default = On mouse button down.
Tab	<p>Defines record settings for tab objects:</p> <ul style="list-style-type: none"> ➤ Record only the object's basic operation. Enables simplified recording on the tab. Using this mode may improve recognition of user operations in non-standard cases. This option is cleared by default and should be used only when the default recording method does not meet your needs. For more information, see "Advanced Information" on page 114. ➤ Record Select. Specifies whether the Select operation should be recorded when the mouse button is pressed (On mouse button down) or when the mouse button is released (On mouse button up). This option is enabled only when Record only the object's basic operation is selected. Default = On mouse button up.

Category	Option
<p>Toolbar</p>	<p>Defines record settings for toolbar objects:</p> <ul style="list-style-type: none"> ▶ Record only the object's basic operation. Enables simplified recording on the toolbar. Using this mode may improve recognition of user operations in non-standard cases. This option is cleared by default and should be used only when the default recording method does not meet your needs. For more information, see "Advanced Information" on page 114. ▶ Record Press. Specifies whether the Press operation should be recorded when the mouse button is pressed (On mouse button down) or when the mouse button is released (On mouse button up). This option is enabled only when Record only the object's basic operation is selected. Default = On mouse button up.
<p>Tree view</p>	<p>Defines record settings for tree view objects:</p> <ul style="list-style-type: none"> ▶ Record only the object's basic operation. Enables simplified recording on the tree view. Using this mode may improve recognition of user operations in non-standard cases. This option is cleared by default and should be used only when the default recording method does not meet your needs. For more information, see "Advanced Information" on page 114. ▶ Record Select. Specifies whether the Select operation should be recorded when the mouse button is pressed (On mouse button down) or when the mouse button is released (On mouse button up). This option is enabled only when Record only the object's basic operation is selected. Default = On mouse button up. ▶ Record tree items. Specifies whether tree items are recorded By name or By virtual index. Default = By name.

Category	Option
Window	<p>Defines record settings for window objects:</p> <ul style="list-style-type: none"> ▶ Record only the object's basic operation. Enables simplified recording on the window. Using this mode may improve recognition of user operations in non-standard cases. This option is cleared by default and should be used only when the default recording method does not meet your needs. For more information, see "Advanced Information" on page 114. ▶ Record Click. Specifies whether the Click operation should be recorded when the mouse button is pressed (On mouse button down) or when the mouse button is released (On mouse button up). This option is enabled only when Record only the object's basic operation is selected. Default = On mouse button up.
Keyboard	<p>Defines record settings for operations performed on the keyboard:</p> <ul style="list-style-type: none"> ▶ Keyboard state detection. Specifies which API QuickTest should use to detect the keyboard state. Default = Standard. <p>For more information, see "Advanced Information" on page 114.</p>
Utility object	<p>Defines record settings for utility objects:</p> <ul style="list-style-type: none"> ▶ Record SystemUtil.Run commands. Specifies whether QuickTest records SystemUtil.Run commands when you open an application during a recording session. This option is selected by default. For more information on the SystemUtil.Run method, see the <i>HP QuickTest Professional Object Model Reference</i>.

Run Settings Options

You can specify how QuickTest treats certain objects when running a test or component. The Windows Applications > Advanced pane includes the following **Run settings** options:

Option	Description
Edit Box	Defines run settings for Edit objects: <ul style="list-style-type: none"> ▶ Click Edit box before inserting text. Specifies whether QuickTest performs a Click operation to set the focus in an edit box before inserting text in it while running a test or component. This option is cleared by default. ▶ Use keyboard events to perform Set operations. When selected, instructs QuickTest to simulate keyboard events when performing Set operations on edit boxes during a run session. When cleared, instructs QuickTest to use API or Window messages for edit box Set operations. This option is cleared by default.

Advanced Information

The following information is intended for users with expertise in the Win32 API and the Windows messages model. It expands on the information provided for some of the Windows Applications > Advanced options in the previous section.

Always enumerate child windows

If QuickTest does not correctly record an object in your application, you can select this option to force QuickTest to enumerate all windows in the system. This means that even when QuickTest looks for a window without `WS_CHILD` style, it enumerates all windows in the system and not only the top-level windows.

You should select this option if there is a window in your application that does not have a `WS_CHILD` style but does have a parent (not an owner) window.

Record only the object's basic operation

In general, QuickTest records operations on Windows objects based on Windows messages sent by the application. QuickTest recognizes the sequence of Windows messages sent to a specific application window by the system, and uses a smart algorithm to determine which operation to record.

In rare cases (where a non-standard message sequence is used), the smart algorithm may record unwanted operations. Select this option if you want to record only the object's basic operation when the selected event occurs. When you select this option, you can also select when to record the operation. If you select **On mouse button down**, QuickTest records the operation performed when a WM_LBUTTONDOWN message is detected; if you select **On mouse button up**, QuickTest records the operation performed when a WM_LBUTTONUP message is detected.

Keyboard state detection

If QuickTest does not correctly record keyboard key combinations (for example, CTRL+Y, or ALT+CTRL+HOME), you can try changing the default setting for this option. Following is a brief explanation of each of the options:

- ▶ **Standard.** Uses the **GetKeyboardState** API to detect the keyboard state. For more information, see <http://msdn2.microsoft.com/en-us/library/ms646299.aspx>.
- ▶ **Alternate synchronous.** Uses the **GetKeyState** API to detect the keyboard state. For more information, see <http://msdn2.microsoft.com/en-us/library/ms646301.aspx>.
- ▶ **Alternate asynchronous.** Uses the **GetAsyncKeyState** API to detect the keyboard state. For more information, see <http://msdn2.microsoft.com/en-us/library/ms646293.aspx>.

Menu recording mode

In most applications, Windows sends a WM_CONTEXTMENU message, WM_ENTERMENULOOP message, WM_INITMENU message, WM_INITMENUPOPUP message, or other initialization message when a user opens a menu. Windows then sends a WM_MENUSELECT message when a user selects a menu item.

The **Verify menu initialization** event option instructs QuickTest to record menu operations only after detecting a menu initialization message. If QuickTest does not correctly record menu operations, or if your application does not send initialization messages before sending WM_MENUSELECT messages, try using the **Ignore menu initialization** event option. This instructs QuickTest to always record menu operations.

Part II

The .NET Add-in

4

Using the Silverlight Add-in

You can use the QuickTest Professional Silverlight Add-in to test objects (controls) in Silverlight applications.

For details on supported Silverlight environments, see the **.NET Add-in** section of the *HP QuickTest Professional Product Availability Matrix*, available from the Documentation Library Home page or the root folder of the QuickTest Professional DVD.

The Silverlight Add-in provides test objects, methods, and properties that can be used when testing objects in Silverlight applications. For more information, see the **Silverlight** section of the *HP QuickTest Professional Object Model Reference*.

Note: To work with the Silverlight Add-in, your Silverlight application must be initialized with the **EnableHtmlAccess** property value set to 'True'. For details, see [http://msdn.microsoft.com/en-us/library/cc838264\(VS.95\).aspx](http://msdn.microsoft.com/en-us/library/cc838264(VS.95).aspx)

The following table summarizes basic information about the Silverlight Add-in and how it relates to some commonly-used aspects of QuickTest.

General Information	
Add-in Type	This is a Web-based add-in. Much of its functionality is the same as other Web-based add-ins. This add-in is installed as a sub add-in of the .NET Add-in. See "Testing Web-Based Applications" on page 45.

<p>Checkpoints and Output Values</p>	<ul style="list-style-type: none"> ▶ See the sections describing checkpoints and output values in the <i>HP QuickTest Professional User Guide</i> (for tests), and the <i>HP QuickTest Professional for Business Process Testing User Guide</i> (for components). ▶ See "Supported Checkpoints and Output Values Per Add-in" on page 769.
<p>Extending the Silverlight Add-in</p>	<p>Silverlight Add-in Extensibility (described on page 121) enables you to develop support for testing third-party and custom Silverlight controls that are not supported out-of-the-box by the QuickTest Professional Silverlight Add-in.</p>
<p>Other</p>	<p>You can use the QuickTest Silverlight Add-in to test Silverlight out-of-browser applications. To do this you must register the Microsoft sslauncher.exe as a browser control. This executable is located in the Silverlight installation folder, for example, C:\Program Files\Microsoft Silverlight. You can do this using the QuickTest Register Browser Control Utility, which is available from Start > HP QuickTest Professional > Tools > Register New Browser Control. For more information, see "Registering Browser Controls" on page 658.</p>
<p>Prerequisites</p>	
<p>Opening Your Application</p>	<p>You must open QuickTest before opening your Silverlight application.</p>
<p>Add-in Dependencies</p>	<p>The Web Add-in must be loaded.</p>
<p>Other</p>	<p>To work with the Silverlight Add-in, .NET FrameWork 3.0 or later must be installed on your computer.</p>
<p>Setting Preferences</p>	
<p>Options Dialog Box</p>	<p>Use the Web pane. (Tools > Options > Web node)</p> <p>For more information, see "Setting Web Testing Options" on page 52.</p>
<p>Record and Run Settings Dialog Box (tests only)</p>	<p>Use the Web tab. (Automation > Record and Run Settings)</p> <p>See "Setting Web Record and Run Options" on page 46.</p>

Test Settings Dialog Box (tests only)	Use the Web pane. (File > Settings > Web node) See "Defining Web Settings for Your Test" on page 69.
Custom Active Screen Capture Settings Dialog Box (tests only)	Use the Web section. (Tools > Options > Active Screen node > Custom Level) See the section on the Custom Active Screen Capture Settings dialog box in the <i>HP QuickTest Professional User Guide</i> .
Application Area Settings Dialog Box (components only)	Use the Web pane. (File > Settings > Web node) See "Defining Web Settings for Your Application Area" on page 71.

This chapter includes:

- Silverlight Add-in Extensibility on page 121
- Troubleshooting and Limitations - Silverlight on page 123

Silverlight Add-in Extensibility

QuickTest Professional Silverlight Add-in Extensibility enables you to develop support for testing third-party and custom Silverlight controls that are not supported out-of-the-box by the QuickTest Professional Silverlight Add-in.

If the test object class that QuickTest uses to represent a control does not provide the operations and properties necessary to operate on your control, you can use Silverlight Add-in Extensibility to create a new test object class.

You can then map the control to the new test object class, and design the test object class behavior using .NET programming. You can program how operations are performed on the control, how properties are retrieved, and more.

You can also teach QuickTest to treat a control that contains a set of lower-level controls as a single functional control, instead of relating to each lower-level control separately.

To implement Silverlight Add-in Extensibility, you need to be familiar with:

- QuickTest Professional and its Object Model Reference
- The behavior of the custom control (operations, properties, events)
- .NET programming in C#
- XML (basic knowledge)

You can install the WPF and Silverlight Add-in Extensibility SDK from the **Add-in Extensibility and Web 2.0 Toolkits** option in the QuickTest Professional setup program.

The SDK also includes project templates and a wizard for Microsoft Visual Studio, that simplify setting up of your Silverlight Add-in Extensibility project.

For details on implementing Silverlight Add-in Extensibility, see the WPF and Silverlight Add-in Extensibility Help, available from the QuickTest Professional Extensibility Documentation program group (**Start > Programs > HP QuickTest Professional > Extensibility > Documentation**).

A printer-friendly (PDF) version of the *HP QuickTest Professional WPF and Silverlight Add-in Extensibility Developer Guide* is available in the **<QuickTest Professional installation folder>\help\Extensibility** folder.

Troubleshooting and Limitations - Silverlight

This section describes troubleshooting and limitations for the Silverlight Add-in.

- ▶ QuickTest retrieves incorrect values for the **all items** and **selection** properties for **ListBox** and **ComboBox** controls that are bound to data via a template.
- ▶ In some versions of Internet Explorer, the Silverlight application becomes active only after a **Click** operation is performed. In these cases, QuickTest may fail to run test steps unless an initial **Click** operation is performed.

Workaround: Insert a step containing a **Click** operation on the Silverlight application before performing other operations on the application.

- ▶ Applications containing numerous controls might have performance problems while recording.

Workaround: Change the Active Screen capture level to **Partial** or **Minimum** to capture less information. To do this, select the **Tools > Options > Active Screen** node and modify the setting.

- ▶ If a recovery scenario uses the **Object State** trigger, the following may occur:
 - ▶ The recovery scenario may detect redundant test objects when checking a **SlvWindow** state.
 - ▶ The run results may not include all nodes related to the recovery scenario.
- ▶ You cannot create a virtual object for an area in a Silverlight application.
- ▶ If you insert a text area checkpoint or a text area output value using the Windows API text recognition mechanism (as opposed to the OCR mechanism), all of the text on the Silverlight control is captured (instead of only the text from the selected area).
- ▶ For some test objects, if you try to insert a text checkpoint from the Active Screen, the text checkpoint cannot be inserted and an error message is displayed.
- ▶ Recording on windowless Silverlight applications is not supported on Mozilla Firefox.

- If you open a Silverlight context menu when creating or editing a test, you must close the context menu control (for example, by pressing ESC) before you close the browser. Otherwise, during a run session, the browser window will remain open.

Workaround: Add the following line to the test before the line that closes the browser:

```
Browser("SilverLightAUT").Page("SilverLightAUT").SlvWindow("Page").SlvButton("Login").Type micEsc
```

Example:

```
Browser("SilverLightAUT").Page("SilverLightAUT").SlvWindow("Page").SlvButton("Login").ShowContextMenu
```

```
Browser("SilverLightAUT").Page("SilverLightAUT").SlvWindow("Page").SlvButton("Login").Type micEsc
```

```
Browser("SilverLightAUT").Close
```

5

Testing .NET Web Forms Applications

You can use the .NET Add-in to test .NET Web Forms objects (controls).

The .NET Add-in provides test objects, methods, and properties that can be used when testing objects in .NET Web Forms applications. For more information, see the **.NET Web Forms** section of the *HP QuickTest Professional Object Model Reference*.

For details on supported .NET Web Forms environments, see the **.NET Add-in** section of the *HP QuickTest Professional Product Availability Matrix*, available from the Documentation Library Home page or the root folder of the QuickTest Professional DVD.

The following table summarizes basic information about testing .NET Web Forms applications and how it relates to some commonly-used aspects of QuickTest.

General Information	
Add-in Type	The .NET Add-in functions like a Web-based add-in when testing .NET Web Forms controls. Much of its functionality is the same as other Web-based add-ins. See "Testing Web-Based Applications" on page 45.
Checkpoints and Output Values	<ul style="list-style-type: none">▶ See the sections describing checkpoints and output values in the <i>HP QuickTest Professional User Guide</i> (for tests), and the <i>HP QuickTest Professional for Business Process Testing User Guide</i> (for components).▶ See "Supported Checkpoints and Output Values Per Add-in" on page 769.▶ See "Checking .NET Web Forms Objects and Outputting Values" on page 128.

Prerequisites	
Opening Your Application	You must open QuickTest and set the Record and Run options before opening your .NET Web Forms application. Open your application only after you begin the recording session.
Add-in Dependencies	The Web Add-in must be loaded.
Setting Preferences	
Options Dialog Box	Use the Web pane. (Tools > Options > Web node) See "Setting Web Testing Options" on page 52.
Record and Run Settings Dialog Box (tests only)	Use the Web tab. (Automation > Record and Run Settings) See "Setting Web Record and Run Options" on page 46
Test Settings Dialog Box (tests only)	Use the Web pane. (File > Settings > Web node) See "Defining Web Settings for Your Test" on page 69.
Custom Active Screen Capture Settings Dialog Box (tests only)	Use the Web section. (Tools > Options > Active Screen node > Custom Level) See the section on the Custom Active Screen Capture Settings dialog box in the <i>HP QuickTest Professional User Guide</i> .
Application Area Settings Dialog Box (components only)	Use the Web pane. (File > Settings > Web node) See "Defining Web Settings for Your Application Area" on page 71.

This chapter includes:

- ▶ Considerations for Testing .NET Web Forms Applications on page 127
- ▶ Checking .NET Web Forms Objects and Outputting Values on page 128
- ▶ Troubleshooting and Limitations - .NET Web Forms on page 129

Considerations for Testing .NET Web Forms Applications

When testing .NET Web Forms Applications applications, consider the following:

- When QuickTest learns .NET Web Forms objects, it does not learn the HTML elements that comprise the test objects. For example, when QuickTest learns the WbfGrid test object, the WbfGrid object is the bottommost object in the hierarchy, and the HTML elements used to create the grid's cells are not learned.
- When you load the .NET Add-in, the Web event recording configurations designed for this add-in are loaded and are used whenever you record on a .NET Web Forms object. The .NET Web Forms Web event recording configurations do not affect the way QuickTest behaves when you record on other non-.NET Web Forms Web objects. For more information, see "Web Event Recording Configurations" on page 74.
- For more information on QuickTest functionality, see the *HP QuickTest Professional User Guide* and the *HP QuickTest Professional for Business Process Testing User Guide*.

Checking .NET Web Forms Objects and Outputting Values

You can check or output values from supported .NET Web Forms controls and use the **Object** property to retrieve internal properties.

Accessing Internal Properties and Methods of Run-Time .NET Web Forms Objects

You can use the **Object** property to retrieve internal (native) properties and activate internal methods of any .NET Web Forms object in your application.

In the example below, the **orientation** property of the **WbfTabStrip** control is returned and displayed in a message box.

```
MsgBox Browser("WebControls:").Page("Page").WbfTabStrip("WbfTabStrip").  
    Object.Orientation
```

The **Object** property is also useful for verifying the value of properties that are not available using a standard checkpoint.

For more information on the **Object** property and for information on .NET Web Forms test objects, methods, and properties see the **.NET Web Forms** section of the *HP QuickTest Professional Object Model Reference*.

Troubleshooting and Limitations - .NET Web Forms

This section describes troubleshooting and limitations for the .NET Web Forms Add-in.

General

- ▶ **xpath** and **css** properties are not supported for .NET Web Forms test objects or for other Web-based test objects that have .NET Web Forms parent test objects.
- ▶ Tests on **WbfTreeView** test objects that contain special characters may not run as expected.

Workaround: To run a test on a **WbfTreeView** item that contains special characters, use the **#index** format. See the *.NET Web Forms Object Model Reference Help* for details.

- ▶ **WbfTreeView**, **WbfToolBar**, and **WbfTabStrip** test objects are not supported for browser control applications.
- ▶ Active Screen operations are not supported for **WbfTreeView**, **WbfToolBar**, and **WbfTabStrip** objects.
- ▶ Performing a **Select** or **Expand** operation on a **WbfTreeView** object that causes page navigation may fail due to a synchronization problem.

Workaround: Try running the test on the **WbfTreeView** object step-by-step, meaning instead of using **WbfTreeView.Select "item1;item2;item3;"** Use
WbfTreeView.Expand "item1"
WbfTreeView.Expand "item1;item2"
WbfTreeView.Select "item1;item2;item3;"

- ▶ Working on a .NET Web Forms application that has calendars with more than one unified style is not fully supported.
- ▶ The value of the **Selected Date** and **Selected Range** identification properties is always **none** for **WbfCalendar** objects in selection mode **none**.
- ▶ To retrieve correct values for **WbfCalendar Selected Date** and **Selected Range** identification properties, the selected date or range must be currently visible in your Web Forms application.

- ▶ All operations on grouping areas in **WbfUltraGrid** objects (**Infragistics UltraWebGrid**) are not recorded.
- ▶ Operations performed in a rapid sequence on **WbfUltraGrid** objects may not be recorded.

Workaround: Try to limit the recording to 1-2 operations per second.

- ▶ **WbfUltraGrid** column names are comprised of the inner HTML of the column header, and therefore may include extraneous information.
- ▶ **WbfUltraGrid** may fail to sort columns in a descending order when the column is not already sorted.

Workaround: Split the Sort call into two calls—first sort in ascending order, then sort in descending order. For example:

Change:

```
WbfUltraGrid("UltraWebGrid1").Sort "Model","Descending"
```

To:

```
WbfUltraGrid("UltraWebGrid1").Sort "Model","Ascending"  
WbfUltraGrid("UltraWebGrid1").Sort "Model","Descending"
```

Creating, Editing, and Running Testing Documents

- ▶ QuickTest may recognize some Web Forms grids as **WebTables** instead of **WbfGrid** test objects.

Workaround: Do one of the following:

- ▶ Modify the Web forms control so that it meets one of the following conditions:
 - ▶ The **class** attribute contains the string **DataGrid**.
 - ▶ The **id** attribute contains at least one of the strings **DataGrid** or **GridView**.

- ▶ Modify the rules that QuickTest uses to determine when to identify a Web Forms table control as a DataGrid or GridView (and learn it as a WbfGrid test object).

These rules are defined in:

<QuickTest installation folder>\dat\WebFormsConfiguration.xml.

The file contains comments that describe its format and explain how to use it.

- ▶ If you record a test containing .NET Web Forms objects, you can run it only on Microsoft Internet Explorer.

Checkpoint and Output Values

- ▶ WbfTreeView, WbfToolbar, and WbfTabStrip objects are not properly recognized in the Active Screen. Therefore:
 - ▶ You cannot insert checkpoint or output value steps for these objects from the Active Screen.
 - ▶ If you select to insert checkpoints for these objects from the Keyword View or Expert View while in edit mode, the expected values of these objects may be incorrect.

Workaround: Insert checkpoint or output value steps on these objects during a recording session or remove the Active Screen for the relevant step and then insert a checkpoint from the Keyword View or Expert View with your application open to the proper location, so that the values will be retrieved from the application.

- ▶ Text checkpoints are not supported for WbfTreeView, WbfToolbar, and WbfTabStrip objects.

- ▶ The Active Screen image for a `WbfCalendar` object is always saved before navigation. For example, if you click a **NextMonth** link, the Active Screen displays the current month. Therefore, if you create a checkpoint from the Active Screen and insert it after the **Calendar.ShowNextMonth** line, the checkpoint will fail.

Workaround: Do one of the following:

- ▶ Insert checkpoints on calendar objects while recording.
 - ▶ While editing your test, edit the expected value for the checkpoint or insert the checkpoint before the current step.
-
- ▶ Table checkpoints are supported for `WbfUltraGrid` objects only while recording.
 - ▶ When using the **WbfUltraGrid.RowCount** and **WbfUltraGrid.ColumnCount** methods or performing a table checkpoint on a grid that also contains additional grid controls inside it, QuickTest retrieves the rows or columns only for the outermost table. Note that the `rows` property and `RowCount` method count only the non-grouping rows.

6

Testing .NET Windows Forms Applications

You can use the QuickTest Professional .NET Add-in to test .NET Windows Forms objects (controls).

For details on supported .NET Windows Forms environments, see the **.NET Add-in** section of the *HP QuickTest Professional Product Availability Matrix*, available from the Documentation Library Home page or the root folder of the QuickTest Professional DVD.

You can also test most custom .NET controls inherited from the **System.Windows.Forms.Control** regardless of which language was used to create the application (for example, VisualBasic .NET, C#, and so on).

The .NET Add-in provides test objects, methods, and properties that can be used when testing objects in .NET Windows Forms applications. For more information, see the **.NET Windows Forms** section of the *HP QuickTest Professional Object Model Reference*.

The following table summarizes basic information about .NET Windows Forms testing support and how it relates to some commonly-used aspects of QuickTest.

General Information	
Add-in Type	The .NET Windows Forms testing support functions like a Windows-based add-in. Much of its functionality is the same as other Windows-based add-ins. See "Testing Windows-Based Applications" on page 87.

<p>Checkpoints and Output Values</p>	<ul style="list-style-type: none"> ▶ See the sections describing checkpoints and output values in the <i>HP QuickTest Professional User Guide</i> (for tests), and the <i>HP QuickTest Professional for Business Process Testing User Guide</i> (for components). ▶ See "Considerations for Testing .NET Windows Forms Applications" on page 135. ▶ See "Supported Checkpoints and Output Values Per Add-in" on page 769.
<p>Extending the .NET Add-in</p>	<p>.NET Add-in Extensibility (described on page 151) enables you to develop support for testing third-party and custom .NET Windows Forms controls that are not supported out-of-the-box by the QuickTest Professional .NET Add-in.</p>
<p>Prerequisites</p>	
<p>Opening Your Application</p>	<p>You must open QuickTest before opening your .NET Windows Forms application.</p>
<p>Add-in Dependencies</p>	<p>The .NET Add-in must be installed.</p>
<p>Setting Preferences</p>	
<p>Options Dialog Box</p>	<p>Use the Windows Applications pane. (Tools > Options > Windows Applications node) See "The Options Dialog Box: Windows Applications Pane" on page 103.</p>
<p>Record and Run Settings Dialog Box (tests only)</p>	<p>Use the Windows Applications tab. (Automation > Record and Run Settings) See "The Record and Run Settings Dialog Box: Windows Applications Tab" on page 90. Note: If you select the Record and Run only on radio button in the Record and Run Settings dialog box, the settings also apply to (limit) the applications that are recognized for the .NET Windows Spy, the Object Spy, and other pointing hand operations.</p>

Custom Active Screen Capture Settings Dialog Box (tests only)	Use the Windows applications section. (Tools > Options > Active Screen node > Custom Level) See the section on the Custom Active Screen Capture Settings dialog box in the <i>HP QuickTest Professional User Guide</i> .
Application Area Settings Dialog Box (components only)	Use the Applications pane. (File > Settings > Settings node) See the section on defining Application Settings for your application area in the <i>HP QuickTest Professional for Business Process Testing User Guide</i> .

This chapter includes:

- ▶ Considerations for Testing .NET Windows Forms Applications on page 135
- ▶ Checking .NET Windows Forms Objects and Outputting Values on page 136
- ▶ Using the .NET Windows Forms Spy on page 139
- ▶ .NET Add-in Extensibility on page 151
- ▶ Troubleshooting and Limitations - .NET Windows Forms on page 152

Considerations for Testing .NET Windows Forms Applications

- ▶ You can use the Keyword View and Expert View to activate .NET Windows Forms test object operations and native (run-time object) operations, retrieve and set the values of properties, and check that objects in your application exist and function as expected.
- ▶ When you create a checkpoint on a .NET Windows Forms object, QuickTest stores the selected property values of the object. If your application changes, you can modify the captured values to match the new expected values.

- ▶ For more information on QuickTest functionality, see the *HP QuickTest Professional User Guide* and the *HP QuickTest Professional for Business Process Testing User Guide*.

Checking .NET Windows Forms Objects and Outputting Values

You can check or output values from supported .NET Windows Forms grid controls and use the **Object** property to retrieve internal properties.

For more information, see:

- ▶ "Checking .NET Windows Forms Tables and Outputting Their Values" on page 136
- ▶ "Accessing Internal Properties and Methods of Run-Time .NET Windows Forms Objects" on page 138

Checking .NET Windows Forms Tables and Outputting Their Values

You check or output values from supported .NET Windows Forms grid controls using the Table Checkpoint Properties dialog box.

For tables with more than 100 rows, you can specify the rows you want to include in the checkpoint or output value in the Define Row Range dialog box. If you do not specify the rows to include, the table checkpoint or output value captures all data in the current level or view as follows:

When working with:	The table checkpoint or output value captures:
ComponentOne C1FlexGrid and C1TrueDBGrid	The entire grid.
Microsoft Data Grid and DataGrid View	The currently displayed table (parent or child).

When working with:	The table checkpoint or output value captures:
Infragistics UltraWinGrid	The band in which a cell, column, or row is selected.
DevExpress XtraGrid	The view that was most recently set. Tip: Insert a SetView method before your table checkpoint to ensure that the view you want is displayed when the table checkpoint runs.

Apart from the difference in captured information as listed above, you define a table checkpoint or output value for .NET Windows Forms in the same way as you do for any other table. For more information, see the sections on checkpoints and output values in the *HP QuickTest Professional User Guide*.

Notes:

For a list of supported .NET Windows Forms grid control versions, see the *HP QuickTest Professional Product Availability Matrix*, available from the Documentation Library Home page or the root folder of the QuickTest Professional DVD.

.NET Windows Forms table checkpoints and output value steps can be created only for objects that QuickTest recognizes as SwfTable objects. QuickTest does not treat SwfPropertyGrid test objects as table objects.

Accessing Internal Properties and Methods of Run-Time .NET Windows Forms Objects

You can use the **Object** property to retrieve internal (native) properties and activate internal methods of any .NET Windows Forms object in your application.

For example, you can set the focus to a particular button and change its caption using statements similar to the following:

```
Set theButton = SwfWindow("frmWin").SwfButton("OK").Object
theButton.SetFocus
theButton.Caption = "Yes"
```

The **Object** property is also useful for verifying the value of properties that are not available using a standard checkpoint.

When you use the **Object** property to retrieve arrays of structures, the **Object** property returns the COM wrapper of the **system.array** object. In your VBScript test or component steps, you can then use the **system.array** object to access the array members.

For example, suppose a button object in your application has a **PointArray** property, which is an array of Point structures. To access the first item in the **PointArray** property, you would use the following expression:

```
SwfWindow("Form1").SwfButton("button1").Object.PointArray.GetValue1(0)
```

If the same object had an **IntArray** property, which was an array of integers, you would use the following expression to access the first item in the **IntArray** property:

```
SwfWindow("Form1").SwfButton("button1").Object.IntArray(0)
```

For additional information on the **Object** property and for information on .NET Windows Forms test objects, methods, and properties see the **.NET Windows Forms** section of the *HP QuickTest Professional Object Model Reference*.

Using the .NET Windows Forms Spy

The .NET Windows Forms Spy enables you to select a specific control in your .NET application, view its run-time object properties and values, change property values in the application in run-time, listen to events on a specific control, view the event arguments, and fire events back at the application.

You can use the .NET Windows Forms Spy to help you develop extensibility for .NET Windows Forms controls.

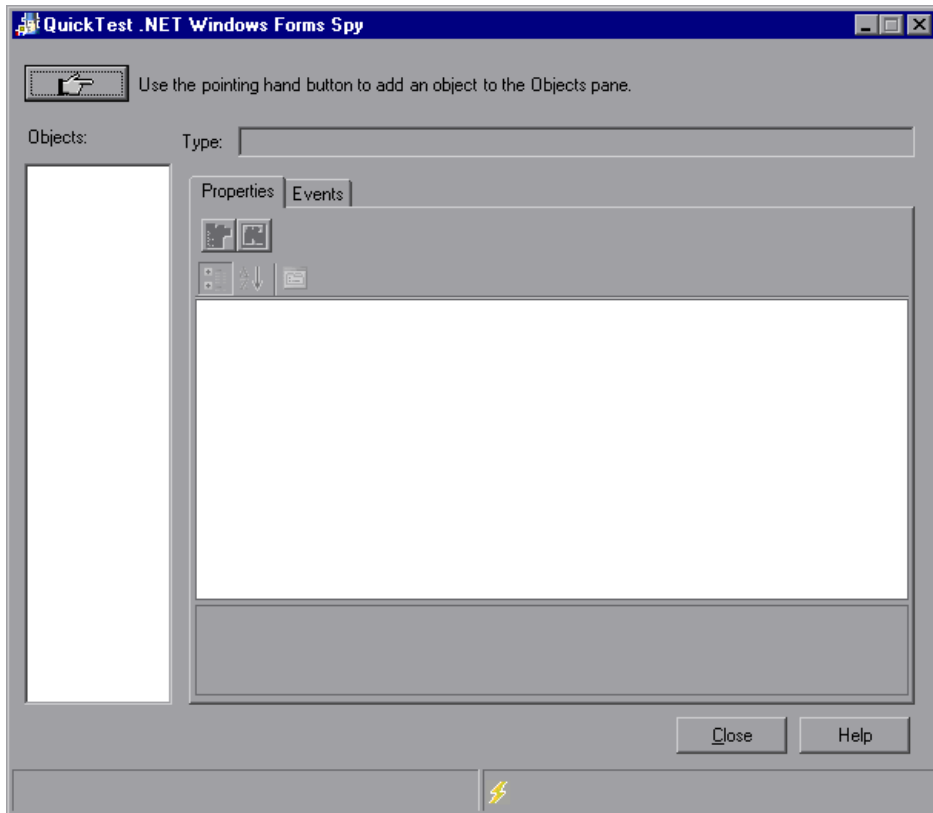
To spy on a .NET Windows Forms application, make sure that the application is specified in the Windows Applications tab of the Record and Run Settings dialog box, and that the application is running with Full Trust. If the application is not defined to run with Full Trust, you cannot spy on the .NET application's Windows Forms controls with the .NET Windows Forms Spy. For information on defining trust levels for .NET applications, see Microsoft documentation.

The .NET Windows Forms Spy is intended for advanced QuickTest users, especially those who are using .NET Add-in Extensibility to create support for custom .NET Windows Forms controls. The .NET Windows Forms Spy can assist you in examining .NET Windows Forms controls within your application and seeing which events cause it to change (to facilitate recording and running) and how the changes manifest themselves in the control's state.

Note: The .NET Windows Forms Spy runs in the context of your .NET application, not in the QuickTest context. The objects and run-time object properties on which you are spying are the raw .NET objects in your application, and not the .NET test objects used in QuickTest. Since the .NET Windows Forms Spy runs in the context of your .NET application, you can close QuickTest while you use the .NET Windows Forms Spy. However, QuickTest must be open if you want to use the pointing hand mechanism to spy on additional objects. If you close the .NET application on which you are spying, the QuickTest .NET Windows Forms Spy window is closed automatically.

To use the .NET Windows Forms Spy to spy on an object:

- 1** Make sure that the application on which you want to spy is specified in the Windows Applications tab of the Record and Run Settings dialog box, and that the application is running with Full Trust.
- 2** Open the .NET Windows Forms application to the window containing the object on which you want to spy.
- 3** Select **Tools > .NET Windows Forms Spy**, or press CTL+SHIFT+T. The QuickTest .NET Windows Forms Spy window opens.

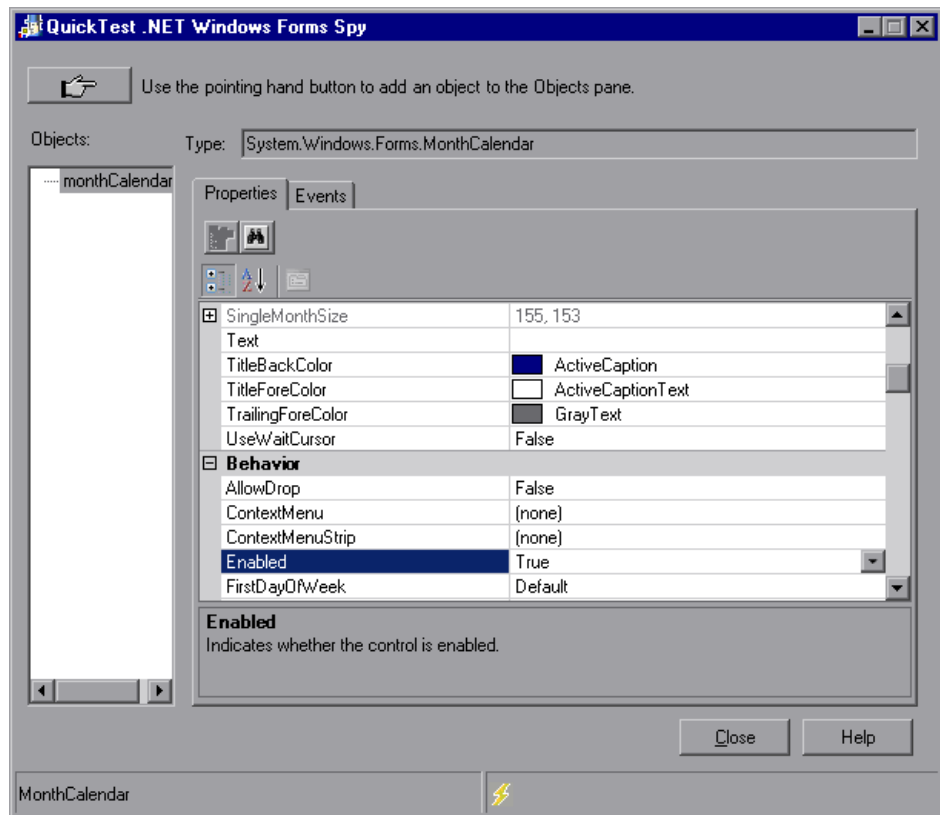





- 4 In the QuickTest .NET Windows Forms Spy window, click the pointing hand. Both QuickTest and the .NET Windows Forms Spy are minimized so that you can point to, and click on, any object in the open application.

For more information on using the pointing hand, see the section describing the pointing hand in the *HP QuickTest Professional User Guide*.

- 5 Click the object whose properties you want to view. If the location you clicked in your application is associated with more than one object, the Object Selection dialog box opens. The objects associated with the location you clicked are displayed in hierarchical order.
- 6 Select the .NET Windows Forms object on which you want to spy and click **OK**. The QuickTest .NET Windows Forms Spy window opens showing the properties and values for the selected object.



The QuickTest .NET Windows Forms Spy window includes the following:

Item	Description
<p>Pointing hand button</p> 	<p>Enables you to select a .NET Windows Forms object on which to spy. You can spy on as many objects within a single .NET application as you want. Each object that you select is added to the Objects pane.</p> <p>Note: If you select an object from a different .NET application, an additional QuickTest .NET Windows Forms Spy window opens, showing the information for the selected object.</p>
<p>Type</p>	<p>Displays the full type name of the selected object.</p>
<p>Objects pane</p>	<p>Displays a hierarchical tree of the objects you selected to spy. For more information, see "Working with the Objects Pane" on page 143.</p>
<p>Properties tab</p>	<p>Enables you to view and modify values of run-time object properties in your .NET application. For more information, see "Working with the Properties Tab" on page 145.</p>
<p>Events tab</p>	<p>Enables you to listen to events in your .NET application and fire them at the application. For more information, see "Working with the Events Tab" on page 148.</p>
<p>status bar</p>	<p>Displays the class name of the object that is selected in the Objects pane, and the event handling status.</p>

- 7 You can repeat steps 4 to 6 to spy on additional objects and add them to the Objects pane in the QuickTest .NET Windows Forms Spy window.

Note: QuickTest must be open if you want to use the pointing hand mechanism to spy on additional objects.

You can now view and modify values of the run-time object's properties. In addition, you can view, listen to, and fire events on the object.

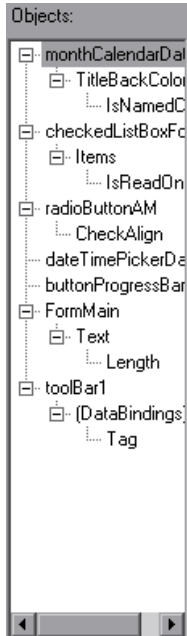
Working with the Objects Pane

The Objects pane contains a list of the objects in your .NET application on which you have spied. Each time you spy on another object in the same .NET application, it is added to the Objects pane. You can spy on as many objects from the same .NET application as you want, using the pointing hand button in the QuickTest .NET Windows Forms Spy window.

The Objects pane also contains any embedded objects that you added from the Properties tab. Each time you add an embedded object to the Objects pane, it is added below its parent object, in a hierarchical format. For more information on viewing properties for embedded objects, see "Working with the Properties Tab" on page 145.

You can select an object in the Objects pane and view or modify its properties and property values, and listen to and fire its events.

You can remove objects from the Objects pane if you no longer need them. You cannot delete the last remaining parent object from the Objects pane. When you remove an object, its descendants (if any) are also removed.



To remove objects from the Objects pane:

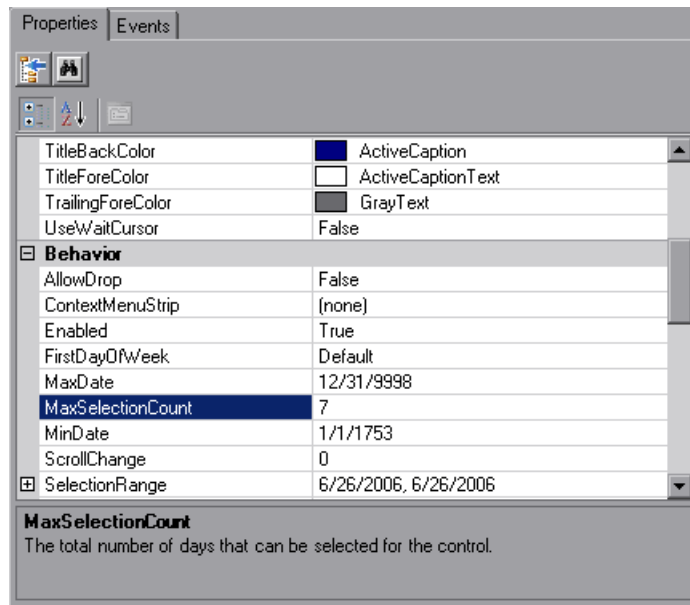
- 1** Select the object that you want to remove.
- 2** Perform one of the following:
 - Right-click the object and select **Remove Object**.
 - Press DELETE.

Working with the Properties Tab

The Properties tab enables you to view run-time object properties and values for objects in your .NET application. You can select a property to display a description of the property below the property grid. You can choose to display the properties alphabetically or by category.

You can change property values in the .NET Windows Forms Spy and apply those changes to your .NET application in run-time.

You can also add embedded objects from the Properties tab to the Objects pane to view their properties.



To view values of .NET Windows Forms run-time object properties:

In the Objects pane, select the object whose run-time object properties you want to view. The properties for the selected object are displayed in the Properties tab, with the property names on the left, and the property values on the right. A description of the selected property is displayed below the properties grid.

To modify values of .NET Windows Forms run-time object properties:

- 1 In the Properties tab, click the property value you want to modify. Properties shown in gray are defined as read-only in the .NET application and cannot be modified.
- 2 Edit the property value as required. The property value displays different types of edit fields, depending on the needs of a particular property. These edit fields include edit boxes, drop-down lists, and links to custom editor dialog boxes.

After you modify a property value, the new value is applied to the run-time instance of the .NET application. For example, you can change the text of an edit box label, change the background color of a dialog box from gray to red, and so on.

Note: Any changes you make to the values of run-time object properties in the .NET application remain in effect only for the current instance of the .NET application. The next time you run the .NET application, the properties will return to their original run-time values.

To view properties of embedded objects:

- 1 In the Properties tab, select the property whose embedded object properties you want to view. For information on locating a property by value, see "To locate a property by its value:" on page 147.



- 2 Click the **Add Selected Property** button. The property is added to the Objects pane, and its run-time object properties and property values (if any) are shown in the Properties tab. Each time you add an embedded object to the Objects pane, it is added below its parent object, in a hierarchical format.

Note: The **Add Selected Property** button is disabled if the property's value is null, or the property is an object with no properties of its own.

To locate a property by its value:



- 1 Click the **Search Property by Value** button. The Find Property by Value dialog box opens.
- 2 In the **Find what** box, specify the value for which you want to search.
- 3 To find only those occurrences in which the capitalization matches the text you entered, select **Match case**.
- 4 Specify the direction from the current cursor location in which you want to search: **Up** or **Down**.
- 5 Click **Find Next**. The .NET Windows Forms Spy locates the property whose value you specified.

To sort the properties grid:

Click one of the following buttons to sort the properties grid in the Properties tab:



- **Categorized.** Lists all properties and property values for the selected object, by category. Categories are listed alphabetically. You can collapse a category to reduce the number of visible properties. When you expand or collapse a category, a plus (+) or minus (-) is displayed to the left of the category name.



- **Alphabetical.** Alphabetically sorts all run-time object properties for the selected object.

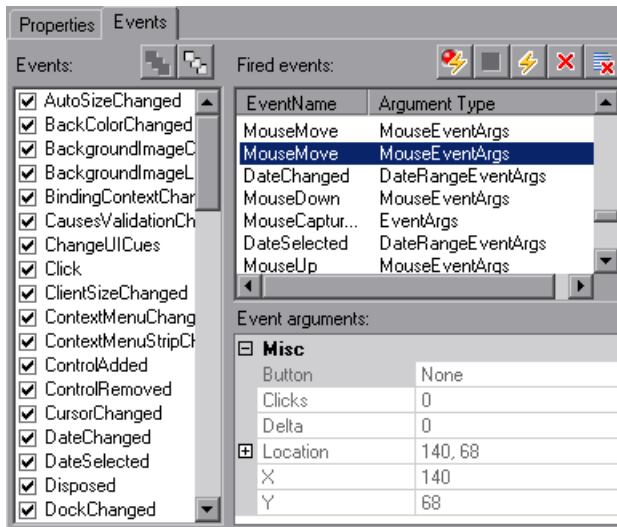


Note: The **Property Pages** button is not currently supported.

Working with the Events Tab

The Events tab enables you to listen to selected events on a specific control in your .NET application. You can then view the event arguments, and fire selected events back at the application.

This is especially useful if you are using .NET Add-in Extensibility to create support for custom .NET Windows Forms controls. You can see which events cause your .NET application to change, so you can implement extensibility for recording operations on specific controls, and also check which events need to be fired to make your .NET application behave the way you want.



To listen to specified events for a .NET Windows Forms object:

- 1 In the Objects pane, select the object to whose events you want to listen.
- 2 In the Events list, select the check boxes for the event types to which you want to listen.

Note: The events that you select affect only the events that are listened to and logged by QuickTest. If you select or clear a check box for an event type after listening to events for an object, the events in the Fired Events list are not changed.



Tip: You can click the **Select All Events** or **Clear All Events** buttons to select or clear all the event check boxes. You can also right-click the Events list and select **Select All** or **Clear All**.



3 Click the **Listen to Selected Events** button. QuickTest starts listening to the specified events on the selected object, and **Listening** is displayed in the status bar.

4 In your .NET application, perform the operations on the object to whose events you want to listen. The specified events are logged as they occur and are shown in the Fired Events list.



5 When you want to stop listening to events, click the **Stop Listening to Events** button. QuickTest stops listening to and logging the specified events.

To view event arguments for a .NET Windows Forms object:

1 In the Objects pane, select the object whose event arguments you want to view.

2 Select the event in the Fired Events list whose arguments you want to view. The selected events arguments and argument values are shown directly below the event, in the Event Arguments list.

To fire selected events on a .NET Windows Forms object:

- 1 In the Objects pane, select the object whose events you want to fire.
- 2 In the Fired Events list, select one or more events that you want to fire on your .NET application. You can select multiple events using standard Windows selection techniques (CTRL and SHIFT keys).

Tip: The selected events are fired in the order in which they appear in the Fired Events list. If the events do not appear in the Fired Events list in the order in which you want to fire them, listen to more events on the object until the events you want are added to the Fired Events list in the required order.

- 3 If the events you selected have editable arguments, you can change their argument values in the Event Arguments list if needed before firing the events. When the events are fired, they will be fired with the modified argument values.



- 4 Click the **Fire Selected Events** button. The selected events are fired in the order in which they appear in the Fired Events list. You can view the effect that firing these events has on the relevant object in your .NET application. The status bar displays that the event firing is in progress, and when it ends.

To remove specific events from the Fired Events list:

- 1 In the Objects pane, select the object whose events you want to remove from the Fired Events list.
- 2 Select the events in the Fired Events list that you want to remove. You can select multiple events using standard Windows selection techniques (CTRL and SHIFT keys).



- 3 Click the **Clear Selected Events** button. The selected events are removed from the Fired Events list.

To clear all events from the Fired Events list:

- 1 In the Objects pane, select the object whose events you want to remove from the Fired Events list.
- 2 Click the **Clear Event List** button. All the logged events are removed from the Fired Events list.



.NET Add-in Extensibility

QuickTest Professional .NET Add-in Extensibility enables you to develop support for testing third-party and custom .NET Windows Forms controls that are not supported out-of-the-box by the QuickTest Professional .NET Add-in.

If the test object class that QuickTest uses to represent a control does not provide the operations and properties necessary to operate on your control, you can use .NET Add-in Extensibility to customize this behavior.

- You can instruct QuickTest to use a different test object class to represent the control.
- You can add operations or override existing ones, using .NET programming, to operate as necessary on the control.
- You can also teach QuickTest to treat a control that contains a set of lower-level controls as a single functional control, instead of relating to each lower-level control separately.

To implement .NET Add-in Extensibility, you need to be familiar with:

- QuickTest Professional and its Object Model Reference
- The behavior of the custom control (operations, properties, events)
- .NET programming in C# or Visual Basic
- XML (basic knowledge)

You can install the .NET Add-in Extensibility SDK from the **Add-in Extensibility and Web 2.0 Toolkits** option in the QuickTest Professional setup program.

The SDK also includes:

- ▶ Project templates and a wizard for Microsoft Visual Studio, that simplify setting up of your .NET Add-in Extensibility project.
- ▶ Samples of support developed using .NET Add-in Extensibility, which you can use to gain a better understanding of how to create your own support.

For installation and implementation details, see the .NET Add-in Windows Forms Extensibility Help, available from the QuickTest Professional Extensibility Documentation program group (**Start > Programs > HP QuickTest Professional > Extensibility > Documentation**).

A printer-friendly (PDF) version of the *HP QuickTest Professional .NET Add-in Extensibility Developer Guide* is available in the <**QuickTest Professional installation folder**>\help\Extensibility folder.

Troubleshooting and Limitations - .NET Windows Forms

This section describes troubleshooting and limitations for the .NET Windows Forms Add-in.

- ▶ Applications containing numerous controls might have performance problems while recording.
Workaround: Change the **Active Screen capture level** to **Partial** or **Minimum** to capture less information. To do this, select the **Tools > Options > Active Screen** node and modify the setting.
- ▶ Navigating in grid controls using keyboard keys (for example, to select cells, rows, and so on) may not be recorded correctly.
Workaround: Use the mouse to navigate in the grid control.
- ▶ If you call the **Back** method for a Microsoft DataGrid control on a table that does not have a parent row, no operation is performed when the statement runs, and no error message is displayed.
- ▶ Grid controls in the Card View mode are not supported.
- ▶ Changing the format of a **DateTimePicker** control during a test run or between record and run sessions (for example, from "Long Date" to "Time") will cause the test run to fail.

- Combo box objects of style **Simple ComboBox** are not supported.
- If a window in the tested application has an opacity property value not equal to 100% (that is, the form is completely or partially transparent), the Active Screen captures the image displayed below the form, and not the transparent window.
- Operations on a grid cell that was selected before you started recording on the grid control may be recorded incorrectly. For example, a child cell element operation may be recorded instead of the parent grid operation (for example, SetCellData).

Workaround: Before performing operations on a cell that is already selected, begin recording, move the focus to another cell, select the required cell, and then perform the required operation.

- When recording steps using low-level recording, default description properties for **WinObject** and **Window** objects do not have constant values. This may lead to different description property values during a run session, which causes steps on these objects to fail.

Workaround:

- **Window test objects.** Before recording, remove the **regexpwndclass** property from the list of mandatory, assistive, and Smart Identification properties using the Object identification dialog box.
- **WinObject test objects.** Do the following:
 - Before recording, remove the **window id** property from the list of mandatory, assistive, and Smart Identification properties using the Object identification dialog box.
 - After recording, change the **regexpwndclass** property value to a regular expression for each WinObject test object in the object repository, and edit the property value to remove everything except for the control type, for example:


```
Change
WindowsForms10.BUTTON.app3
to
.*BUTTON.*
```


7

Using the Windows Presentation Foundation Add-in

You can use the QuickTest WPF Add-in to test WPF (Windows Presentation Foundation) objects (controls).

For details on supported Windows Presentation Foundation environments, see the **WPF Add-in** section of the *HP QuickTest Professional Product Availability Matrix*, available from the Documentation Library Home page or the root folder of the QuickTest Professional DVD.

The WPF Add-in provides test objects, methods, and properties that can be used when testing objects in WPF applications. For more information, see the **NET Windows Presentation Foundation** section of the *HP QuickTest Professional Object Model Reference*.

The following table summarizes basic information about the WPF Add-in and how it relates to some commonly-used aspects of QuickTest.

General Information	
Add-in Type	This is a Windows-based add-in. Much of its functionality is the same as other Windows-based add-ins. This add-in is installed as a sub add-in of the .NET Add-in. See "Testing Windows-Based Applications" on page 87.

<p>Checkpoints and Output Values</p>	<ul style="list-style-type: none"> ▶ See the sections describing checkpoints and output values in the <i>HP QuickTest Professional User Guide</i> (for tests), and the <i>HP QuickTest Professional for Business Process Testing User Guide</i> (for components). ▶ See "Checking WPF Objects and Outputting Values" on page 159. ▶ See "Supported Checkpoints and Output Values Per Add-in" on page 769.
<p>Extending the WPF Add-in</p>	<p>WPF Add-in Extensibility (described on page 162) enables you to develop support for testing third-party and custom WPF controls that are not supported out-of-the-box by the QuickTest Professional WPF Add-in.</p>
<p>Prerequisites</p>	
<p>Opening Your Application</p>	<p>You can open your WPF application before or after opening QuickTest.</p>
<p>Add-in Dependencies</p>	<p>The Web and .NET Add-ins must be installed.</p>
<p>Setting Preferences</p>	
<p>Options Dialog Box</p>	<p>Use the Windows Applications pane. (Tools > Options > Windows Applications node) See "The Options Dialog Box: Windows Applications Pane" on page 103.</p>
<p>Record and Run Settings Dialog Box (tests only)</p>	<p>Use the Windows Applications tab. (Automation > Record and Run Settings) See "The Record and Run Settings Dialog Box: Windows Applications Tab" on page 90. For more information, see "Setting Windows Applications Record and Run Options" on page 89.</p>
<p>Custom Active Screen Capture Settings Dialog Box (tests only)</p>	<p>Use the Windows applications section. (Tools > Options > Active Screen node > Custom Level) See the section on the Custom Active Screen Capture Settings dialog box in the <i>HP QuickTest Professional User Guide</i>.</p>

Application Area Settings Dialog Box (components only)	Use the Applications pane. (File > Settings > Applications node) See the section on defining Application Settings for your application area in the <i>HP QuickTest Professional for Business Process Testing User Guide</i> .
---	--

This chapter includes:

- ▶ Considerations for Working with the WPF Add-in on page 157
- ▶ About WPF User Interface Automation on page 158
- ▶ Checking WPF Objects and Outputting Values on page 159
- ▶ Using WPF Objects, Methods, and Properties to Enhance Your Test or Component on page 160
- ▶ WPF Add-in Extensibility on page 162
- ▶ Troubleshooting and Limitations - Windows Presentation Foundation on page 163

Considerations for Working with the WPF Add-in

- ▶ You can test most custom WPF controls inherited directly or indirectly from the **System.Windows.Controls.Control** class regardless of which language was used to create the application (for example, VisualBasic, .NET, C#, and so on), as well as third-party WPF controls that are inherited from the **System.Windows.Controls.Control** class and implement automation interfaces.
- ▶ WPF uses UI (User Interface) Automation to define UI objects. UI Automation provides standardization of controls and properties for the functionality of objects. The .NET Add-in supports UI Automation through the AutomationElement and Automation Pattern properties.
- ▶ You can use the Keyword View and Expert View to activate WPF test object, Automation object and run-time object methods, retrieve and set the values of properties, and check that objects exist.

- For more information on QuickTest functionality, see the sections describing checkpoints and output values in the *HP QuickTest Professional User Guide* (for tests), and the *HP QuickTest Professional for Business Process Testing User Guide* (for components).

About WPF User Interface Automation

UI Automation provides a single, consistent, reference object for UI elements in multiple frameworks (For example, Win32, WPF, and Trident). With UI Automation, the functionality of objects in the UI is defined by a set of standard control patterns and properties that are common to all objects of that type.

To learn more about UI Automation, see the UI Automation Fundamentals page of the Microsoft Developer Network library at <http://msdn2.microsoft.com/en-us/library/ms753107.aspx>.

Automation Elements

UI Automation exposes every element in the UI as an **Automation Element**. Automation Elements expose common properties of the UI elements they represent.

For example, a button control has the **Automation Element NameProperty**, which references the name or text associated with a button control. That same property is called **caption** or **alt** in Win32 and HTML, respectively. With UI Automation, all button controls have a **NameProperty**, which is mapped to the corresponding property in each framework.

The **Automation Element** also exposes **control patterns** that provide properties and expose methods specific to their control types.

Control Patterns

Control patterns represent discrete pieces of functionality that a control in the UI can perform. The total set of control patterns for a control type define the functionality of that control type.

Control patterns expose **methods** that provide the ability to programmatically manipulate the control.

Control patterns expose **properties** that provide information on the control's functionality and current state.

The set of supported control patterns for a particular control can be dynamically defined. Therefore, a particular control type may not always support the same set of control patterns. For example, a multiline edit box supports scrolling (**scrollpattern** pattern) only if its text exceeds the viewable area.

Some controls types, such as Image controls do not support any control patterns.

QuickTest Professional enables you to access the methods and properties of automation elements and control patterns using special properties in the QuickTest object model for WPF.

For information on how to work with UI Automation in your test or component, see "Accessing Internal Properties and Methods of WPF Objects" on page 160.

Checking WPF Objects and Outputting Values

You use checkpoints to check the properties of WPF objects the same way you check the properties of standard Windows objects. You can also output property values from the objects in your WPF application to use in your test or component.

You can check or output any identification property associated with an object using a standard checkpoint. For a list and description of the identification properties associated with each WPF object, see the **NET Windows Presentation Foundation** section of the *HP QuickTest Professional Object Model Reference*.

To check properties that are not included in the Checkpoint Properties dialog box you can use the **Object**, **AutomationElement**, or **AutomationPattern** property. For more information, see "Accessing Internal Properties and Methods of WPF Objects" on page 160.

For more information on checkpoints and output values, see the sections describing checkpoints and output values in the *HP QuickTest Professional User Guide* (for tests), and the *HP QuickTest Professional for Business Process Testing User Guide* (for components).

Using WPF Objects, Methods, and Properties to Enhance Your Test or Component

A test or component consists of statements coded in Microsoft VBScript. These statements are composed of objects, methods, and/or properties that instruct QuickTest to perform operations or retrieve information. You add these statements using objects from your object repositories, and methods and properties that are available for each object type. In addition, when you record, these statements are generated automatically in response to input to the application. You can also program statements manually, or mix recorded and programmed statements in the same test or component. You create, view, and edit these statements in the Keyword View and/or Expert View.

Accessing Internal Properties and Methods of WPF Objects

When accessing the internal properties and methods of WPF objects, it is important to know which property to use to access the object that contains the information you want to set or retrieve.

- ▶ **AutomationElement property.** Returns the object that gives access to the set of standard properties that expose information about the **Automation Element**.
- ▶ **AutomationPattern property.** Returns the object that gives access to the specific instance of a **Control Pattern**. For more information on the methods and properties that are accessible through the AutomationPattern property, see the .NET Framework Developer Center of the Microsoft Developer Network library at <http://msdn2.microsoft.com/en-us/library/system.windows.automation.aspx>.
- ▶ **Object property.** Returns the object that gives access to properties specific to the actual run-time UI object, as defined by the developer.

Many of the properties and methods accessible through the **AutomationElement** and **AutomationPattern** properties contain the same information as the properties and methods accessible through the **Object** property. However, information available through UI Automation that is accessed through the **Object** property lacks the standardization provided by UI Automation.

Custom properties designed by the developer are accessible only through the **Object** property.

Working with Test Object Methods

QuickTest provides a variety of test object methods that you can use with WPF objects. You can record some of these methods while recording on WPF objects. You can add additional functionality to your test or component by entering statements manually in the Keyword View or Expert View. For information on the available WPF test objects, methods, and properties, see the **NET Windows Presentation Foundation** section of the *HP QuickTest Professional Object Model Reference*.

Note: Because WPF Automation Elements of the same control type may support a different set of control patterns, the test object methods or properties that QuickTest supports for a specific test object may be different from the standard set of methods and properties listed in the **NET Windows Presentation Foundation** section of the *HP QuickTest Professional Object Model Reference*. For example, although the `WpfButton` test object generally does not support the **Set** method, QuickTest may record a **Set** step when clicking a toggle button if the control pattern that was activated during the record session corresponds to the **Set** method.

The .NET Add-in supports IntelliSense and statement completion in the Expert View, including the display of available internal (native) methods and properties when using the **Object** and **AutomationElement** properties in a statement. The .NET Add-in also supports generating statements that access run-time object methods and properties in the Step Generator.

To use IntelliSense with the **Object** and **AutomationElement** properties, ensure that the application you are referencing is open and displays the object to which you are referring.

WPF Add-in Extensibility

QuickTest Professional WPF Add-in Extensibility enables you to develop support for testing third-party and custom WPF controls that are not supported out-of-the-box by the QuickTest Professional WPF Add-in.

If the test object class that QuickTest uses to represent a control does not provide the operations and properties necessary to operate on your control, you can use WPF Add-in Extensibility to create a new test object class.

You can then map the control to the new test object class, and design the test object class behavior using .NET programming. You can program how operations are performed on the control, how properties are retrieved, and more.

You can also teach QuickTest to treat a control that contains a set of lower-level controls as a single functional control, instead of relating to each lower-level control separately.

To implement WPF Add-in Extensibility, you need to be familiar with:

- ▶ QuickTest Professional and its Object Model Reference
- ▶ The behavior of the custom control (operations, properties, events)
- ▶ .NET programming in C#
- ▶ XML (basic knowledge)

You can install the WPF Add-in Extensibility SDK from the **Add-in Extensibility and Web 2.0 Toolkits** option in the QuickTest Professional setup program.

The SDK also includes:

- ▶ Project templates and a wizard for Microsoft Visual Studio, that simplify setting up of your WPF Add-in Extensibility project.
- ▶ Samples of support developed using WPF Add-in Extensibility, which you can use to gain a better understanding of how to create your own support.

For details on implementing WPF Add-in Extensibility, see the WPF Add-in Extensibility Help, available from the QuickTest Professional Extensibility Documentation program group (**Start > Programs > HP QuickTest Professional > Extensibility > Documentation**).

A printer-friendly (PDF) version of the *HP QuickTest Professional WPF Add-in Extensibility Developer Guide* is available in the <**QuickTest Professional installation folder**>\help\Extensibility folder.

Troubleshooting and Limitations - Windows Presentation Foundation

This section describes troubleshooting and limitations for the Windows Presentation Foundation Add-in.

- ▶ When you spy on a WPF object using the Object Spy (or the .NET Windows Forms Spy when the .NET Add-in is loaded), and the Record and Run Settings dialog box is not configured to record on the WPF application on which you are spying, QuickTest recognizes the object as a standard Windows object.

Workaround: Close your WPF application. In QuickTest, open the Record and Run Settings dialog box (**Automation > Record and Run Settings**) and in the Windows Application tab, select the **Record and run test on any Windows application** option. Reopen your WPF application and then spy on it again.

- ▶ When recording steps using low-level recording, default description properties for Windows Presentation Foundation test objects do not have constant values. This may lead to different description property values during a run session, which causes steps on these objects to fail.

Workaround: After recording, change the **regexwndclass** property value for each test object in the object repository to a regular expression, and set the value to `HwndWrapper.*`.

- ▶ When the Active Screen Capture level is set to **Partial**, the first step recorded on each WPF or Silverlight object takes a long time.

Workaround: Set the Active Screen Capture level to **Minimum** (**Tools > Options > Active Screen**).

Part III

The ActiveX Add-in

8

Using the ActiveX Add-in

You can use the QuickTest Professional ActiveX Add-in to test ActiveX objects (controls).

For details on supported ActiveX environments, see the **ActiveX Add-in** section of the *HP QuickTest Professional Product Availability Matrix*, available from the Documentation Library Home page or the root folder of the QuickTest Professional DVD.

The ActiveX Add-in provides test objects, methods, and properties that can be used when testing ActiveX objects in applications. For more information, see the **ActiveX** section of the *HP QuickTest Professional Object Model Reference*.

The following table summarizes basic information about the ActiveX Add-in and how it relates to some commonly-used aspects of QuickTest.

General Information	
Add-in Type	This is a Windows-based add-in. Much of its functionality is the same as other Windows-based add-ins. See "Testing Windows-Based Applications" on page 87.
Checkpoints and Output Values	<ul style="list-style-type: none"> ▶ See the sections describing checkpoints and output values in the <i>HP QuickTest Professional User Guide</i> (for tests), and the <i>HP QuickTest Professional for Business Process Testing User Guide</i> (for components). ▶ See "Supported Checkpoints and Output Values Per Add-in" on page 769.
Prerequisites	
Opening Your Application	The application containing the ActiveX controls on which you want to record must be closed before you begin a QuickTest recording session and set the Record and Run options. Open the application only after you begin the recording session.
Add-in Dependencies	None
Setting Preferences	
Options Dialog Box	Use the Windows Applications pane. (Tools > Options > Windows Applications node) See "The Options Dialog Box: Windows Applications Pane" on page 103.

<p>Record and Run Settings Dialog Box (tests only)</p>	<p>Use the Windows Applications tab. (Automation > Record and Run Settings)</p> <p>See "The Record and Run Settings Dialog Box: Windows Applications Tab" on page 90.</p> <p>Notes:</p> <ul style="list-style-type: none"> ▶ If you select the Record and Run only on radio button in the Record and Run Settings dialog box, the settings also apply to (limit) the applications that are recognized for Object Spy and other pointing hand operations. ▶ QuickTest recognizes ActiveX objects only in applications that are opened after changing the record and replay settings in the Windows Applications tab of the Record and Run Settings dialog box.
<p>Custom Active Screen Capture Settings Dialog Box (tests only)</p>	<p>Use the Windows applications section. (Tools > Options > Active Screen node > Custom Level)</p> <p>See the section on the Custom Active Screen Capture Settings dialog box in the <i>HP QuickTest Professional User Guide</i>.</p>
<p>Application Area Settings Dialog Box (components only)</p>	<p>Use the Applications pane. (File > Settings > Applications node)</p> <p>See the section on defining Application Settings for your application area in the <i>HP QuickTest Professional for Business Process Testing User Guide</i>.</p>

This chapter includes:

- ▶ Considerations for Working with the ActiveX Add-in on page 170
- ▶ Troubleshooting and Limitations - ActiveX Add-in on page 171

Considerations for Working with the ActiveX Add-in

- ▶ When you create a checkpoint on an ActiveX control, QuickTest captures all the properties for an ActiveX control, but it does not select any properties to check.
- ▶ When testing ActiveX objects in a browser, the top-level ActiveX object is inserted within the standard Web object hierarchy, for example, `Browser.Page.ActiveX`.
- ▶ QuickTest can record on standard controls within an ActiveX control and if an ActiveX control contains another ActiveX control, then QuickTest can record and run on this internal control as well. For example, suppose your ActiveX control is a calendar that contains a drop-down list from which you can choose the month. If you record a click in the list to select the month of May, QuickTest records this step in the Expert View as:

```
Dialog("ActiveX Calendars").ActiveX("SMonth Control").  
WinComboBox("ComboBox").Select "May"
```

- ▶ Loading the ActiveX and Siebel add-ins together may cause problems when recording on some ActiveX methods.
- ▶ When creating a programmatic description for an ActiveX test object and the relevant run-time object is windowless (has no window handle associated with it), you must add the **windowless** property to the description and set its value to **True**.

For example:

```
Set ButDesc = Description.Create  
ButDesc("ProgId").Value = "Forms.CommandButton.1"  
ButDesc("Caption").Value = "OK"  
ButDesc("Windowless").Value = True  
Window("Form1").AcxButton(ButDesc).Click
```

For more information, see the section on using programmatic descriptions in the *HP QuickTest Professional User Guide*.

- ▶ If a "windowless" ActiveX radio button object is not first activated by clicking on it (**AcxRadioButton.Click**) or by using the **Set** method, a step containing the **AcxRadioButton.GetVisibleText** method will return an error stating that the object is not visible.

Workaround: Insert a step using the **Click** or **Set** methods prior to any step that uses the **GetVisibleText** method on a "windowless" ActiveX radio button object.

- ▶ For more information on QuickTest functionality, see the *HP QuickTest Professional User Guide* and the *HP QuickTest Professional for Business Process Testing User Guide*.

Troubleshooting and Limitations - ActiveX Add-in

This section describes troubleshooting and limitations for the ActiveX Add-in.

Creating, Editing, and Running Testing Documents

- ▶ In the following ActiveX test object methods, if you specify the column by name, an error occurs when you run the test: **ActivateCell**, **ActivateColumn**, **SelectCell**, **SetCellData**, **SelectColumn**.

Workaround: When calling these methods, specify the column by number.

- ▶ When inserting steps in the Expert View for a Web application that has a mixed hierarchy of Java objects inside an ActiveX control, then it may take a long time for QuickTest to retrieve the possible argument values (dynamic list of values) for ActiveX arguments.

Workaround: Insert these steps using the Keyword View (where the dynamic list of values functionality is not used).

- ▶ If QuickTest Professional does not recognize an ActiveX control inside a Web page, reduce the security level within your Microsoft Internet Explorer browser.

- ▶ If an ActiveX control's internal properties have the same name as the ActiveX properties created by QuickTest Professional, retrieval and verification of such properties may be problematic.

Workaround: You can access the internal properties of an ActiveX control using the **Object** property.

- ▶ Methods performed on row and column positions for Apex, DataBound, and Sheridan grids return the values of the visible positions and not the absolute positions within the tables.

Workaround: Use the scroll bar while recording in order to display the required cells.

- ▶ When recording on an ActiveX control, wait for the recorded step to appear before moving the mouse. Moving the mouse too quickly may result in a corrupted Active Screen for that step.
- ▶ The **AcxTable.RowCount** method is not supported for the Microsoft Data Bound Grid control.
- ▶ QuickTest may fail to properly capture some of the internal properties of windowless ActiveX controls, such as **x**, **y**, **height**, and **width**.
- ▶ Recording on windowless ActiveX controls may result in some additional steps added to your test or component (such as a **Click** method in addition to a **Set** method on an AcxRadioButton object). These additional steps will not cause the run session to fail.
- ▶ **Drag** and **Drop** operations on windowless ActiveX controls are not supported.

Checkpoints and Output Values

- ▶ ActiveX table checkpoints capture only visible rows in data bound grids.
- ▶ When you insert a checkpoint on an ActiveX table from the Active Screen, the browser (or application) must be open to the same page (or screen). Otherwise, some data from the ActiveX table will be missing.

Workaround: Create ActiveX table checkpoints while recording.

- ▶ Checkpoints and output values for ActiveX properties of type VT_DISPATCH are not supported.

- ▶ Checkpoints and output values for write-only ActiveX properties are not supported.
- ▶ If you perform an update run (**Automation > Update Run Mode**) on a test that contains checkpoints or output values for windowless ActiveX controls, and then you rerun the test, the run session may fail. This is because a hidden property called "windowless" is missing from the test object description.

Workaround: You can either relearn the problematic ActiveX controls, or you can add the "windowless" property with a value of 1 to all problematic, windowless ActiveX controls.

Part IV

The Delphi Add-in

9

Using the Delphi Add-in

You can use the QuickTest Professional Delphi Add-in to test Delphi objects (controls).

The Delphi Add-in supports testing on Delphi controls created in the Delphi IDE and based on the Win32 VCL library. For details on supported Delphi environments, see the **Delphi Add-in** section of the *HP QuickTest Professional Product Availability Matrix*, available from the Documentation Library Home page or the root folder of the QuickTest Professional DVD.

The Delphi Add-in provides test objects, methods, and properties that can be used when testing objects in Delphi applications. For more information, see the **Delphi** section of the *HP QuickTest Professional Object Model Reference*.

The following table summarizes basic information about the Delphi Add-in and how it relates to some commonly-used aspects of QuickTest.

General Information	
Add-in Type	This is a Windows-based add-in. Much of its functionality is the same as other Windows-based add-ins. See "Testing Windows-Based Applications" on page 87.
Checkpoints and Output Values	<ul style="list-style-type: none"> ▶ See the sections describing checkpoints and output values in the <i>HP QuickTest Professional User Guide</i> (for tests), and the <i>HP QuickTest Professional for Business Process Testing User Guide</i> (for components). ▶ See "Supported Checkpoints and Output Values Per Add-in" on page 769.
Extending the Delphi Add-in	Delphi Add-in Extensibility (described on page 182) enables you to develop support for testing third-party and custom Delphi controls that are not supported out-of-the-box by the QuickTest Professional Delphi Add-in.
Prerequisites	
Opening Your Application	You can open your Delphi application before or after opening QuickTest.
Add-in Dependencies	None
Other	Before running a test on a Delphi application, the application being tested must be compiled with the QuickTest Professional agent MicDelphiAgent . See "Enabling Communications Between QuickTest Professional and Your Delphi Application" on page 180.
Setting Preferences	
Options Dialog Box	Use the Windows Applications pane. (Tools > Options > Windows Applications node) See "The Options Dialog Box: Windows Applications Pane" on page 103.

<p>Record and Run Settings Dialog Box (tests only)</p>	<p>Use the Windows Applications tab. (Automation > Record and Run Settings)</p> <p>See "The Record and Run Settings Dialog Box: Windows Applications Tab" on page 90.</p> <p>Notes:</p> <ul style="list-style-type: none"> ▶ QuickTest recognizes only Delphi applications that have been precompiled with the MicDelphiAgent.pas module. For more information, see "Enabling Communications Between QuickTest Professional and Your Delphi Application" on page 180. ▶ In some cases, if you select the Record and Run only on radio button, the settings may also apply to (limit) the applications that are recognized for Object Spy and other pointing hand operations.
<p>Custom Active Screen Capture Settings Dialog Box (tests only)</p>	<p>Use the Windows section. (Tools > Options > Active Screen pane > Custom Level button)</p> <p>See the section on the Custom Active Screen Capture Settings dialog box in the <i>HP QuickTest Professional User Guide</i>.</p>
<p>Application Area Settings Dialog Box (components only)</p>	<p>Use the Applications pane. (File > Settings > Applications node)</p> <p>See the section on defining application settings for your application area in the <i>HP QuickTest Professional for Business Process Testing User Guide</i>.</p>

This chapter includes:

- ▶ Enabling Communications Between QuickTest Professional and Your Delphi Application on page 180
- ▶ Delphi Add-in Extensibility on page 182

Enabling Communications Between QuickTest Professional and Your Delphi Application

You must use the **MicDelphiAgent.pas** module to enable communications between QuickTest Professional and each Delphi project you want to test.

If your application includes the **TwwDBGrid** from InfoPower, you must also configure support for the grid.

Linking to the MicDelphiAgent.pas Module to Enable Communications

You must perform the following steps for each application that you want to test.

To link to the **MicDelphiAgent.pas** module:

- 1** Add the **<QuickTest Professional Installation folder>\dat\Extensibility\Delphi** folder to your Delphi project search path or copy the contents of the **<QuickTest Professional Installation folder>\dat\Extensibility\Delphi** folder to your project folder.
- 2** Add **MicDelphiAgent** to the **Uses** section of your application's project file (**project.dpr**) as shown in the example below:

```
program flight;
uses
  MicDelphiAgent,
  Forms,
  Windows;
($R*.RES)
begin
  Application.Initialize
  Application.Title :='Flight Reservation';
  Application.Run;
end.
```

3 Compile your Delphi project.

Note: If your application includes the **TwwDBGrid** from InfoPower, you must add support for this grid as described below.

Configuring Support for TwwDBGrid

If your application includes the **TwwDBGrid** from InfoPower, follow the following instructions to enable support for this grid.

- 1 Add **MicWWSupport** to the **Uses** section of your application's project file (**project.dpr**) after **MicDelphiAgent**, as shown in the example below:

```
program flight;
uses
  MicDelphiAgent,
  MicWWSupport,
  Forms,
  Windows;
($R*.RES)
begin
  Application.Initialize
  Application.Title := 'Flight Reservation';
  Application.Run;
end.
```

- 2 Recompile your application.

You are now ready to create and run tests on Delphi applications.

Delphi Add-in Extensibility

QuickTest Professional Delphi Add-in Extensibility enables you to develop support for testing third-party and custom Delphi controls that are not supported out-of-the-box by the QuickTest Professional Delphi Add-in.

If the test object class that QuickTest uses to represent your control does not provide the operations and properties necessary to operate on your control, you can use Delphi Add-in Extensibility to customize this behavior.

- ▶ You can map the control to an existing test object class
- ▶ You can map the control to a new test object class that you create, and design the test object class behavior in Delphi code. You can program how operations are performed on the control, how properties are retrieved, and more.
- ▶ You can also teach QuickTest to treat a control that contains a set of lower-level controls as a single functional control, instead of relating to each lower-level control separately.

To implement Delphi Add-in Extensibility, you need to be familiar with:

- ▶ QuickTest Professional and its Object Model Reference
- ▶ The behavior of the custom control (operations, properties, events)
- ▶ XML (basic knowledge)
- ▶ Delphi programming

Delphi Add-in Extensibility is available as part of the Delphi Add-in and does not require an additional installation.

QuickTest also provides samples of support developed using Delphi Add-in Extensibility, which you can use to gain a better understanding of how to create your own support.

For details on implementing Delphi Add-in Extensibility, see the Delphi Add-in Extensibility Help, available from the QuickTest Professional Extensibility Documentation program group (**Start > Programs > HP QuickTest Professional > Extensibility > Documentation**).

A printer-friendly (PDF) version of the *HP QuickTest Professional Delphi Add-in Extensibility Developer Guide* is available in the **<QuickTest Professional installation folder>\help\Extensibility** folder.

Part V

The Java Add-in

10

Using the Java Add-in

You can use the QuickTest Professional Java Add-in to test Java objects (controls) and applets. You can run steps on Java objects in environments such as Internet Explorer, Mozilla Firefox, Java Web Start, Applet Viewer, and in standalone Java applications.

For details on supported Java toolkits and versions, see the **Java Add-in** section of the *HP QuickTest Professional Product Availability Matrix*, available from the Documentation Library Home page or the root folder of the QuickTest Professional DVD.

The Java Add-in provides customized Java test objects, methods, and properties that can be used when testing objects in Java Add-in applications. For more information, see the **Java** section of the *HP QuickTest Professional Object Model Reference*.

The following table summarizes basic information about the Java Add-in and how it relates to some commonly-used aspects of QuickTest.

Prerequisites	
Opening Your Application	You can open your Java application before or after opening QuickTest.
Add-in Dependencies	See "Understanding Java Add-in Dependencies and Conflicts" on page 191.
General Information	
Checkpoints and Output Values	<ul style="list-style-type: none"> ▶ See the sections describing checkpoints and output values in the <i>HP QuickTest Professional User Guide</i> (for tests), and the <i>HP QuickTest Professional for Business Process Testing User Guide</i> (for components). ▶ See "Creating Checkpoints on SWT-Based Java Tree Objects with Columns" on page 222. ▶ See "Using Text Checkpoints and Text Output Value Steps with Java Objects" on page 223. ▶ See "Supported Checkpoints and Output Values Per Add-in" on page 769.
Extending the Java Add-in	Java Add-in Extensibility (described on page 191) enables you to develop support for testing third-party and custom Java controls that are not supported out-of-the-box by the QuickTest Professional Java Add-in.
Setting Preferences	
Options Dialog Box	Use the Java pane. (Tools > Options > Java node) See "The Options Dialog Box: Java Pane" on page 194.
Record and Run Settings Dialog Box (tests only)	Use the Java tab. (Automation > Record and Run Settings) See "The Record and Run Settings Dialog Box: Java Tab" on page 209.
Test Settings Dialog Box (tests only)	Use the Java pane. (File > Settings > Java node) See "The Settings Dialog Box: Java Pane" on page 204.

Custom Active Screen Capture Settings Dialog Box (tests only)	Use the Java section. (Tools > Options > Active Screen node > Custom Level) See the section on the Custom Active Screen Capture Settings dialog box in the <i>HP QuickTest Professional User Guide</i> .
Application Area Settings Dialog Box (components only)	<ul style="list-style-type: none"> ▶ Use the Java pane. (File > Settings > Java node) ▶ See "The Settings Dialog Box: Java Pane" on page 204. (If an application area is associated with the business component, settings can be modified only in the Application Area Settings dialog box.)

This chapter includes:

- ▶ Considerations for Working with the Java Add-in on page 189
- ▶ Understanding Java Add-in Dependencies and Conflicts on page 191
- ▶ Java Add-in Extensibility on page 191

Considerations for Working with the Java Add-in

When learning objects and running steps on Java applications, consider the following:

- ▶ After installing the Java Add-in, Java applets and applications will always open with Java support active. You can confirm that your Java environment has opened properly by checking the Java console for a message similar to the following confirmation message: "Loading QuickTest Professional Java Support (version x.x.x.x) (<App> version x.x.x.x)." (where <App> is IE, IBM, or SUN).
- ▶ The **Object** property can access only **public** methods and properties. A recommended alternative to using the **Object** property is to extend QuickTest support for the required Java object using QuickTest Java Add-in Extensibility. For more information, see the *HP QuickTest Professional Java Add-in Extensibility Developer Guide*.

- ▶ You cannot add SWT-based JavaMenu 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 an SWT-based JavaMenu 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. Alternatively, you can add a JavaMenu object using the **Navigate and Learn** option in the Object Repository Manager. For more information, see the section on adding test objects using the Navigate and Learn option in the *HP QuickTest Professional User Guide*.
- ▶ In earlier releases of QuickTest, Java identification properties were not case-sensitive. If you learned a test object in a QuickTest version earlier than 11.00, you need to re-learn the object with properties that are case-sensitive by performing an **Update Run** (using the **Update test object descriptions** option). For more information, see the section on Updating Test Object Descriptions in the *HP QuickTest Professional User Guide*.
- ▶ In QuickTest, table data is always loaded from the application itself, even if the Active Screen contains an image of the table. For this reason, you must first open the table in the application before creating a table checkpoint in a test.
 - ▶ In some cases you may have to scroll to the last row of the table to make sure that all the data is loaded.
 - ▶ It is not necessary to open the table in your application to edit an existing table checkpoint.
- ▶ If you load or unload an add-in that is displayed as a child of the Java add-in in the Add-in Manager, only applications that are opened after loading or unloading the add-in are affected.
- ▶ For more information on QuickTest functionality, see the *HP QuickTest Professional User Guide* and the *HP QuickTest Professional for Business Process Testing User Guide*.

Understanding Java Add-in Dependencies and Conflicts

The QuickTest Professional Java Add-in can be installed and run together with any other QuickTest Professional add-in. When testing Java applets in a Web browser, if your tests include operations on Web test objects, you must load the Web Add-in as well as the Java Add-in and use the Web tab of the Record and Run Settings dialog box to specify your record and run preferences.

Java Add-in Extensibility

QuickTest Professional Java Add-in Extensibility enables you to develop support for testing third-party and custom Java controls that are not supported out-of-the-box by the QuickTest Professional Java Add-in.

If the test object class that QuickTest uses to represent a control does not provide the operations and properties necessary to operate on your control, you can use Java Add-in Extensibility to customize this behavior.

- ▶ You can map a custom control to an existing test object class, or to a new test object class that you define
- ▶ You can design and customize the behavior of the test object classes by developing custom Java support classes. You can program how operations are performed on the control, how properties are retrieved, and more.
- ▶ You can also teach QuickTest to treat a control that contains a set of lower-level controls as a single functional control, instead of relating to each lower-level control separately.

To implement Java Add-in Extensibility, you need to be familiar with:

- ▶ QuickTest Professional and its Object Model Reference
- ▶ The behavior of the custom control (operations, properties, events)
- ▶ XML (basic knowledge)
- ▶ Java programming

You can install the Java Add-in Extensibility SDK from the **Add-in Extensibility and Web 2.0 Toolkits** option in the QuickTest Professional setup program.

The SDK also includes:

- ▶ A plug-in for the Eclipse Java development environment, which provides wizards and commands that help you create and edit the support that you develop.
- ▶ Samples of support developed using Java Add-in Extensibility, which you can use to gain a better understanding of how to create your own support.

For details on installing and implementing Java Add-in Extensibility, see the Java Add-in Extensibility Help, available from the QuickTest Professional Extensibility Documentation program group (**Start > Programs > HP QuickTest Professional > Extensibility > Documentation**).

A printer-friendly (PDF) version of the *HP QuickTest Professional Java Add-in Extensibility Developer Guide* is available in the **<QuickTest Professional installation folder>\help\Extensibility** folder.

11

Creating and Running Tests on Java Objects

This chapter explains how to use QuickTest to set testing preferences and to record and run steps on Java applets and applications. The chapter assumes basic knowledge of QuickTest features and capabilities. For more information on working with QuickTest, see the *HP QuickTest Professional User Guide*.

Note: Some of the features described in this chapter are relevant only for tests (and scripted components). For information on the features that are available when working with business components, see the *HP QuickTest Professional for Business Process Testing User Guide*.

This chapter includes:

- ▶ Defining Java Testing Options on page 194
- ▶ Defining Java Settings for Individual Tests and Components on page 203
- ▶ Defining Java Record and Run Options for Tests on page 209
- ▶ Defining Application Details Environment Variables for Tests on page 214
- ▶ Optimizing Settings for Other Record and Run Settings Dialog Box Tabs on page 215
- ▶ Recording Tests and Components on Java Objects on page 216

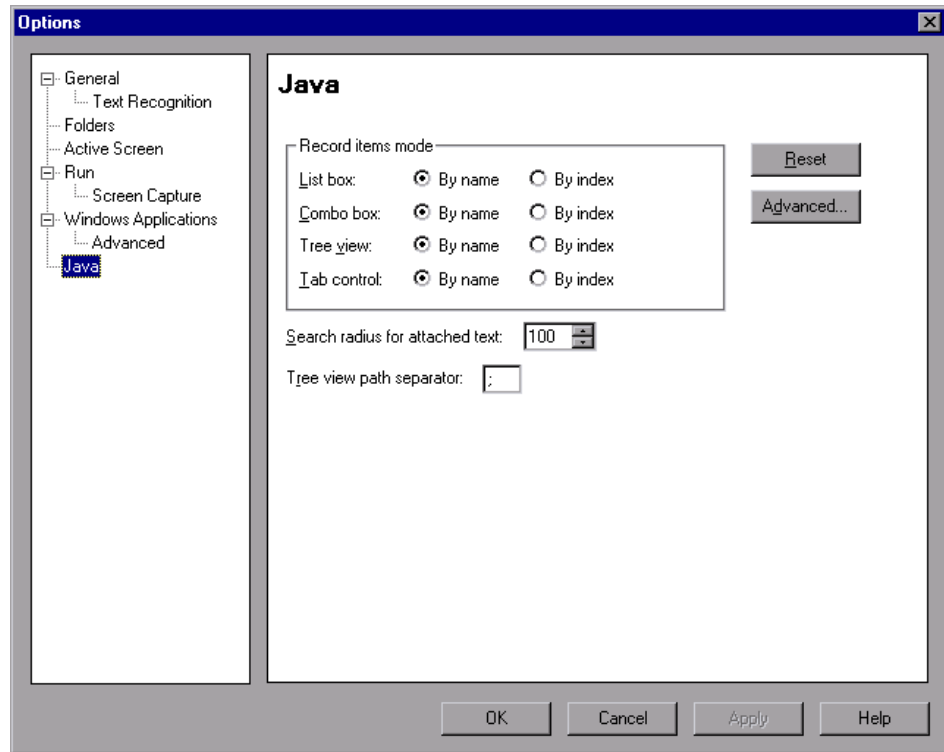
Defining Java Testing Options

You can use the Java pane of the Options dialog box to set QuickTest record and run options on Java applets or applications. You can also open the Advanced Java Options dialog box that enables you to set table record mode preferences, enable text retrieval for checkpoints and output values, and specify lists of controls.

The Options Dialog Box: Java Pane

Description	Enables you to configure how QuickTest records and runs tests on Java applets or applications.
How to Access	Tools menu > Options item > Java node
Important Information	The Java pane is available only when the Java or Oracle Add-ins are installed and loaded. If you are using the Oracle Add-in, and you add steps to your test for Java objects within your Oracle application, the options in this pane are relevant for the Java steps in your test.
Learn More	<p>Conceptual overview: "Using the Java Add-in" on page 187.</p> <p>Additional related topics:</p> <ul style="list-style-type: none"> ▶ "Additional References" on page 198 ▶ See the <i>HP QuickTest Professional User Guide</i> for more information on the Options dialog box

Below is an image of the Java Pane in the Options dialog box:



Options Dialog Box: Java Pane Options

Option	Description
Record items mode	<p>Determines how QuickTest records operations on items in List box, Combo box, Tree view, and Tab control objects. Select one of the following options for each object:</p> <ul style="list-style-type: none"> ▶ By name. (Default) Records operations on an item within the object (for example, selected list item or tab) according to the item's name. ▶ By index. Records operations on an item within the object (for example, selected list item or tab) according to the item's position within the Java object. <p>Notes:</p> <ul style="list-style-type: none"> ▶ If you select the By index option for Tree view, do not specify "#" as the default separator in the Tree view path separator option below. ▶ This option corresponds to the <code>Setting.Java("record_by_num")</code> variable.
Search radius for attached text	<p>Sets the maximum distance in pixels to search for attached text.</p> <p>Default value: 100</p> <p>Notes:</p> <ul style="list-style-type: none"> ▶ This option is relevant only when the label identification property is unavailable. ▶ This option corresponds to the <code>Setting.Java("max_text_distance")</code> variable.

Option	Description
Tree view path separator	<p>Specifies the default separator used to separate entries in a path to a node of a Tree view control.</p> <ul style="list-style-type: none"> ▶ Default value: ; ▶ Possible value: One or more single-character separators <p>Notes:</p> <ul style="list-style-type: none"> ▶ If you enter more than one character, QuickTest treats each of the characters as a separator (but not both of them in sequence). If a path contains two consecutive separators, QuickTest interprets the path as if it contains a node with no name between the two separators. For example, if you specify %\$ for this option and a particular path contains MyNode%\$MySubNode, then QuickTest treats the % character as a separator for a node with no name, and the \$ character as the separator for an additional node named MySubNode. ▶ If you select the By index option for Tree View in the Record Items mode area above, do not specify "#" as the default separator. ▶ This option corresponds to the <code>Setting.Java("treeview_path_separator")</code> variable.
Reset	Resets the Java test settings to their default values.
Advanced	Opens the Advanced Java Options dialog box. For more information, see "The Advanced Java Options Dialog Box" on page 198.

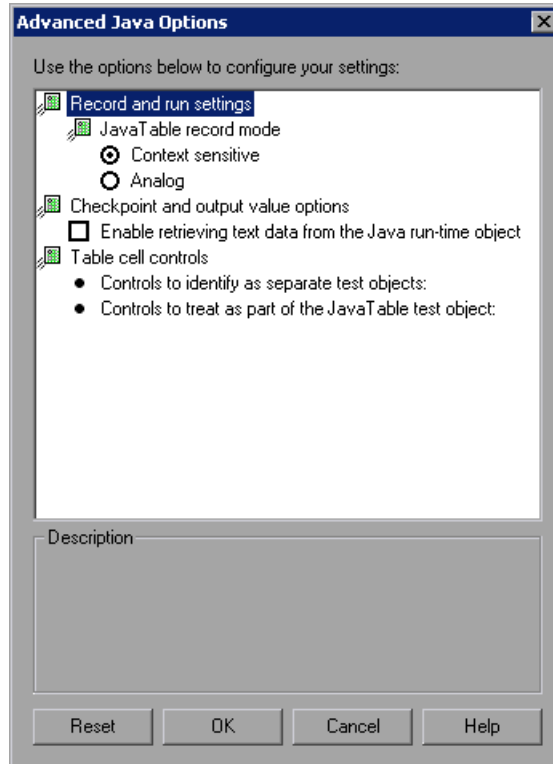
Additional References

Related User Interface Topics	<ul style="list-style-type: none"> ▶ "The Advanced Java Options Dialog Box" on page 198. ▶ "The Settings Dialog Box: Java Pane" on page 204.
Related Tasks	<ul style="list-style-type: none"> ▶ "Defining Java Record and Run Options for Tests" on page 209. ▶ "Defining Application Details Environment Variables for Tests" on page 214. ▶ "Optimizing Settings for Other Record and Run Settings Dialog Box Tabs" on page 215.
Related Concepts	"Recording Tests and Components on Java Objects" on page 216.

The Advanced Java Options Dialog Box

Description	Enables you to specify additional Java options. You can configure table record mode preferences, enable retrieving text information from the run-time object for checkpoints and output values (tests only), and specify lists of controls.
How to Access	Tools menu > Options item > Java node > Advanced button
Important Information	If you are using the Oracle Add-in, and you add steps to your test for Java objects within your Oracle application, the options in this dialog box are relevant for the Java steps in your test.
Learn More	<p>Conceptual overview: "Using the Java Add-in" on page 187</p> <p>Additional related topics: "Additional References" on page 202</p>

Below is an image of the Advanced Java Options dialog box:



Advanced Java Options Dialog Box Options

Option	Description
JavaTable record mode	<p>Sets the record mode for table objects. Select one of the following modes:</p> <ul style="list-style-type: none"> ▶ Context Sensitive. (Default) Records operations on table objects in context-sensitive mode: SetCellData, SelectRow, and so on. ▶ Analog. Records only low-level (analog) table methods: ClickCell, DoubleClickCell, and Drag. <p>Note: This option corresponds to the <code>Setting.Java("table_record_mode")</code> variable.</p>

Option	Description
<p>Checkpoint and output value options</p>	<p>Sets preferences for checkpoint and output value steps on Java objects. It contains the following option:</p> <p>Enable retrieving text data from the Java run-time object: Enables QuickTest to retrieve text information from the Java objects in the application for checkpoints and output value steps. This option is not relevant if QuickTest is configured to use the OCR mechanism for text recognition (Tools > Options > General > Text Recognition pane).</p> <p>Notes:</p> <ul style="list-style-type: none"> ▶ Retrieving text information from the run-time object is supported only for Java objects that meet very specific criteria. Therefore, this option is disabled by default. ▶ Text checkpoints and output values are not supported for business components.

Option	Description
Table cell controls	<p>Sets preferences for the way that QuickTest identifies controls inside table cells. It includes the following options:</p> <ul style="list-style-type: none"> ▶ Controls to identify as separate test objects: Specifies the list of controls that you want QuickTest to identify as separate test objects and not as part of a JavaTable object. Use this option to access methods that are specific to the object type or to otherwise improve the functionality of steps that QuickTest would normally record and run as operations on a JavaTable object. <p>Notes:</p> <ul style="list-style-type: none"> ▶ This option is relevant for JTable Swing toolkit tables. ▶ Specify control class names separated by a space, tab, newline, or return character. Values are case sensitive. ▶ This option corresponds to the <code>Setting.Java("table_internal_editors_list")</code> variable. <ul style="list-style-type: none"> ▶ Controls to treat as part of the JavaTable test object: Specifies the list of controls for which you want QuickTest to record and run JavaTable operations. Use this option to record and run JavaTable operations (such as SetCellData and Select) on controls that QuickTest would normally treat as separate test objects. <p>Notes:</p> <ul style="list-style-type: none"> ▶ This option is relevant for JTable Swing toolkit tables. ▶ Specify editor class names separated by a space, tab, newline, or return character. Values are case sensitive. ▶ This option corresponds to the <code>Setting.Java("table_external_editors_list")</code> variable. <p>See also:</p> <ul style="list-style-type: none"> ▶ "Modifying Table Cell Control Options" on page 202 ▶ "Finding the Toolkit Class of a JTable Cell Editor" on page 219

Additional References

Related User Interface Topics	<ul style="list-style-type: none">▶ "The Options Dialog Box: Java Pane" on page 194▶ "The Settings Dialog Box: Java Pane" on page 204
Related Tasks	"Defining Java Record and Run Options for Tests" on page 209
Related Concepts	<ul style="list-style-type: none">▶ "Recording Tests and Components on Java Objects" on page 216▶ "Using Text Checkpoints and Text Output Value Steps with Java Objects" on page 223▶ "Recording on Table Objects" on page 218

Modifying Table Cell Control Options

In the Advanced Java Options dialog box, you can specify a list of table cell controls that you want QuickTest to identify as separate test objects. You can also specify a list of table cell controls for which you want QuickTest to record and run JavaTable operations.

Notes:

- ▶ Any changes you make are not applied to the currently open test or component. To apply your changes, close your test or component and reopen it.
 - ▶ You can restore the default settings in the Advanced Java Options dialog box by clicking the **Reset** button.
-

To modify one of the Table Cell Controls options:

- 1** In the Advanced Options dialog box, click the relevant option once to highlight it.
- 2** Click the option again or press F2 to open an edit box in which you can add or modify a list of controls.
- 3** Change the value as necessary.

Note: Specify editor class names separated by a space, tab, newline, or return character. Values are case sensitive.

- 4** When you finish editing the value, click another location in the dialog box to set the value.
- 5** When you finish making all of the required changes in this dialog box, click **OK** and close the dialog box.

Defining Java Settings for Individual Tests and Components

You set Java test or component variables using one of the following:

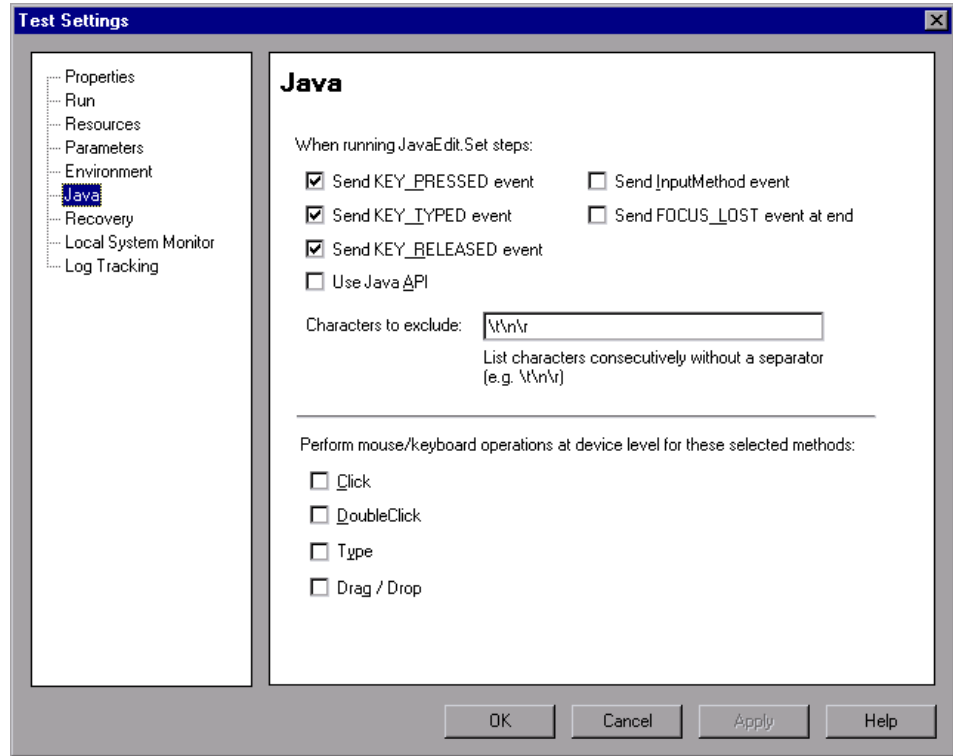
- The Java pane of the Test Settings dialog box.
- The Java pane of the Business Component Settings dialog box.
- The Java pane of the Application Area Settings dialog box.

The Settings Dialog Box: Java Pane

The options shown in the Java pane are the same in the Application Area Settings dialog box, the Business Component Settings dialog box, and the Test Settings dialog box.

Description	Enables you to set Java test or component variables.
How to Access	File menu > Settings item > Java node
Important Information	<ul style="list-style-type: none"> ▶ The Java pane is available only when the Java or Oracle Add-ins are installed and loaded. If you are using the Oracle Add-in, and you add steps to your test for Java objects within your Oracle application, the options in this pane are relevant for the Java steps in your test. ▶ If an application area is associated with the business component, the Business Component Settings dialog box displays in read-only the options selected in the Application Area Settings dialog box.
Learn More	<p>Conceptual overview: "Using the Java Add-in" on page 187</p> <p>Additional related topics:</p> <ul style="list-style-type: none"> ▶ "Additional References" on page 208 ▶ For more information on the Test Settings dialog box, see the <i>HP QuickTest Professional User Guide</i> ▶ For more information on the Settings dialog box, see the <i>HP QuickTest Professional for Business Process Testing User Guide</i>

Below is an image of the Java pane:



Settings: Java Pane Options

Option	Description
When running JavaEdit.Set steps	<p>Specifies how operations are performed on edit boxes during a test run. It is recommended not to modify these settings unless you fully understand Java key events and input methods, as well as the implications of sending or not sending these events. Note that JavaEdit.Set steps may fail during a run session if an incorrect value is used for these settings. You can set one or more of the following options:</p> <ul style="list-style-type: none"> ▶ Send KEY_PRESSED event. Sends a KEY_PRESSED event to the object for every character from the input string. (Selected by default.) This setting corresponds to the P value of the <code>Setting.Java("edit_replay_mode")</code> variable. ▶ Send KEY_TYPED event. Sends a KEY_TYPED event to the object for every character from the input string. (Selected by default.) This setting corresponds to the T value of the <code>Setting.Java("edit_replay_mode")</code> variable. ▶ Send KEY_RELEASED event. Sends a KEY_RELEASED event to the object for every character from the input string. (Selected by default.) This setting corresponds to the R value of the <code>Setting.Java("edit_replay_mode")</code> variable. ▶ Use Java API. Calls the <code>setValue()</code> method to set a value of the edit object. This setting corresponds to the S value of the <code>Setting.Java("edit_replay_mode")</code> variable. ▶ Send InputMethod event. Sends an <code>InputMethod</code> event to the object for every character from the input string. This event is used with Unicode applications (for example, for some non-English applications). This setting corresponds to the I value of the <code>Setting.Java("edit_replay_mode")</code> variable. ▶ Send FOCUS_LOST event at end. Generates a <code>FOCUS_LOST</code> event after running the step. This setting corresponds to the F value of the <code>Setting.Java("edit_replay_mode")</code> variable.

Option	Description
Characters to exclude	<p>Instructs QuickTest to ignore the specified characters during a run session. List characters consecutively, without a separator.</p> <p>Default value: \t\n\r</p> <p>Notes:</p> <ul style="list-style-type: none"> ▶ This option is relevant only if the Use Java API check box is selected in the upper section of this dialog box, or if the value of the <code>Setting.Java("edit_replay_mode")</code> variable is set to S. ▶ This setting corresponds to the <code>Setting.Java("exclude_control_chars")</code> variable.
Perform mouse/ keyboard operations at device level for these selected methods	<p>By default, QuickTest performs mouse operations at the context-sensitive level. You can use this option to select specific operations to perform using device-level replay. Device-level replay simulates mouse or key operations exactly as if they occur on the mouse or keyboard drivers. When a mouse action is simulated on device replay, the mouse pointer moves on the screen to the point where the action is to be performed during the run session. You can select from the following mouse and keyboard methods:</p> <ul style="list-style-type: none"> ▶ Click ▶ DoubleClick ▶ Type ▶ Drag / Drop <p>Default value: All check boxes are cleared.</p> <p>This option corresponds to the <code>Setting.Java("device_replay_mode")</code> variable.</p>

Additional References

Related User Interface Topics	"The Options Dialog Box: Java Pane" on page 194
Related Tasks	<ul style="list-style-type: none">➤ "Defining Java Record and Run Options for Tests" on page 209➤ "Defining Application Details Environment Variables for Tests" on page 214➤ "Optimizing Settings for Other Record and Run Settings Dialog Box Tabs" on page 215
Related Concepts	"Recording Tests and Components on Java Objects" on page 216
Other Related Information	<ul style="list-style-type: none">➤ For more information on JFC or AWT-based Java key events and input methods, see Java documentation at http://java.sun.com.➤ For more information on SWT-based Java key events, see Java documentation at http://www.eclipse.org.

Defining Java Record and Run Options for Tests

You can use the Java tab of the Record and Run Settings dialog box to instruct QuickTest to open your Java applet or application each time you begin a recording session, or to instruct QuickTest to record on any open Java application.

Note: Components do not require specific record and run settings to work with Java applets and applications. To record a component, you need to first open the Java applet or application manually. Alternatively, you can include steps in your component that connect to the Java applet or application, for example, you can include a step that contains the OpenApp operation.

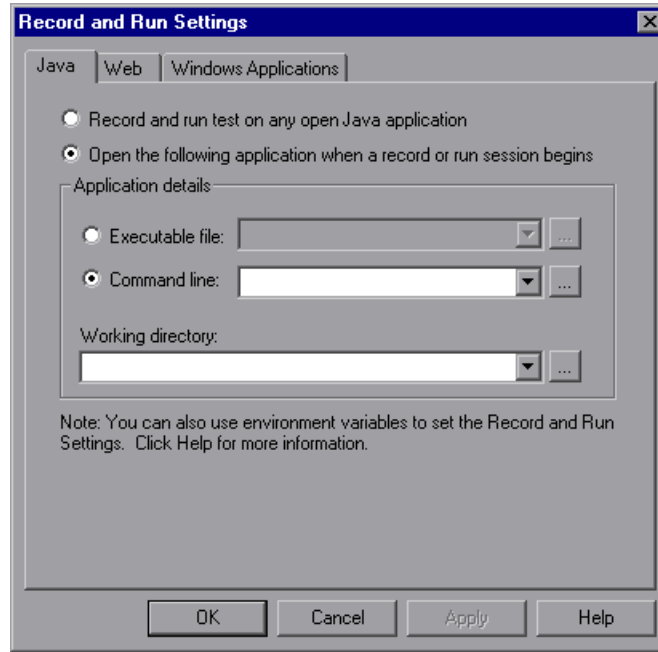
When you begin recording a new component, the Applications dialog box opens (unless you previously specified a Windows environment in the Application Area Settings or Business Component Settings dialog box). Click OK in the dialog box without making modifications to begin recording. For more information on the Applications tab and Applications dialog box, see "Setting Windows Applications Record and Run Options" on page 89.

The Record and Run Settings Dialog Box: Java Tab

Description	Enables you to instruct QuickTest to open your Java applet or application each time you begin a recording session, or to instruct QuickTest to record on any open Java application.
How to Access	<ul style="list-style-type: none"> ▶ Automation menu > Record and Run Settings item > Java tab) ▶ If you do not modify the record and run settings before you begin recording, the Record and Run Settings dialog box opens automatically when you begin recording a new test (by clicking Record or choosing Automation > Record).

<p>Important Information</p>	<ul style="list-style-type: none"> ▶ When testing Java applets in a Web browser, you must load both the Web Add-in and the Java Add-in. In this case, you use the Web tab of the Record and Run Settings dialog box to specify your record and run preferences. ▶ The Java and Web tabs in the Record and Run Settings dialog box are available only when the corresponding add-ins are installed and loaded. If other add-ins are loaded, the corresponding tabs (if any) are also displayed. ▶ When you run a test, or if you begin a new recording session on an existing test, QuickTest automatically uses the existing record and run settings for the test and does not open the Record and Run Settings dialog box. However, it is important to confirm that the options in the Record and Run Settings Java tab are appropriate for the first step of your test before running it because you (or someone else) may have modified the Record and Run Settings dialog box manually in a prior record session.
<p>Learn More</p>	<p>Conceptual overview: "Using the Java Add-in" on page 187</p> <p>Additional related topics:</p> <ul style="list-style-type: none"> ▶ "Additional References" on page 213 ▶ For more information on the Web tab, see "Testing Web-Based Applications" on page 45.

Below is an image of the Java tab:



Java Tab Options

Option	Description
Record and run test on any open Java application	Instructs QuickTest to record and run the test on any open Java application or applet.

Option	Description
<p>Open the following application when a record or run session begins</p>	<p>Instructs QuickTest to open a new Java application or applet using the specified application details.</p> <p>Notes:</p> <ul style="list-style-type: none"> ▶ This setting controls only which Java application, if any, is opened at the beginning of a record or run session. It does not affect the applications that QuickTest recognizes. Even if this radio button is selected and no application is specified, QuickTest can still record, recognize, and run on any open Java application. ▶ When working with a Java applet inside a browser, use the Web tab of the Record and Run Settings dialog box to open the URL containing the applet.
<p>Application details</p>	<p>Defines details of the Java application on which to run the test:</p> <ul style="list-style-type: none"> ▶ Executable file. Instructs QuickTest to open the specified executable or batch file. ▶ Command line. Instructs QuickTest to open the application from the specified command line. ▶ Working directory. Instructs QuickTest to run the specified executable file or command line from the specified directory. Make sure you specify the full directory path, for example, C:\Program Files\Java\jre1.6.0\bin. <p>Note:</p> <ul style="list-style-type: none"> ▶ If you define values for the EXEPATH_ENV, CMDLINE_ENV, and/or WORKDIR_ENV test environment variables, these values override the values in the Executable file, Command line, and Working Directory boxes of the Java tab during a run session. For more information, see "Defining Application Details Environment Variables for Tests" on page 214. ▶ Always enter a value in the Working directory field, otherwise QuickTest is unable to open your Java application.

Additional References

Related User Interface Topics	<ul style="list-style-type: none"> ➤ "The Options Dialog Box: Java Pane" on page 194 ➤ "The Advanced Java Options Dialog Box" on page 198
Related Tasks	<ul style="list-style-type: none"> ➤ "Defining Application Details Environment Variables for Tests" on page 214 ➤ "Optimizing Settings for Other Record and Run Settings Dialog Box Tabs" on page 215 ➤ "Using the Record and Run Settings Dialog Box" on page 38.
Related Concepts	<ul style="list-style-type: none"> ➤ "Recording Tests and Components on Java Objects" on page 216 ➤ For more information on the Record and Run Settings dialog box, see the <i>HP QuickTest Professional User Guide</i>.

Defining Application Details Environment Variables for Tests

You can use application details environment variables to specify the applications you want to use for recording and running your test. If you define any of these application details environment variables, they override the values in the **Executable file**, **Command line**, and **Working directory** boxes in the Java tab of the Record and Run Settings dialog box. For more information, see "Defining Java Record and Run Options for Tests" on page 209.

Use the variable names listed in the table below to define Java application details:

Option	Variable Name	Description
Executable file	EXEPATH_ENV	The executable file or a batch file to open.
Command line	CMDLINE_ENV	The command line to use to open the file.
Working directory	WORKDIR_ENV	The folder to which the specified command line or executable file refers.

For more information on defining and working with environment variables, see the *HP QuickTest Professional User Guide*.

Optimizing Settings for Other Record and Run Settings Dialog Box Tabs

In addition to setting the appropriate settings in the Java tab (or Web tab for applets in browsers), you should confirm that the other tabs in the dialog box have the appropriate settings for your test. The following settings are recommended:

- ▶ **Windows Applications tab.** Select **Record and run only on** and confirm that all check boxes are cleared.
- ▶ **Other tabs.** (If displayed.) Select the option to record and run on any open application (upper radio button of each tab).

While these settings do not directly affect your record or run sessions when working with Java applets and applications, these settings prevent you from inadvertently recording operations performed on Windows applications (such as e-mail) during your recording session. These settings also prevent QuickTest from opening unnecessary applications when you record or run tests on Java applets and applications.

For more information on the Record and Run Settings dialog box, see "The Record and Run Settings Dialog Box: Overview" on page 37.

Recording Tests and Components on Java Objects

When you record an operation on an applet, application, or Java object, QuickTest records the appropriate object icon next to the step in the Keyword View (for tests and components) and adds the relevant statement in the Expert View (for tests only).

If you try to record an operation on an unsupported or custom Java object, QuickTest records a generic **JavaObject.Click** statement that includes the coordinates of the click and the mouse button (that is, left or right) that was clicked. You can create support for your custom object using the QuickTest Professional Java Add-in Extensibility. For more information, see the *HP QuickTest Professional Java Add-in Extensibility Developer Guide*.

Note: The way in which QuickTest records operations depends on the type of JTable cell editor in the table cell. For more information, see "Recording on Table Objects" on page 218.

The QuickTest recorded hierarchy is composed of two or three levels of Java test objects. The top level is represented by the **JavaApplet**, **JavaDialog**, or **JavaWindow** object, as appropriate. The actual object on which you performed an operation may be recorded as a second or third level object. If the object is located directly in the top level object, it is recorded as a second level object (for example, **JavaApplet.JavaButton**). If a **JavaDialog** or **JavaInternalFrame** exists at the second level, then the object on which you performed the operation is recorded as a third level object (for example, **JavaWindow.JavaDialog.JavaButton**).

When testing applets in a browser, the two- or three-level hierarchy is recorded within the standard Web object hierarchy (for example, **Browser.Page.JavaApplet.JavaTestObject.SubJavaTestObject**).

Even though the object on which you record may be embedded in several levels of objects, the recorded hierarchy does not include these objects. For example, if the **JavaList** object on which you record is actually contained in several **JPanel** objects, which are all contained in a **JavaWindow**, the recorded hierarchy is only **JavaWindow.JavaList**.

For example, in a test, if you record a click on a Java check box, the Keyword View may be displayed as follows:

▼ Action1			
▼ Microsoft Internet Explorer			
msctls_statusbar32	Click	776,2	Click the "msctls_statusbar32" status bar.
▼ Periodic			
Toggle	Set	"ON"	Set the state of the "Toggle" check box to "ON".

QuickTest records this step in the Expert View as:

```
Window("Microsoft Internet Explorer").JavaApplet("Periodic").
  JavaCheckBox("Toggle").Set "ON"
```

In a component, if you record a click on this same Java check box, the Keyword View would displayed as follows:

Toggle	Set	"ON"	Set the state of the "Toggle" check box to "ON".
--------	-----	------	--

You can view the recorded hierarchy of a test object in the object repository.

You can access the full hierarchy of an object when using the pointing hand mechanism in the Step Generator (tests only), when inserting a checkpoint or output value step while recording, or when using the Object Spy.

For more information on recording tests and components on Java Objects, see:

- ▶ "Recording on Table Objects" on page 218
- ▶ "Creating Checkpoints on SWT-Based Java Tree Objects with Columns" on page 222
- ▶ "Using Text Checkpoints and Text Output Value Steps with Java Objects" on page 223
- ▶ "Viewing the Full Object Hierarchy" on page 224

Recording on Table Objects

When you record an operation that changes the data in a cell of a Java table object, QuickTest generally records the end result of the data in the cell in the form of a `JavaTable.SetCellData` statement. (`JavaTable.SetCellData` is not used when the **JavaTable record mode** is set to **Analog**. For more information on JavaTable record mode, see "The Advanced Java Options Dialog Box" on page 198.)

Recording on Standard Cell Editors in Swing JTable Tables

The QuickTest Professional Java Add-in also provides built-in support for several standard Swing JTable cell editor types. This means that by default, QuickTest records operations on these standard cell editors in the same way as other table objects, using `SetCellData` statements.

Recording on Custom Cell Editors in Swing JTable Tables

When a JTable contains a custom (non-standard) cell editor, the default `SetCellData` statement cannot be recorded. For example, if a cell contains both a check box and a button that opens a dialog box, then a `SetCellData` statement may not always provide an accurate description of the operations performed inside the cell.

If you record an operation on a custom cell editor, QuickTest records a statement that reflects the operation you performed on the object inside of the cell. For example, if the cell editor contains a custom check box, QuickTest might record the following statement:

```
Browser("Periodic").Page("Periodic").JavaWindow("CoolJava").JavaDialog("Set  
Options").JavaCheckBox("MyCheckBox").Set "ON"
```

instead of:

```
Browser("Periodic").Page("Periodic").JavaWindow("CoolJava").JavaDialog("Set  
Options").JavaTable("MyTable").SetCellData "ON"
```

Modifying the Default JTable Recording Behavior (Advanced)

In most cases, the default recording behavior for JTables (described in the preceding sections) works well and maximizes the readability of your test. However, if you are not satisfied with the value that QuickTest records for the SetCellData statement of a particular editor, you can set that editor to be recorded, like a custom cell editor, in terms of the operation performed on the object inside the cell.

To do this, use the **Table cell controls > Controls to identify as separate test objects** option in the Advanced Java Options dialog box and specify specific cell editor types that should always be treated as separate objects, and not as part of a JavaTable object. Alternatively, use a Setting.Java ("table_internal_editors_list") statement. For more information, see "The Advanced Java Options Dialog Box" on page 198, and the *HP QuickTest Professional Object Model Reference*.

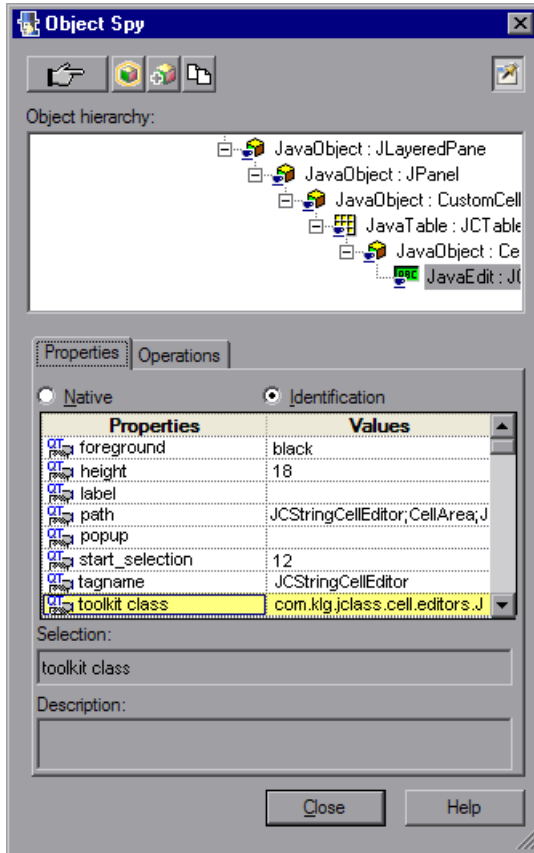
Finding the Toolkit Class of a JTable Cell Editor

If you do not know the value of the toolkit class for an editor for use with the **table_external_editors_list** variable, you can find it either by using the Object Spy, by running a short test in QuickTest to retrieve the value, or by creating a user-defined function and inserting it as a step.

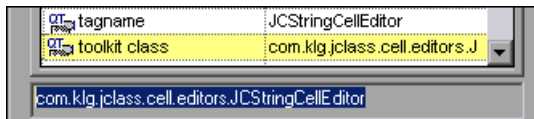
To find the toolkit class of a JTable cell editor using the Object Spy:

- 1** Open the table and activate a cell in the cell editor column. For example, make sure the cursor is blinking inside an edit field or display the drop-down list of a combo box.
- 2** With the appropriate cell activated, use the Object Spy to point to the active cell. For information on using the Object Spy, see the *HP QuickTest Professional User Guide*.

- 3 Make sure the Properties tab of the Object Spy is displayed and select the **Identification** radio button.



- 4 In the **Properties** column, scroll to **toolkit class**.
- 5 In the **Values** column, select the value of the **toolkit class**. The value is displayed in the box below the Properties tab.



- 6 Copy and paste the value from the Object Spy to the **Table cell controls > Controls to identify as separate test objects** option or your `Setting.Java("table_internal_editors_list")` statement.

Finding the Toolkit Class of a JTable Editor by Running a QuickTest Script

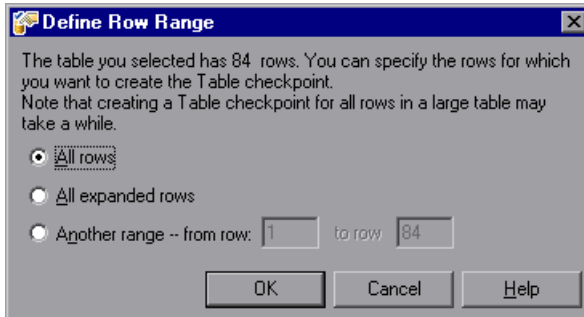
For some cell editors, it is difficult or impossible to capture an activated cell with the Object Spy because the cell does not stay activated for a long enough period of time. For example, after a check box is selected or cleared, the cell is no longer active. If you need to find the toolkit class value to use for these types of cell editors, you can run a short test in QuickTest to retrieve the value. If you are working with components, you can create a user-defined function and insert it as a step.

You can insert steps similar to the following example:

```
' Sample test to retrieve the toolkit class of a table cell editor
' that cannot be made continuously active
Set table = JavaWindow("TableDemo").JavaTable("Left table").Object
Set jTableCS = table.mic_get_supp_class()
Set comp = jTableCS.getComponentAt(table, 0, 6) 'row 0, col 6
MsgBox comp.getClass().getName()
' Set the value of TABLE_EXTERNAL_EDITORS_LIST
Setting.Java("TABLE_EXTERNAL_EDITORS_LIST") =
comp.getClass().getName()
```

Creating Checkpoints on SWT-Based Java Tree Objects with Columns

When working with tests, if you create a checkpoint on an SWT-based Java tree with columns, a table checkpoint is created. When you create the checkpoint, the Define Row Range dialog box opens, enabling you to select the range of rows you want to include in your checkpoint.



You can include:

- ▶ **All rows.** Includes all of the rows in the tree. Note that capturing all of the data for large tree objects may take some time.
- ▶ **All expanded Rows.** Includes all of the expanded rows in the table, even if they are not visible on the screen.
- ▶ **Another range -- from row X to row Y.** You can specify any row range between 1 and the number of rows listed in the table.

The Table Checkpoint Properties dialog box displays the range you select, and also enables you to modify this range after the checkpoint is created.

For more information on table checkpoints, see the *HP QuickTest Professional User Guide*.

Using Text Checkpoints and Text Output Value Steps with Java Objects

When working with tests, you can use checkpoints or output values to check that text in your Java application or applet displays correctly. Similar to many other supported environments, it is recommended to retrieve and check text from your Java applet or application by inserting a standard checkpoint or output value for the object containing the desired text, and selecting to check or output its **text** (or similar) identification property (for example, **text**, **attached text**, or **label**).

If the object you want to work with does not have an appropriate identification property, or, if for any other reason, the above recommendation does not answer your needs (for example, the text before or after the selected text is important), you can consider inserting a QuickTest text checkpoint or text output value step for a Java object if it meets the following criteria:

- ▶ The object must draw the text itself (and not delegate the drawing task to the underlying operating system, as is the case with most AWT components).
- ▶ The object must draw text by overriding the `paint()` method and calling the **standard `graphics.drawString()`** method to draw text. For example, the object cannot use special drawing methods for writing text, such as using a method that can draw oval circles to draw the letter O.
- ▶ The object cannot use the **double (image) buffering** drawing technique.

Note: Because many Java objects do not answer these criteria, the text checkpoint and text output mechanism for Java objects is disabled by default. You can enable it in the Advanced Java Options dialog box. For more information, see "The Advanced Java Options Dialog Box" on page 198.

Viewing the Full Object Hierarchy

The Java Add-in enables you to view the full object hierarchy of each of the objects in your application in the Object Spy and Object Selection dialog boxes. In contrast to the recorded object hierarchy, the full object hierarchy shows you all of the parent objects associated with the clicked locations and, in some cases, the child objects of the clicked object.

The full object hierarchy enables you to view associated operations and properties of non-recorded objects in the Object Spy. When working with tests, you can also access non-recorded objects from the Object Selection dialog box that opens when using the Step Generator (tests only) or when inserting a checkpoint or output value step during a recording session.

The Object Spy and Object Selection dialog boxes enable you to view details, insert statements, or perform operations even for elements of an object (class components) that are not recorded, such as **java.awt.Component**. For example, you can access the edit box, drop-down list, and button elements of a combo box.

For more information on the Object Spy and Object Selection dialog boxes, see the *HP QuickTest Professional User Guide*.

12

Using Advanced Java Test Object Methods

Java test object classes include test object methods that you can use in your tests to enhance the interaction between QuickTest and the application being tested.

This chapter includes:

- Creating Objects in Your Applet or Application (Advanced) on page 225
- Working with Static Members on page 226
- Firing Java Events on page 227

Creating Objects in Your Applet or Application (Advanced)

You can use the **CreateObject** method to create an instance of any Java object within your applet or application. The **CreateObject** method returns an object reference to the newly created Java object. For information on the syntax of this method, see the **Java** section of the *HP QuickTest Professional Object Model Reference*.

You can activate the methods of an object you create in the same way as you would activate the methods of any returned object from a prior call. Because the **CreateObject** method returns an object reference, there is no need to use the **Object** property when activating methods of the created object.

For example, you can use the **CreateObject** method to create a rectangle object. The return value is an object reference.

```
Set Rect =  
Browser("Periodic").Page("Periodic").JavaApplet("Periodic").JavaObject  
("Panel").CreateObject ("java.awt.Rectangle", 10, 20)
```

Note: The **CreateObject** method can be performed on any Java test object. The class loader of the Java test object on which the **CreateObject** method is performed is used to load the class of the newly created Java object.

It is recommended to use the **CreateObject** method on a Java test object from the same toolkit as the object you want to create. For example, to create a Swing/JFC object, use the **CreateObject** method on an existing Swing/JFC Java test object.

Working with Static Members

You can invoke any static method, or you can set or retrieve the value of any static property of a Java class using the **GetStatics** method. For information on the syntax of this method, see the **Java** section of the *HP QuickTest Professional Object Model Reference*.

GetStatics returns a reference to an object that can access static members of the specified class. The class loader of the Java test object on which the **GetStatics** method is performed is used to load the class specified as a parameter of the **GetStatics** method.

For example, to invoke the **gc** method of **class.java.lang.System**, which runs the garbage collector on the application, you can insert a statement similar to the following:

```
Browser("Browser").Page("Page").JavaApplet("mybuttonapplet.htm").  
JavaObject("MyButton").GetStatics("java.lang.System").gc
```

To retrieve the value of the `out` property of the `java.lang.System` class, you can insert a statement similar to the following:

```
Set OutStream=
Browser("Browser").Page("Page").JavaApplet("mybuttonapplet.htm").
JavaObject("MyButton").GetStatics("java.lang.System").out
```

To print a message to the Java console, you can insert a statement similar to the following:

```
Set OutStream=
Browser("Browser").Page("Page").JavaApplet("mybuttonapplet.htm").
JavaObject("MyButton").GetStatics("java.lang.System").out
OutStream.println "Hello, World!"
```

Firing Java Events

You can simulate an event on a Java object during a run session with the **FireEvent** and **FireEventEx** methods. The **FireEvent** method simulates an event on a Java object using one of several pre-defined event constants. If the list of pre-defined constants does not cover the event you want to fire, you can use the **FireEventEx** method to fire any Java event. For information on the syntax of these methods and for the list of pre-defined event constants, see the **Java** section of the *HP QuickTest Professional Object Model Reference*.

For example, you can use the **FireEvent** method to fire a `MouseClicked` event on the `JavaObject` called `MyButton_0`.

```
Browser("Browser").Page("Page").Applet("mybuttonapplet.htm").JavaObject
("MyButton_0").FireEvent micMouseClicked, 0, "BUTTON1_MASK", 4, 4, 1, "OFF"
```

Alternatively, you can use the **FireEventEx** method to fire the same event as follows:

```
Browser("Browser").Page("Page").Applet("mybuttonapplet.htm").JavaObject  
("MyButton_0").FireEventEx "java.awt.event.MouseEvent",  
"MOUSE_CLICKED", 0, "BUTTON1_MASK", 4,4, 1, "False"
```

Note that you can pass any Java constant that is used as one of the event's constructor parameters using its string, rather than its value. In the example above, the "java.awt.event.MouseEvent" Java constant `MOUSE_CLICKED` is supplied as a string argument instead of its value (500 in this example).

13

Troubleshooting Testing Java Applets and Applications

This chapter is intended to help pinpoint and resolve some common problems that may occur when testing Java applets and applications.

This chapter includes:

- ▶ Identifying and Solving Common Problems on page 230
- ▶ Checking Java Environment Variables Settings on page 233
- ▶ Locating the Java Console on page 235
- ▶ Running an Application or Applet with the Same Settings on page 237
- ▶ Running the Java Add-in on Multiple Environments on page 238
- ▶ Disabling Dynamic Transformation Support (Advanced) on page 239
- ▶ Additional Notes and Limitations on page 241

Identifying and Solving Common Problems

The QuickTest Professional Java Add-in provides a number of indicators that help you identify whether your add-in is properly installed and functioning. The following table describes the indicators you may see when your add-in is not functioning properly and suggests possible solutions:

Indicator	Solution
<p>You cannot record or run tests on Java applets or applications, or the Object Spy identifies Java objects as Standard Windows objects.</p>	<ul style="list-style-type: none"> ▶ Make sure that the Java Add-in is loaded with QuickTest. To check this, select Help > About QuickTest Professional and verify that the Java Add-in check box is selected. <p>You load the Java Add-in using the Add-in Manager. For more information, see "Loading QuickTest Add-ins" on page 28.</p>
<p>You cannot record or run tests on Java applets running on Microsoft Internet Explorer, and the Object Spy identifies Java objects in these applets as Standard Windows objects.</p>	<p>If you are working on the Microsoft Vista operating system, or a later version of Microsoft Windows, and you are using the Sun Java 6 or 7 JRE on Microsoft Internet Explorer 7 or later, the JVM might not use the Java settings added to your system's environment variables.</p> <p>Use the Java Add-in JRE Support Tool to adjust your computer's configuration to overcome this problem. The tool is available in the Start > Programs > HP QuickTest Professional > Tools program group.</p> <p>For more information, see:</p> <ul style="list-style-type: none"> ▶ "Checking Java Environment Variables Settings" on page 233 ▶ "Using the Java Add-in on Applets Running on Microsoft Windows Vista or Later and Internet Explorer 7 or Later" on page 234

Indicator	Solution
<p>The Java console does not display a line containing text similar to "Loading Java Support".</p>	<p>Check that the settings in your environment correspond to the environment settings defined in this chapter, or check for a batch file that may override the settings.</p> <p>For more information, see:</p> <ul style="list-style-type: none"> ▶ "Checking Java Environment Variables Settings" on page 233 ▶ "Locating the Java Console" on page 235
<p>A different applet or application works with the QuickTest Professional Java Add-in, but the application you want to test does not work.</p>	<p>First check whether you can record and run tests if you invoke the other Java applet or application using exactly the same settings.</p> <p>Check that the settings in your environment correspond to the environment settings defined in this chapter, or check for a batch file that may override the settings.</p> <p>For more information, see:</p> <ul style="list-style-type: none"> ▶ "Running an Application or Applet with the Same Settings" on page 237 ▶ "Checking Java Environment Variables Settings" on page 233
<p>After installing the Java Add-in, you cannot run Java applications using the IBM Java 6 JVM.</p>	<p>Check that the settings in your environment correspond to the environment settings defined in "Checking Java Environment Variables Settings" on page 233, or check for a batch file that may override the settings.</p> <p>In addition, you may need to do the following:</p> <ol style="list-style-type: none"> 1 Remove -Xrunjvms from the _JAVA_OPTIONS and IBM_JAVA_OPTIONS environment variables. 2 Add -agentlib:jvms at the beginning of the _JAVA_OPTIONS and IBM_JAVA_OPTIONS environment variables. 3 Delete the JAVA_TOOL_OPTIONS environment variable.

Indicator	Solution
<p>The add-in does not function properly with applications that run with the -Xincgc option.</p>	<p>Either remove the -Xincgc option, or run without dynamic transformation support.</p> <p>For more information, see:</p> <ul style="list-style-type: none"> ▶ "Running the Java Add-in on Multiple Environments" on page 238. ▶ "Disabling Dynamic Transformation Support (Advanced)" on page 239.
<p>Your Java console contains the line: Could not find -Xrun library: jvmhook.dll.</p>	<p>Check that the jvmhook.dll is located within your java.library.path For more information, see "Running the Java Add-in on Multiple Environments" on page 238.</p>
<p>None of the indicators above describe my problem.</p>	<p>See "Additional Notes and Limitations" on page 241</p>

If, after reviewing the above indicators and solutions, you are still unable to record and run tests on your Java applet or application, see the HP Software Support Web site.

Checking Java Environment Variables Settings

This section describes the environment variables that need to be set when you load your Java application with QuickTest Professional Java Add-in support. You need to set one or more environment variables to the short path name of the Java Add-in support classes folder.

Set the `_JAVA_OPTIONS` environment variable (Sun) or the `IBM_JAVA_OPTIONS` environment variable (IBM) as follows:

```
-Xrunjvmhook -Xbootclasspath/a:<program files>\HP\QUICKT~1\bin\
JAVA_S~1\classes;<program files>\HP\QUICKT~1\bin\JAVA_S~1\classes\
jasmine.jar
```

The above settings should appear on one line (no newline separators).

If you are working with Sun Java 6 or 7 (versions 1.6 or 1.7), you must set an additional environment variable, `JAVA_TOOL_OPTIONS`, with the value `-agentlib:jvmhook`

Note: `<program_files>` denotes the short path of the **Program Files** folder. For example, if the **Program Files** folder is located in `C:\Program Files`, then the value for `-Xbootclasspath` is as follows:

```
-Xbootclasspath/a:C:\PROGRA~1\HP\QUICKT~1\bin\JAVA_S~1\classes;
C:\PROGRA~1\HP\QUICKT~1\bin\JAVA_S~1\classes\jasmine.jar
```

Tip: If needed, you can temporarily remove Java support by renaming the `_JAVA_OPTIONS` or `IBM_JAVA_OPTIONS` environment variable. (If you are working with Java 6 or 7, you need to rename the `JAVA_TOOL_OPTIONS` environment variable as well.) For example, you must remove Java support if you want to test ActiveX controls that are embedded in SWT- or Eclipse-based applications.

Running Java applications on the IBM Java Runtime Environment (JRE) 1.6

In some cases, after installing the Java Add-in, Java applications running on the IBM Java 6 JVM cannot be started. The error message displayed may indicate that Mercury Interactive support could not be loaded and the Java Virtual Machine could not be created.

Workaround:

- 1 Remove `-Xrunjvmhook` from the `_JAVA_OPTIONS` and `IBM_JAVA_OPTIONS` environment variables.
- 2 Add `-agentlib:jvmhook` at the beginning of the `_JAVA_OPTIONS` and `IBM_JAVA_OPTIONS` environment variables.
- 3 Delete the `JAVA_TOOL_OPTIONS` environment variable.

Using the Java Add-in on Applets Running on Microsoft Windows Vista or Later and Internet Explorer 7 or Later

In some cases, when using the Microsoft Windows Vista operating system, or a later version of Microsoft Windows, and running Java applets using the Sun Java 6 or 7 JRE on Microsoft Internet Explorer 7 or later, the Java Add-in does not recognize the applet as belonging to the Java environment. It does not recognize objects in the applet as Java objects, and cannot record or run steps on them.

This happens when the JVM does not use the Java Add-in's settings from the environment variables. In this case, you need to set `-agentlib:jvmhook -Xbootclasspath/ a:C:\PROGRA~1\HP\QUICKT~1\bin\JAVA_S~1\classes; C:\PROGRA~1\HP\QUICKT~1\bin\JAVA_S~1\classes\jasmine.jar` in the JVM Runtime Parameters.

Use the Java Add-in JRE Support Tool to set this string in the Runtime Parameters for the relevant JVM. The tool is available from: **Start > Programs > HP QuickTest Professional > Tools > Java Add-in JRE Support Tool**

Locating the Java Console

The Java console is the window in which your Java application or applet displays messages. The location of the Java console changes according to your application setup. Your Java application can be:

- ▶ A standalone application
- ▶ Run in an applet viewer
- ▶ An applet run in Microsoft Internet Explorer or Mozilla Firefox

If your Java application is a standalone application:

Open the batch file or shortcut that invokes the application and look for the command that launches Java (**java.exe**, **javaw.exe**, **jre.exe**, or **jrew.exe**).

- ▶ If the application was run with **java.exe** or **jre.exe**, it will load with a console (Command prompt window).
- ▶ If the application was run with **javaw.exe** or **jrew.exe**, it will not load with a console (the console is unavailable). You can check for Java Add-in support by invoking the application with **java.exe** or **jre.exe**. Do this by altering your batch file or the shortcut invoking your application.

Note: **java.exe** and **javaw.exe** are nearly identical, as are **jre.exe** and **jrew.exe**. The only difference between them is whether they launch a console window.

If your Java application runs in an applet viewer:

Look in the DOS command prompt window that invoked the applet viewer.

If there is no DOS command prompt window, your applet viewer may be run by a batch file similar to a standalone application. For more information, see the information on **javaw** and **jrew** in the standalone application section above.

If your Java applet runs in Microsoft Internet Explorer or Mozilla Firefox:

- If your applet runs in Microsoft Internet Explorer using the Sun Java plug-in:

Right-click the Java (plug-in) icon in your taskbar tray and click the option that opens the console (for example, Open Console or Show Console, depending on the installed version).

If you do not see the Java (plug-in) icon in your taskbar tray, select **Start > Settings > Control Panel** and double-click the Java icon or option (select the Java version used by your application). Then, in the displayed dialog box, select the option to show the Java console (for example, Show console). Note that the actual name of the option, and its location in the dialog box, depend on the Java version used by your application.) Confirm the change (for example, by clicking Apply). Restart the browser.

Note: To find out whether your Microsoft Internet Explorer works with the Sun Java plug-in, select **Tools > Internet Options > Advanced**. Under **Java (Sun)** verify that **Use Java** is selected. Java plug-in version 1.3 or later automatically configures Internet Explorer to work with the Sun Java plug-in.

- If your applet runs using the Microsoft Internet Explorer internal Virtual Machine:

In Microsoft Internet Explorer, select **Tools > Internet Options**. In the Advanced tab, look for **Microsoft VM**. Select **Java console enabled (requires restart)** and click **OK**. Restart the browser and invoke your application. Select **View > Java Console**.

- If your applet runs in Mozilla Firefox:

In Mozilla Firefox, select **Tools > Java Console**.

If you do not see the **Java Console** option in the **Tools** menu, install the **Open Java Console** extension from <https://addons.mozilla.org/firefox/141/>. This extension provides the menu option on the **Tools** menu for opening the Java Console from Mozilla Firefox. It also provides a toolbar button in the JavaScript Console for opening Java Console.

Running an Application or Applet with the Same Settings

In some cases, running another Java application or applet with the exact same settings helps determine whether you are encountering a general problem with the Java Add-in or an application-specific problem.

To run an application or applet with the same settings:

- Determine whether the application is a standalone application or an applet.
- If the application is an applet, check the browser type.
- If the applet is executed from a shortcut, execute the applet with the same command.
- If the applet is executed from a batch file, copy the batch file and change only the class file that invokes the applet.

Note: If the classpath must also be changed, add only the new items needed. Do not remove any of the items from the original application or applet classpath.

Running the Java Add-in on Multiple Environments

The Java add-in uses a mechanism that supports multiple Java environments (such as, SUN JRE, IBM JRE, and Oracle JInitiator) and multiple Java versions (such as, JDK 1.5.x, 1.6.x and so on) without requiring any configuration changes. (For a list of supported environments and versions, see the *HP QuickTest Professional Product Availability Matrix*, available from the Documentation Library Home page or the root folder of the QuickTest Professional DVD.)

This mechanism, known as the **dynamic transformation support** mechanism, adjusts the Java Add-in support classes according to the Java environment and version used. The dynamic transformation support mechanism uses the Tool Interface of the Java Virtual Machine (JVMTI) (or the Profiler Interface (JVMPI) when working with JDK 1.5 and earlier).

The dynamic transformation support mechanism is invoked by the **-Xrunjvmhook** option, which is supplied to the JVM. If the **-Xrunjvmhook** option is specified, the JVM hook profiler (part of the Java Add-in support) is loaded with every Java application or applet that loads. The JVM hook profiler dynamically transforms the necessary classes to enable context-sensitive Java support.

When you run the Java Add-in on Java 6 or Java 7 environments, the dynamic transformation support mechanism is invoked by the **-agentlib:jvmhook**, which is defined in the `JAVA_TOOL_OPTIONS` environment variable.

Note: When working with Sun Java 6 or Java 7 there is no conflict between **-agentlib:jvmhook** (defined in the `JAVA_TOOL_OPTIONS` environment variable) and **-Xrunjvmhook** (defined in the `_JAVA_OPTIONS` environment variable) because Java 6 and Java 7 ignore **-Xrunjvmhook**.

When working with IBM Java 6 or Java 7, these environment variables may conflict. For workaround details, see "Running Java applications on the IBM Java Runtime Environment (JRE) 1.6" on page 234.

The Java agent searches for the **jvmhook.dll** according to the **java.library.path** system property. You can identify any override of this system property using the Java command line: **-djava.library.path = <path>**. However, although you can override the **java.library.path** system property, it is recommended to extend the **java.library.path** and not to overwrite it.

By default, the value of the **java.library.path** system property is the system path. If your application is loaded with a different library path, you must either add the **jvmhook.dll** to a location within the **java.library.path**, or change the **java.library.path** to contain **<Windows installation folder>/system32**.

The **<JRE root folder>/bin** folder is always located in the **java.library.path**. If needed, you can manually copy the **jvmhook.dll** to this folder. However, if you need to modify more than one computer, it is recommended to modify the batch file that alters the **java.library.path**.

Disabling Dynamic Transformation Support (Advanced)

If the dynamic transformation support mechanism does not work properly, you can disable it and manually configure the Java environment to use the Java Add-in without dynamic transformation support.

In addition, the dynamic transformation support mechanism is not supported when using the incremental garbage collector (**-Xincgc** option). Therefore, if you absolutely must use the **-Xincgc** option, you need to disable dynamic transformation support.

You disable dynamic transformation support by performing the following steps:

- Save the dynamically transformed classes, as described on page 240
- Disable dynamic transformation support by disabling the JVM hook profiler, as described on page 241

After you perform these steps, the saved transformed classes will be used instead of dynamic transformation.

To save the dynamically transformed classes:

- 1** Specify the folder in which to save the dynamically transformed classes that will be generated during the preliminary launching of your java applet or application.

To do this, open the registry editor (select **Start > Run**, type `regedit` in the **Open** box and click **OK**) and navigate to the **JavaAgent** main key, located in: `HKEY_LOCAL_MACHINE\SOFTWARE\Mercury Interactive\JavaAgent`. Define a new string value named **ClassesDumpFolder**, and set its value data to an existing folder (preferably empty) on your computer, for example, `C:\JavaSupportClasses`.

Note: If the **ClassesDumpFolder** string value already exists, you can modify its value data to an existing folder on your computer.

- 2** If you are using the **-Xincgc** option, temporarily remove it from the command line to enable the JVM hook profiler to transform and save the necessary classes.
- 3** Launch your applet or application and perform some basic operations on it. This ensures that all of the necessary classes are transformed and saved. Close your applet or application. All of the dynamically transformed classes are now saved in the folder you specified in the previous step (for example, `C:\JavaSupportClasses`).
- 4** If you temporarily removed the **-Xincgc** option from the command line in step 2, you can restore it now.

Now that you saved the transformed classes, you are ready to disable dynamic transformation support.

To disable dynamic transformation support:

- 1 Remove the `-Xrunjvhook` option from the `_JAVA_OPTIONS` (or `IBM_JAVA_OPTIONS` for IBM VM-based applications, and `JAVA_TOOL_OPTIONS` if you are working with Java 6) environment variable.
- 2 Add the following option instead:
`-Xbootclasspath/p:<ClassesDumpfolder>\Final` where `<ClassesDumpfolder>` is the value of the folder in which the dynamically transformed classes were saved (step 1 on page 240). For example, after your modification the `_JAVA_OPTIONS` environment variable might look like this:

```
-Xbootclasspath/p:C:\JavaSupportClasses\Final -Xbootclasspath/
a:C:\PROGRA~1\HP\QUICKT~1\bin\JAVA_S~1\classes;C:\PROGRA~1\HP\QUICK
T~1\bin\JAVA_S~1\classes\jasmine.jar
```

Additional Notes and Limitations

This section contains general information and limitations about the Java add-in, and includes the following sections:

- "Installing the Java Add-in" on page 242
- "Loading the Java Add-in" on page 242
- "Creating and Running Testing Documents" on page 243
- "Record and Run Options" on page 243
- "Working with Java Controls" on page 244
- "Test Objects and Methods" on page 245
- "Checkpoints and Output Values" on page 246

Installing the Java Add-in

In Windows XP and Windows 2003, after you install the QuickTest Professional Java Add-in, the Windows Remote Shell Service (**rshsvc.exe**) may fail and display an error message every time you restart the computer. This occurs only if the Remote Shell Service is configured to run automatically.

Workaround: Either disable the automatic launching of the Remote Shell Service, or move the following variables from the System Variables section of the Environment Variables dialog box to the User Variables section: `_classload_hook`, `_JAVA_OPTIONS`, `IBM_JAVA_OPTIONS`, and `MSJAVA_ENABLE_MONITORS`.

Loading the Java Add-in

You load Java Add-in extensibility support by selecting a child add-in under Java in the Add-in Manager. If you load support that was developed using a Java Add-in Extensibility SDK version earlier than version 10.00, then when you open one of the QuickTest dialog boxes that display test object classes for a selected environment (such as the Object Identification dialog box), the extensibility test object classes are displayed in the wrong list. If you select the child add-in in the **Environment** list, the list of test object classes is empty. Instead, the extensibility test object classes are displayed directly under the Java environment instead of being displayed under the child add-in in the **Environment** list.

Additionally, in some cases, the **Generate Script** button in the Object Identification dialog box does not function properly.

Workaround:

- 1 Locate the test object configuration file associated with the child add-in: `<QuickTest Installation Folder>\dat\Extensibility\Java\<add-in name>TestObjects.xml`.
- 2 (If working with Quality Center: `<QuickTest Add-in for Quality Center Installation Folder>\dat\Extensibility\Java\<add-in name>TestObjects.xml`.)
- 3 In the XML file, locate the **PackageName** attribute in the **TypeInformation** element, and change its value from `JavaPackage` to the name of the child add-in.

- 4 Save the file and reopen QuickTest.
- 5 If this extensibility support (child add-in) was developed by a third party, you may want to contact them for assistance.

Creating and Running Testing Documents

- ▶ If, while recording keyboard operations in a JFC single-line edit box in an IME composition window, you press the ENTER key to select the composition string, the key press may be recorded as the **Activate** method, thereby generating an extra step. For example:
JavaWindow("Application").JavaEdit("User Name").Activate
This extra step generally does not affect the run session adversely.

Workaround: Before running your test or component, remove the extra step that was recorded.

- ▶ The ALT+F4 keyboard shortcut (used for closing a Java applet or Java application) is not supported for recording or running.

Workaround: Use a **Close** menu command or button to close a Java applet or Java application during a recording session. Alternatively, manually add a JavaWindow(...).Close step.

Record and Run Options

- ▶ Adding a **-Xincgc** flag to the **java.exe** command line (in the Record and Run Settings dialog box or in a batch file) prevents the Java support from working properly.

Workaround: When testing with QuickTest Java support, do not use **-Xincgc** in your command line, or, alternatively, do not use the dynamic transformation support mechanism. For more information, see the *HP QuickTest Professional Add-ins Guide*.

- ▶ When selecting a JAR file from the command line in the Record and Run Settings dialog box, you should manually add **-jar** to the **Command line** box before you invoke the Java application.
- ▶ If you intend to launch your Java application using the Record and Run Settings dialog box without using a batch file (or another executable file), and without the **-jar** command line option (after selecting a JAR file), you should include the fully qualified name of the Java class in the **Command line** box.

Working with Java Controls

- ▶ By default, moving and resizing of Java windows are not recorded. This is because it may cause redundant recordings in some cases.

Workaround: To instruct the Java Add-in record these actions, use the **Setting.Java** method to set the **record_win_ops** variable to **1**. For example:
`Setting.Java("RECORD_WIN_OPS") = 1`

- ▶ AWT popup menus are recorded by the Standard Window control support WinMenu test object (while other Java menus are recorded using the JavaMenu test object). You cannot perform checkpoints or Active Screen operations on such menus.

Workaround: Use other verification methods (such as using **GetTOProperty**). For more details on verification methods, see the *HP QuickTest Professional User Guide*.

- ▶ A call to **.Object.startModal** of a `JavaInternalFrame` or `JavaDialog` object may cause QuickTest to behave unexpectedly until the dialog box is closed.
- ▶ The use of multibyte characters in a multiline edit field object is not supported.
- ▶ The Java Add-in does not record or run steps for hovering over identifiers in an Eclipse window.

- For button objects (either `JavaButton` or a `button` in a `JavaToolbar`) whose label is determined by the name of the image file they display, the process of naming the test object when running in JDK 1.6 is different than the one used when running in JDK 1.5.

Therefore, if you have a test or component containing button objects that were learned on JDK 1.5 and labeled according to their image file, when you run it on JDK 1.6, the test or component may fail.

Workaround:

- For a `JavaButton` object—relearn the object on JDK 1.6. Then modify the test to use the new test object, or delete the old object from the object repository and rename the new test object to match the object name used in the step. (Make sure the **Automatically update test and components steps when you rename test objects** option is selected in the General pane of the Options dialog box.)
- For a `button` in a `JavaToolbar` object—modify the **Item** argument in the `JavaToolbar` statement to refer to the relevant button. You can specify the button's index, or you can use the Object Spy to spy on the toolbar button, and then provide the label identification property as the **Item** argument.
- When the Active Screen displays a Java applet or ActiveX control within a Web page, the applet or control is for viewing purposes only and you cannot perform operations (for example, create checkpoints, add methods, and so forth) on the object.

Workaround: Record an operation on the Java applet/ActiveX control to create a step on the object with the ActiveX Add-in and/or Java Add-in loaded. Then you can create a checkpoint, parameterize a step, or add a method from the individual Java applet/ActiveX control in the Active Screen.

Test Objects and Methods

The **PropertyValue** argument (second argument) of the **WaitProperty** method for any Java test object can be only of type **string**.

Workaround: Use a string instead of the original type. For example, instead of **1**, use **"1"**. For example: `y = JavaCheckBox("Active").WaitProperty("enabled", "1", 1000)`

Checkpoints and Output Values

- ▶ You can create text checkpoints and text output values only for Java objects that meet specific criteria. For more information, see the *HP QuickTest Professional Add-ins Guide*.
- ▶ To create a new table checkpoint on a Java table while editing a test or component, you must first open the application containing the table you want to check and display the table in the application.
- ▶ If you add a checkpoints on a `JavaList` or `JavaTree` object while editing a test or component, the **list_content** or **tree_content** property is not available in the checkpoint.

Workaround: Create checkpoints on Java lists and Java trees while recording.

- ▶ Performing a checkpoint on an object that is not always visible (such as a list opening from a combo box selection or a menu item) is not fully supported.

Workaround: If a checkpoint on a transient object is required, make sure the object is visible prior to executing the checkpoint. For example, in the case of combo box list, you should insert a statement that clicks the combo box button before executing the checkpoint.

Part VI

The Oracle Add-in

14

Using the Oracle Add-in

You can use the QuickTest Professional Oracle Add-in to test Oracle Applications and Oracle Forms objects (controls).

For details on supported Oracle environments, see the **Oracle Add-in** section of the *HP QuickTest Professional Product Availability Matrix*, available from the Documentation Library Home page or the root folder of the QuickTest Professional DVD.

The Oracle Add-in provides test objects, methods, and properties that can be used when testing objects in Oracle applications. For more information, see the **Oracle** section of the *HP QuickTest Professional Object Model Reference*.

The following table summarizes basic information about the Oracle Add-in and how it relates to some commonly-used aspects of QuickTest.

Prerequisites	
Opening Your Application	You can open your Oracle application before or after opening QuickTest.
Add-in Dependencies	<ul style="list-style-type: none">▶ The Web Add-in must be loaded. The Web Add-in supports Web-based forms.▶ The Java Add-in must be loaded if your Oracle test or component includes Java test objects.
Other	<ul style="list-style-type: none">▶ Verify that the Oracle Name attribute is unique. See "Verifying Whether the Oracle Server Supplies Unique Name Attributes" on page 252▶ Enable the Oracle Name attribute. See "Enabling the Oracle Name Attribute" on page 253

General Information	
Checkpoints and Output Values	<ul style="list-style-type: none"> ▶ See the sections describing checkpoints and output values in the <i>HP QuickTest Professional User Guide</i> (for tests), and the <i>HP QuickTest Professional for Business Process Testing User Guide</i> (for components). ▶ See "Supported Checkpoints and Output Values Per Add-in" on page 769
Setting Preferences	
Options Dialog Box	<p>Use the Java pane if your Oracle test or component includes Java test objects. (Tools > Options > Java node)</p> <p>See "The Options Dialog Box: Java Pane" on page 194</p>
Record and Run Settings Dialog Box (tests only)	<p>Use the Oracle tab. (Automation > Record and Run Settings)</p> <p>See "The Record and Run Settings Dialog Box: Oracle Tab" on page 267.</p>
Test Settings Dialog Box (tests only)	<ul style="list-style-type: none"> ▶ Use the Web pane. (File > Settings > Web node) See "Defining Web Settings for Your Test" on page 69 ▶ Use the Java pane if your Oracle test or component includes Java test objects. (File > Settings > Java node) See "The Settings Dialog Box: Java Pane" on page 204
Custom Active Screen Capture Settings Dialog Box (tests only)	<p>Use the Oracle applications section. (Tools > Options > Active Screen node > Custom Level)</p> <p>See the section on the Custom Active Screen Capture Settings dialog box in the <i>HP QuickTest Professional User Guide</i>.</p>

Application Area Settings Dialog Box (components only)	<ul style="list-style-type: none"> ▶ Use the Web pane if your test includes Web test objects. (File > Settings) See "Defining Web Settings for Your Application Area" on page 71 ▶ Use the Java pane if your Oracle test or component includes Java test objects. (File > Settings) See "The Settings Dialog Box: Java Pane" on page 204 (The options shown in the Java pane of the Test Settings dialog box are the same options that are available in the Application Area Settings dialog box.)
--	--

This chapter includes:

- ▶ Considerations for Working with the Oracle Add-in on page 251
- ▶ Verifying Whether the Oracle Server Supplies Unique Name Attributes on page 252
- ▶ Enabling the Oracle Name Attribute on page 253

Considerations for Working with the Oracle Add-in

- ▶ After installing the Oracle Add-in, your applications will always open with Java support active. You can confirm that your Oracle environment has opened properly by checking the Java console for the confirmation message similar to:
Loading Oracle Support (version x.x build xxx) (Oracle Corporation x.x.x.xx).

Note: The QuickTest Professional Oracle Add-in supports only Oracle clients that are Java-based. Oracle Developer/2000 is not supported.

- ▶ Before using the Oracle Add-in to test Oracle Applications, you must first enable the Name attribute supplied by the Oracle Applications server. For more information, see "Enabling the Oracle Name Attribute" on page 253.

- ▶ The Oracle Applications server supplies a unique **Name** attribute for many application objects. You can also find the Oracle Applications server **Name** attribute in the Oracle Add-in **developer name** identification property. The **developer name** identification property is used by QuickTest in most test object descriptions to identify Oracle objects.
- ▶ In QuickTest, table data is always loaded from the application itself, even if the Active Screen contains an image of the table. For this reason, you must first open the table in the application before creating a table checkpoint in a test.
 - ▶ In some cases you may have to scroll to the last row of the table to make sure that all the data is loaded.
 - ▶ If the table object is not open in your application when you create the checkpoint, the Table Checkpoint Properties dialog box contains only the Properties tab, and the option to select which type of information to check (content or properties) is disabled.
 - ▶ It is not necessary to open the table in your application to edit an existing table checkpoint.
- ▶ For more information on QuickTest functionality, see the *HP QuickTest Professional User Guide* and the *HP QuickTest Professional for Business Process Testing User Guide*.

Verifying Whether the Oracle Server Supplies Unique Name Attributes

To check whether the server supplies unique **Name** attributes, use the Object Spy to point to a few edit boxes inside the Oracle application and view the **developer name** attribute. If the **developer name** is displayed in all capital letters in the format FORM:BLOCK:FIELD or FORM_BLOCK_FIELD, then the **developer name** attribute is supplied correctly.

If the **developer name** value is empty, then the server does not supply unique **Name** attributes. To use the Oracle Add-in to test Oracle Applications, your Oracle server must supply unique **Name** attributes.

Your Oracle server administrator can assist you in enabling unique **Name** attributes.

To enable the Oracle server to supply unique Name attributes:

- 1 Add the following line to the server configuration file (for example, \$OA_HTML/bin/appsweb_UKTRN_hwu00001.cfg):

```
otherparams=record=names
```

- 2 Restart the Oracle server.

Enabling the Oracle Name Attribute

Before using the Oracle Add-in to test Oracle Applications, you must first enable the **Name** attribute supplied by the Oracle Applications server.

To enable the Name attribute when accessing the application directly:

Add record=names to the URL parameters.

Example:

```
http://oracleapps.mydomain.com:8002/dev60cgi/f60cgi?record=names
```

To enable the Name attribute when using HTML to launch the Oracle application:

- 1 In the startup HTML file that is used to launch the application, locate the line: <PARAM name="serverArgs fndnam= APPS">
- 2 Add the Oracle key: record=names.

Example:

```
<PARAM name="serverArgs" value="module=f:\FNDSCSGN userid=XYZ  
fndnam=apps record=names">
```

To enable the Name attribute when using the Personal Home Page to launch your Forms 6 application:

Set up the following system profile option at (your) user level to enable the **Name** attribute:

- 1** Sign on to your Oracle application and select System Administrator responsibility.
- 2** Select **Nav > Profile > System**.
- 3** In the Find System Profile Values form:
 - ▶ Confirm that **Display: Site and Users** contains your user logon.
 - ▶ Enter %ICX%Launch% in the **Profile** box.
 - ▶ Click the **Find** button.
- 4** Copy the value from the **Site** box of the **ICX: Forms Launcher** profile and paste it in the **User** box. Add `&play=&record=names` to the end of the URL in the **User** box.
- 5** Save your transaction.
- 6** Sign on again using your user name.

Note: If the **ICX: Forms Launcher** profile option is not updatable at the user level, access **Application Developer** and select the **Updatable** check box for the **ICX_FORMS_LAUNCHER** profile.

15

Troubleshooting Testing Oracle Applications

This chapter is intended to help pinpoint and resolve problems that may occur when testing Oracle applications.

This chapter includes:

- ▶ Identifying and Solving Common Problems on page 256
- ▶ Checking Oracle Environment Settings on page 257
- ▶ Locating the Java Console on page 258
- ▶ Understanding Dynamic Transformation Support on page 259
- ▶ Disabling Dynamic Transformation Support (Advanced) on page 260
- ▶ General Notes & Limitations on page 262

Identifying and Solving Common Problems

The QuickTest Professional Oracle Add-in provides a number of indicators that help you identify whether your add-in is properly installed and functioning. The following table describes the indicators you may see when your add-in is not functioning properly, and suggests possible solutions:

Indicator	Solution
You cannot record or run tests on Oracle Applications.	Ensure that the Oracle Add-in is loaded. For more information, see "Loading QuickTest Add-ins" on page 28.
The Java console does not display a line containing the text similar to: Loading Oracle Support.	Check that the settings in your environment correspond to the environment settings defined in this chapter, or check for a batch file that may override the settings. For more information, see: <ul style="list-style-type: none"> ▶ "Checking Oracle Environment Settings" on page 257. ▶ "Locating the Java Console" on page 258.
Your Java console contains the line Could not find – Xrun library: jvmhook.dll.	Check that you have jvmhook.dll in your system folder (WINNT\system32 or windows\system).
You cannot use QuickTest to record on Oracle Applications running on Oracle JInitiator versions 1.1.X.	The version of Oracle JInitiator 1.1.X on which your Oracle Application runs must be installed before you install the QuickTest Professional Oracle Add-in. If you installed Oracle JInitiator versions 1.1.X on your computer after you installed the Oracle Add-in, you should repair the Oracle Add-in installation. For more information, see the section on repairing your QuickTest Professional installation in the <i>HP QuickTest Professional Installation Guide</i> .

If, after reviewing the above indicators and solutions, you are still unable to record and run tests on your Oracle application, contact HP Software Support.

Checking Oracle Environment Settings

This section describes the environment settings you need for loading your Oracle application with QuickTest Oracle Add-in support. For all the environments, you need to set one or more environment variables with the short path name of the Oracle Add-in support classes folder.

Sun Plug-in 1.4.1 and Oracle JInitiator 1.3.1.x

Set the `_JAVA_OPTIONS` environment variable as follows:

```
-Xrunjvmhook  
-Xbootclasspath/a:C:\PROGRA~2\HP\QUICKT~1\bin\JAVA_S~1\classes;  
C:\PROGRA~2\HP\QUICKT~1\bin\JAVA_S~1\classes\jasmine.jar
```

The above settings should appear on one line (no newline separators).

Note that `common_files` denotes the short path of the Common Files folder located in the Program Files folder. For example, if the Common Files folder is in `C:\Program Files\Common Files`, then the value for `-Xbootclasspath` is as follows:

```
-Xbootclasspath/a:C:\Programme\HP\QuickTest Professional\bin\java_shared\  
classes;  
C:\Programme\HP\QuickTest Professional\bin\java_shared\classes\jasmine.jar
```

Oracle JInitiator 1.1.x

Set the `_classload_hook` environment variable to `jvmhook`.

Locating the Java Console

The Java console is the window in which your Oracle application displays messages. The location of the Java console changes according to your application setup, as follows.

If your application runs in Oracle JInitiator 1.3 or higher:

Do one of the following:

- ▶ Right-click the **JInitiator** icon in the taskbar tray and click **Show Console**.
- ▶ If you do not see the JInitiator icon in the taskbar tray, click **Settings > Control Panel** in the **Start** menu. Double-click the **JInitiator** icon (choose the icon for the Java version used by your application). In the Basic tab, select **Show Java console** and click **Apply**. Restart your JInitiator application.

If your application runs in Oracle JInitiator 1.1.x:

If you do not see the JInitiator icon in the taskbar tray, click **Programs > JInitiator Control Panel** in the **Start** menu. In the Basic tab, select **Show Java console** and click **Apply**. Restart your JInitiator application.

If your application runs in JDK 1.4 Plug-in:

Do one of the following:

- ▶ Right-click the Java Plug-in icon in the taskbar tray and click **Open Console**.
- ▶ If you do not see the Java Plug-in icon in the taskbar tray, click **Settings > Control Panel** in the **Start** menu. Double-click the **Java Plug-in** icon. In the Basic tab, select **Show Java in System Tray**. Restart the browser.

Understanding Dynamic Transformation Support

The Oracle Add-in uses a mechanism for supporting multiple Java environments (Sun Plug-in, JInitiator) and their versions (JInitiator 1.1.8, 1.3.1, and so on) without requiring any configuration changes. This mechanism is known as dynamic transformation support.

Dynamic transformation support uses the profiler interface of the Java Virtual Machine (JVM) to adjust the Oracle Add-in support classes according to the Java environment and version in use.

The dynamic transformation support mechanism is invoked by the **-Xrunjvmhook** option (for JInitiator 1.3.1.x and Sun Plug-in 1.4.1) or the **_classload_hook=jvmhook** option (for JInitiator 1.1.x) supplied to the JVM. If this option is specified, the JVM hook profiler, which is part of the Oracle Add-in support, is loaded with every application or applet and dynamically transforms the necessary classes to enable context-sensitive Oracle support.

If the dynamic transformation support mechanism does not work properly, you can disable it and manually configure the Oracle environment to use the Oracle Add-in without dynamic transformation support. For more information, see "Disabling Dynamic Transformation Support (Advanced)" on page 260.

Note: The dynamic transformation support mechanism is not supported when using the incremental garbage collector (**-Xincgc** option). Therefore, if you absolutely must use the **-Xincgc** option, you need to disable dynamic transformation support.

Disabling Dynamic Transformation Support (Advanced)

You can disable the dynamic transformation support mechanism if required and manually configure the Oracle environment to use the Oracle Add-in without dynamic transformation support.

You disable dynamic transformation support by performing the following steps:

- ▶ Launching the application you are testing and saving the dynamically transformed classes, as described on page 260.
- ▶ Disabling dynamic transformation support in Sun Plug-in 1.4.1 or JInitiator 1.3.1.x, as described on page 261.

or

Disabling dynamic transformation support in JInitiator 1.1.x, as described on page 261.

The saved transformed classes are used instead of dynamic transformation.

To save the dynamically transformed classes:

- 1 Specify the folder in which to save the dynamically transformed classes that will be generated during the preliminary launching of your Oracle application.

To do this, open the registry editor (select **Start > Run**, type `regedit` in the **Open** box and click **OK**) and navigate to the **JavaAgent** main key, located in: `HKEY_LOCAL_MACHINE\SOFTWARE\Mercury Interactive\JavaAgent`. Define a new string value named **ClassesDumpFolder**, and set its value data to an existing folder (preferably empty) on your computer, for example, `C:\JavaSupportClasses`.

Note: If the **ClassesDumpFolder** string value already exists, you can modify its value data to an existing folder on your computer.

- 2 If you are using the **-Xincgc** option, temporarily remove it from the command line to enable the JVM hook profiler to transform and save the necessary classes. You can add it back to the command line after performing the following step.
- 3 Launch your applet or application and perform some basic operations on it. This ensures that all of the necessary classes are transformed and saved. Close your application. All of the dynamically transformed classes are now saved in the folder you specified in the previous step (for example, C:\JavaSupportClasses).

After you save the transformed classes, you disable dynamic transformation support.

To disable dynamic transformation support in Sun Plug-in 1.4.1 or JInitiator 1.3.1.x:

- 1 Remove the **-Xrunjvmhook** option from the **_JAVA_OPTIONS** environment variable.
- 2 Add the following option instead: **-Xbootclasspath/p:<ClassesDumpFolder>\Final**, where **<ClassesDumpFolder>** is the value of the folder in which the dynamically transformed classes were saved (step 1 on page 260), appended by the **Final** subfolder. For example, after your modification the **_JAVA_OPTIONS** environment variable might look like this:

```
-Xbootclasspath/p:C:\JavaSupportClasses\Final -Xbootclasspath/
a:C:\Programme\HP\QuickTest Professional\bin \java_shared\classes;
```

To disable dynamic transformation support in JInitiator 1.1.x:

- 1 Remove the **_classload_hook** option from the JDK settings by deleting the environment variable.
- 2 Manually copy the classes from the **<ClassesDumpFolder>**, where **<ClassesDumpFolder>** is the value of the folder in which the dynamically transformed classes were saved (step 1 on page 260), appended by the **Final** subfolder, to the JInitiator 1.1.x classes folder. The JInitiator 1.1.x classes folder can be typically found under C:\Program Files\Oracle\JInitiator 1.1.x\classes.

General Notes & Limitations

This section contains general information and limitations about the Oracle add-in, and includes the following sections:

- ▶ "Installing the Oracle Add-in" on page 262
- ▶ "Test Objects and Methods" on page 263
- ▶ "Recording and Running Testing Documents" on page 263
- ▶ "Record and Run Options" on page 264
- ▶ "Checkpoints" on page 264

Installing the Oracle Add-in

- ▶ If you install an Oracle JInitiator 1.1.x version after you install the QuickTest Professional Oracle Add-in, you must repair QuickTest to test applications running in the newly installed JInitiator version. For more information, see the *HP QuickTest Professional Add-ins Guide*.

Note: It is not necessary to re-install or otherwise configure the QuickTest Professional Oracle Add-in if you installed a new Oracle environment other than JInitiator 1.1.x.

- ▶ In Windows XP and Windows 2003, after you install the Oracle Add-in, the Windows Remote Shell Service (**rshsvc.exe**) may fail and display an error message every time you restart the computer. This occurs only if the Remote Shell Service is configured to run automatically.

Workaround: Either disable the automatic launching of the Remote Shell Service, or move the following variables from the System Variables section of the Environment Variables dialog box to the User Variables section: `_classload_hook`, `_JAVA_OPTIONS`, `IBM_JAVA_OPTIONS`, and `MSJAVA_ENABLE_MONITORS`.

Test Objects and Methods

- ▶ **OracleListOfValues.Select** always selects the first item when there are multiple items with identical values in the first column.

Workaround: To select an item other than the first one, specify the item's index value instead of a string as the argument value.

- ▶ During a run session, an **OracleCalendar.Enter** step may sometimes enter a value in the wrong field.

Workaround: Use an **OracleTextField.Enter** step to enter the value in the date field instead.

- ▶ Test objects that require the index property for their description (for example, range flexfield objects) cannot be created from the Active Screen.

Workaround: Use the **Add Objects** button in the Object Repository dialog box to add these test objects directly from your Oracle Applications instead.

Recording and Running Testing Documents

- ▶ The hierarchy of a tab inside of another tab is not recorded correctly. This stops auto-tab selection from working on such hierarchies during a run session.

Workaround: Add an **OracleTabbedRegion.Select** method for the appropriate tab before any step that performs an operation on an object inside a tab that is inside another tab.

- ▶ Active Screen captures are not supported for OracleListOfValues and OracleNotification test objects.
- ▶ The recovery scenario pop-up window trigger event is not supported when testing Oracle Applications.
- ▶ Running a test or component on an Oracle Applications session that was created by refreshing the browser is not supported.
- ▶ Simultaneous testing of multiple Oracle Applications sessions is not supported.
- ▶ The use of multibyte characters in a multiline edit field object is not supported.

Record and Run Options

The **Log out of the application when the test closes option** in the Record and Run Settings dialog box does not work if the Responsibilities List of Values window is displayed in the Oracle Applications session.

Checkpoints

- ▶ Performing a checkpoint on an object that is not always visible (such as a list opening from a combo box selection or a menu item) is not fully supported.

Workaround: If a checkpoint on a transient object is required, make sure the object is visible prior to executing the checkpoint. For example, in the case of combo box list, you should insert a statement that clicks the combo box button before executing the checkpoint.

- ▶ When testing Oracle applications, a table checkpoint may not capture the values of columns that are not visible.

Workaround: Before creating a table checkpoint, scroll in the table so that the last column is visible.

16

Creating and Running Steps on Oracle Applications

This chapter explains how to use QuickTest to set testing preferences and to create and run steps on Oracle Applications sessions.

This chapter includes:

- ▶ About Creating and Running Steps on Oracle Applications on page 265
- ▶ Defining Record and Run Settings for Oracle Tests on page 266
- ▶ Creating Steps on Oracle Applications on page 272

About Creating and Running Steps on Oracle Applications

As you record on an Oracle Applications session, QuickTest inserts statements into your test or component that represent the operations you perform. The QuickTest Professional Oracle Add-in recognizes specific Oracle objects such as button, form, navigator, list, and tree. It records these objects in relation to the data selected or entered and to the object within its parent object.

Note: QuickTest Professional does not record the selection of Oracle tabs. Each object in an Oracle tab is included in the object repository within the tab hierarchy. QuickTest then uses this hierarchy when the test or component is run, switching to the appropriate tab if needed.

Working with Tests

Each time you begin recording a test, you can use the Oracle tab of the Record and Run Settings dialog box to instruct QuickTest to connect to a specified Oracle Applications server. Alternatively, you can instruct QuickTest to record on any open browser. For more information, see "Defining Record and Run Settings for Oracle Tests" on page 266.

Working with Components

The Record and Run Settings dialog box is not used for components. When you record a component on an Oracle Applications session, you cannot instruct QuickTest to open or connect to a specified Oracle Applications server. You must open and connect to it manually or include statements in your component (using the **OpenApp** operation and **OracleLogon** test object) that open and connect to the Oracle Applications server.

Defining Record and Run Settings for Oracle Tests

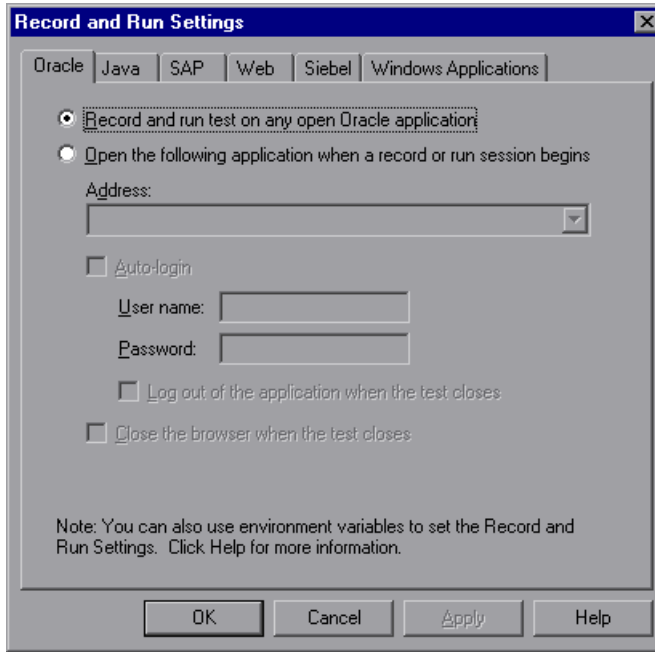
You can use the Oracle tab of the Record and Run Settings dialog box to instruct QuickTest to connect to a particular Oracle server and open an Oracle Applications session each time you begin a recording session. Alternatively, you can instruct QuickTest to record on any open browser.

Components do not require specific record and run settings to work with Oracle applications. To record a component on an Oracle Applications session, you need to first open the Oracle Applications session manually or include steps in your component (using the **SystemUtil** utility object and **OracleLogon** test object) that connect to the Oracle Applications server. When you begin recording a new component, the Applications dialog box opens (unless you previously specified a Windows environment in the Application Area Settings or Business Component Settings dialog box). Click **OK** to close the dialog box and begin recording. For more information on the Applications pane and Applications dialog box, see the *HP QuickTest Professional for Business Process Testing User Guide*.

The Record and Run Settings Dialog Box: Oracle Tab

Description	<p>Enables you to specify whether or not to connect to an Oracle Applications server and open a specified Oracle Applications session when a record or run session begins. If you select to connect to a specific server, you can specify details that will enable QuickTest to automatically log on to the server each time a record or run session begins (instead of recording the log-in steps).</p>
How to Access	<p>Do one of the following:</p> <ul style="list-style-type: none"> ▶ Automation > Record and Run Settings ▶ If you do not modify the Record and Run settings before you begin recording, the Record and Run Settings dialog box opens automatically when you begin recording a new test (by clicking Record (or choosing Automation > Record)).
Important Information	<p>If you load only the QuickTest Professional Oracle Add-in and the Web add-in, then only the Oracle, Web, and Windows Applications tabs are displayed in the Record and Run Settings dialog box. If other add-ins are loaded, the corresponding tabs (if any) are also displayed.</p>
Learn More	<p>Conceptual overview:</p> <ul style="list-style-type: none"> ▶ "Using the Oracle Add-in" on page 249 ▶ "About Creating and Running Steps on Oracle Applications" on page 265 ▶ Defining Record and Run Settings for Oracle Tests on page 266 <p>Additional related topics: "Additional References" on page 269</p>

Below is an image of the Oracle tab:



Oracle Tab Options

Option	Description
Record and run test on any open Oracle Application	Instructs QuickTest to record and run the test on any open Oracle application.
Open the following application when a record or run session begins	Instructs QuickTest to connect to the Oracle Applications server at the specified URL address. Note: This setting controls only which application, if any, is opened at the beginning of a record or run session. It does not affect the applications that QuickTest recognizes. Even if this radio button is selected and no application is specified, QuickTest can still record, recognize, and run on any open Oracle application.
Address	Indicates the URL of the Oracle Applications server to which you want to connect.

Option	Description
Auto-login	<p>Instructs QuickTest to log on to the specified Oracle Applications server using the specified user name and password.</p> <p>Enabled only when Open the following application when a record or run session begins is selected.</p> <p>The Auto-login feature works for the Java interface login only. If you log in to your Oracle applications through a Web interface, the Auto-login feature cannot be used.</p>
User name	<p>The user name used to log on to the specified server.</p> <p>Enabled only when Auto-login is selected.</p>
Password	<p>The password for the specified user name.</p> <p>Enabled only when Auto-login is selected.</p>
Log out of the application when the test closes	<p>Instructs QuickTest to log out of the Oracle Applications session specified in the Record and Run Settings dialog box when the test is closed.</p> <p>Enabled only when Auto-login is selected.</p>
Close the browser when the test closes	<p>Instructs QuickTest to close the browser on which the test is recorded when the test is closed.</p> <p>Enabled only when Open the following application when a record or run session begins is selected.</p>

Additional References

Related Tasks	<ul style="list-style-type: none"> ▶ "Defining Record and Run Environment Variables" on page 270 ▶ "Creating Steps on Oracle Applications" on page 272
----------------------	--

For more information on the Record and Run Settings dialog box, see "Using the Record and Run Settings Dialog Box" on page 38.

Defining Record and Run Environment Variables

You can use record and run environment variables to specify the applications you want to use for recording and running your test. These variables can also be used in external library files for automation scripts.

If you define any of these record and run environment variables, they override the values in the corresponding boxes in the Oracle tab of the Record and Run Settings dialog box. For more information on the Oracle tab, see "Defining Record and Run Settings for Oracle Tests" on page 266.

Use the variable names listed in the table below to define Oracle record and run variables:

Option	Variable Name	Description
Address	ORACLE_URL_ENV	The URL of the Oracle Applications server to which you want to connect.
Auto-login	ORACLE_AUTO_LOGIN_ENV	Instructs QuickTest to log on automatically to the Oracle Applications server. Possible values: True False
User name	ORACLE_USER_NAME_ENV	The user name used to log on to the specified server.
Password	ORACLE_PASSWORD_ENV	The password for the specified user name.

Option	Variable Name	Description
Log out of the application when the test closes	ORACLE_LOGOUT_ENV	Instructs QuickTest to log out of the Oracle Applications session specified in the Record and Run Settings dialog box when the test is closed. Possible values: True False
Close the browser when the test closes	ORACLE_CLOSE_BROWSER_ENV	Instructs QuickTest to close the browser on which the test is recorded when the test is closed. Possible values: True False

For more information on defining and working with environment variables, see the *HP QuickTest Professional User Guide*.

Creating Steps on Oracle Applications

When you record an operation on an Oracle application, QuickTest records a step with the appropriate icon in the Keyword View and adds the corresponding statement in the Expert View.

For example, if you record the selection of an item in an Oracle List of Values window, the Keyword View may be displayed as follows:

Item	Operation	Value	Documentation
▼ Action1			
ORA Responsibilities	Select	"Assets, Vision Operations (USA)"	Select the "Assets, Vision Operations (USA)" item from the "Responsibilities"

QuickTest records this step in the Expert View as:

```
OracleListOfValues("Responsibilities").Select "Assets, Vision Operations (USA)"
```

Note: If you installed a version of JInitiator 1.1.x after installing the Oracle Add-in, a warning is displayed when you start recording your test or component. Versions of JInitiator 1.1.x installed after you install the Oracle Add-in are not supported by QuickTest. In this case, you can repair the Oracle Add-in to enable full support of all currently installed versions of JInitiator 1.1.x. For more information, see the section on repairing your QuickTest Professional installation in the *HP QuickTest Professional Installation Guide*.

If you try to record an action on an Oracle object with an unsupported version of JInitiator 1.1.x, QuickTest records a generic **WinObject.Click** statement that includes the coordinates of the click and the mouse button that was clicked.

The QuickTest learned object hierarchy is composed of one, two, or three levels of Oracle test objects. Depending on the actual object on which you performed an operation, that object may be recorded as a first level object (for example, **OracleLogon**), as a second level object (for example, **OracleFormWindow.OracleList**), or as a third level object (for example, **OracleFormWindow.OracleTabbedRegion.OracleTable**).

Even though the object on which you record may be embedded in several levels of objects, the recorded hierarchy does not include these objects. For example, even if the `OracleListOfValues` object in which you select an item is actually within an Oracle form, which is contained within an Oracle Applications session window, the recorded hierarchy is only `OracleListOfValues`. Select (without the `OracleFormWindow` and `OracleApplications` test objects in the hierarchy).

You may have a combination of Oracle and Java test objects in your Oracle test or component. This occurs when QuickTest encounters a Java applet within your Oracle Applications session and records it using the Java test object hierarchy.

You can edit steps that use Java test objects, methods, and properties in the same way as you edit other standard steps. You can add new steps to existing tests or components using the new Oracle test object model. For information on Java objects, methods, and properties, see the **Java** section of the *HP QuickTest Professional Object Model Reference*, installed together with the Oracle Add-in. For information on Oracle objects, methods, and properties, see the **Oracle** section of the *HP QuickTest Professional Object Model Reference*.

There are specific options and settings you can use in your test or component that apply only to steps that use Java test objects. These options and settings are located in the Java pane of the Test Settings dialog box (**File > Settings > Java** node) and the Java pane of the Options dialog box (**Tools > Options > Java** node). For more information, click the **Help** button in the relevant Java pane. Note that the options in the Java panes do not have any effect on Oracle object steps in your test or component.

Part VII

The PeopleSoft Add-in

17

Using the PeopleSoft Add-in

You can use the QuickTest Professional PeopleSoft Add-in to test PeopleSoft objects (controls).

For details on supported PeopleSoft environments, see the **PeopleSoft Add-in** section of the *HP QuickTest Professional Product Availability Matrix*, available from the Documentation Library Home page or the root folder of the QuickTest Professional DVD.

The PeopleSoft Add-in provides test objects, methods, and properties that can be used when testing objects in PeopleSoft applications. For more information, see the **PeopleSoft** section of the *HP QuickTest Professional Object Model Reference*.

The following table summarizes basic information about the PeopleSoft Add-in and how it relates to some commonly-used aspects of QuickTest.

General Information	
Add-in Type	This is a Web-based add-in. Much of its functionality is the same as other Web-based add-ins. See "Testing Web-Based Applications" on page 45.
Checkpoints and Output Values	<ul style="list-style-type: none">▶ See the sections describing checkpoints and output values in the <i>HP QuickTest Professional User Guide</i> (for tests), and the <i>HP QuickTest Professional for Business Process Testing User Guide</i> (for components).▶ See "Supported Checkpoints and Output Values Per Add-in" on page 769.
Prerequisites	
Opening Your Application	You must open QuickTest before opening your PeopleSoft application.

Add-in Dependencies	The Web Add-in must be loaded.
Setting Preferences	
Options Dialog Box	Use the Web pane. (Tools > Options > Web node) For more information, see "Setting Web Testing Options" on page 52.
Record and Run Settings Dialog Box (tests only)	Use the Web tab. (Automation > Record and Run Settings) See "Setting Web Record and Run Options" on page 46.
Test Settings Dialog Box (tests only)	Use the Web pane. (File > Settings > Web node) See "Defining Web Settings for Your Test" on page 69.
Custom Active Screen Capture Settings Dialog Box (tests only)	Use the Web section. (Tools > Options > Active Screen node > Custom Level) See the section on the Custom Active Screen Capture Settings dialog box in the <i>HP QuickTest Professional User Guide</i> .
Application Area Settings Dialog Box (components only)	Use the Web pane. (File > Settings > Web node) See "Defining Web Settings for Your Application Area" on page 71.

This chapter includes:

- ▶ Considerations for Working with the PeopleSoft Add-in on page 279
- ▶ Troubleshooting and Limitations - PeopleSoft Add-in on page 280

Considerations for Working with the PeopleSoft Add-in

- ▶ When learning **PSFrame** objects, or Web pages containing **PSFrame** objects, the following child objects are automatically filtered out and are not added to the object repository:

Currently the following elements are filtered out when learning a PeopleSoft Frame object (or a Web page containing a **PSFrame** object):

- ▶ WebElement
- ▶ WebTable
- ▶ Images with type "Plain Image"
- ▶ Images with type "Image Link"

If you want to add an object that is automatically filtered out, you can manually add it by selecting it in the Object Selection dialog box.

- ▶ The PeopleSoft Add-in provides a customized **PSFrame** test object to identify PeopleSoft frames. The **PSFrame** object differs from the **Web Frame** object both in its test object description and its algorithm for generating object names. This customization helps make your PeopleSoft tests easy to read and maintain.
- ▶ The PeopleSoft Add-in identifies all other objects in your PeopleSoft application using Web test objects.

For information on PeopleSoft and Web test objects, methods, and properties, see the **PeopleSoft** and **Web** sections of the *HP QuickTest Professional Object Model Reference*.

- ▶ For the purposes of Web event recording, QuickTest treats Web test objects that are child objects of a PSFrame test object as PeopleSoft objects and thus applies the settings in the PeopleSoft event configuration XML file when recording those objects.

For more information on Web event recording configurations, see "Web Event Recording Configurations" on page 74.

- ▶ For more information on QuickTest functionality, see the *HP QuickTest Professional User Guide* and the *HP QuickTest Professional for Business Process Testing User Guide*.

Troubleshooting and Limitations - PeopleSoft Add-in

- ▶ The Active Screen may not function correctly when working with non-English UI servers.
- ▶ If you use the ENTER key to activate a search operation while recording a test, QuickTest may not perform the operation as expected during the test run.

Workaround: Activate the search by clicking the **Search** button with the mouse.

- ▶ The use of keyboard shortcut keys to perform operations while recording is not supported.

Part VIII

The PowerBuilder Add-in

18

Using the PowerBuilder Add-in

You can use the QuickTest Professional PowerBuilder Add-in to test PowerBuilder objects (controls).

For details on supported PowerBuilder environments, see the **PowerBuilder Add-in** section of the *HP QuickTest Professional Product Availability Matrix*, available from the Documentation Library Home page or the root folder of the QuickTest Professional DVD.

The PowerBuilder Add-in provides test objects, methods, and properties that can be used when testing objects in PowerBuilder applications. For more information, see the **PowerBuilder** section of the *HP QuickTest Professional Object Model Reference*.

The following table summarizes basic information about the PowerBuilder Add-in and how it relates to some commonly-used aspects of QuickTest.

General Information	
Add-in Type	This is a Windows-based add-in. Much of its functionality is the same as other Windows-based add-ins. See "Testing Windows-Based Applications" on page 87.
Checkpoints and Output Values	<ul style="list-style-type: none">▶ See the sections describing checkpoints and output values in the <i>HP QuickTest Professional User Guide</i> (for tests), and the <i>HP QuickTest Professional for Business Process Testing User Guide</i> (for components).▶ See "Considerations for Working with the PowerBuilder Add-In" on page 285.▶ See "Supported Checkpoints and Output Values Per Add-in" on page 769.

Other	See "Considerations for Working with the PowerBuilder Add-In" on page 285.
Prerequisites	
Opening Your Application	You can open your PowerBuilder application before or after opening QuickTest.
Add-in Dependencies	None
Setting Preferences	
Options Dialog Box	Use the Windows Applications pane. (Tools > Options > Windows Applications node) See "The Options Dialog Box: Windows Applications Pane" on page 103.
Record and Run Settings Dialog Box (tests only)	Use the Windows Applications tab. (Automation > Record and Run Settings) See "The Record and Run Settings Dialog Box: Windows Applications Tab" on page 90. Note: If you select the Record and Run only on radio button in the Record and Run Settings dialog box, the settings also apply to (limit) the applications that are recognized for Object Spy and other pointing hand operations.
Custom Active Screen Capture Settings Dialog Box (tests only)	Use the Windows applications section. (Tools > Options > Active Screen node > Custom Level) See the section on the Custom Active Screen Capture Settings dialog box in the <i>HP QuickTest Professional User Guide</i> .
Application Area Settings Dialog Box (components only)	Use the Applications pane. (File > Settings > Applications node) See the section on defining Application Settings for your application area in the <i>HP QuickTest Professional for Business Process Testing User Guide</i> .

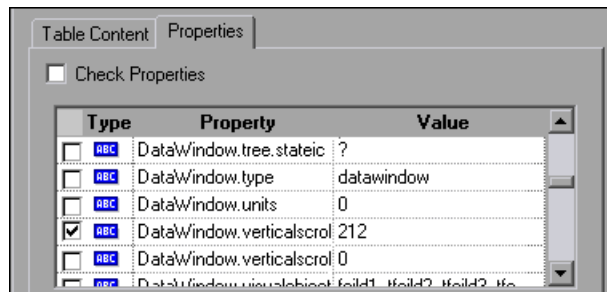
This chapter includes:

- ▶ Considerations for Working with the PowerBuilder Add-In on page 285
- ▶ Troubleshooting and Limitations - PowerBuilder Add-in on page 286

Considerations for Working with the PowerBuilder Add-In

The PowerBuilder Add-in provides the PbDataWindow test object with customized methods and properties to help you test PowerBuilder's DataWindow control.

- ▶ When you insert a checkpoint or output value step on a DataWindow control, QuickTest treats it as a table and opens the Table Checkpoint Properties or Table Output Value Properties dialog box (not supported for components). It enables you to check or retrieve values for the table content and the object properties.
- ▶ When you insert a checkpoint or output value step on a DataWindow control during a recording session, the properties available to be checked or retrieved in the Properties tab include the DataWindow control's inner attributes (such as DataWindow.color) in addition to the identification properties (such as enabled and focused).



The set of DataWindow inner attributes available in the dialog box is the same as the list of properties that would be returned if you run a DataWindow.Describe ("DataWindow.attributes") statement. Properties of the inner objects of the table (objects that can be retrieved using a DataWindow.Describe ("DataWindow.objects") statement) are not available in this list.

- ▶ When you insert a checkpoint or output value step on a DataWindow control while editing (from the Active Screen, or on a step for which Active Screen data was captured), only the identification properties are available in the list.

For more information on the DataWindow test object, see the **PowerBuilder** section of the *HP QuickTest Professional Object Model Reference*.

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

Troubleshooting and Limitations - PowerBuilder Add-in

This section describes troubleshooting and limitations for the PowerBuilder Add-in.

- ▶ When learning or recording on toolbars in PowerBuilder applications, QuickTest no longer records the PbToolbar test object. Instead, it records a **PbObject.Click** object. The PbToolbar test object is no longer available in QuickTest dialog boxes or in the documentation.

If a PbToolbar test object exists in an old object repository, it will be recognized and supported, but toolbar-specific methods such as **CheckItem**, **GetContent**, **GetItem**, **GetItemProperty**, **GetItemCount**, **GetSelection**, **Press**, **ShowDropDown**, and **WaitItemProperty** are not supported for this object.

You should update object repositories and tests to use the PbObject test object for toolbar steps.

Part IX

The Add-in for SAP Solutions

19

Using the Add-in for SAP Solutions on Web-based SAP Applications

You can use the SAP Web testing support provided with the QuickTest Professional Add-in for SAP Solutions to test objects in Web-based SAP applications, including SAP GUI for HTML, SAP Enterprise Portal (versions 5.0 to 7.0), Internet Transaction Server, SAP Customer Relationship Management (CRM) 2007, and the Interaction Centre Web Client.

The QuickTest Professional Add-in for SAP Solutions has been certified by SAP AG.

For details on supported Web-based SAP environments, see the **Add-in for SAP Solutions** section of the *HP QuickTest Professional Product Availability Matrix*, available from the Documentation Library Home page or the root folder of the QuickTest Professional DVD.

The Add-in for SAP Solutions provides test objects, methods, and properties that can be used when testing objects in Web-based SAP applications. For more information, see the **SAP Web** section of the *HP QuickTest Professional Object Model Reference*.

When the QuickTest Professional Add-in for SAP Solutions is loaded, QuickTest can learn objects and run steps on both Web-based and Windows-based SAP applications. For information on recording and running tests and components on SAP GUI for Windows applications, see "Using the Add-in for SAP Solutions on SAP GUI for Windows Applications" on page 335.

The following table summarizes basic information about QuickTest Web-based SAP support and how it relates to some commonly-used aspects of QuickTest.

General Information	
Add-in Type	QuickTest SAP Web testing support functions like a Web-based add-in. Much of its functionality is the same as other Web-based add-ins. See "Testing Web-Based Applications" on page 45.
Checkpoints and Output Values	<ul style="list-style-type: none"> ▶ See the sections describing checkpoints and output values in the <i>HP QuickTest Professional User Guide</i> (for tests), and the <i>HP QuickTest Professional for Business Process Testing User Guide</i> (for components). ▶ See "Enhancing Your SAP Web Test" on page 299. ▶ See "Supported Checkpoints and Output Values Per Add-in" on page 769.
Other	The Add-in for SAP Solutions recognizes special SAP Web objects such as frames, table controls, iViews, and portals. See "Adding SAP Web Statements to Your Test or Component" on page 303.
Prerequisites	
Opening Your Application	<ul style="list-style-type: none"> ▶ Open QuickTest before you open your Web-based SAP Application. ▶ If you are working in a SAP GUI application with HTML objects in it, you can log on to your application before opening QuickTest, but you must open QuickTest before navigating to the transaction containing the HTML objects. ▶ For SAP GUI for HTML or Interaction Centre Web Client (ICWC) applications, confirm that you have properly configured your SAP server and client. See "Setting Up Your SAP GUI for Windows Environment" on page 313.
Add-in Dependencies	The Web Add-in must be loaded.

Setting Preferences	
Options Dialog Box	Use the Web pane. (Tools > Options > Web node) See "Setting Web Testing Options" on page 52.
Record and Run Settings Dialog Box (tests only)	<ul style="list-style-type: none"> ▶ Use the SAP tab (Automation > Record and Run Settings) to connect to the SAP GUI Client for SAP GUI for HTML or Interaction Centre Web Client (ICWC) applications. This is because ICWC opens from inside the SAP GUI Client. See "Setting Up Your SAP GUI for Windows Environment" on page 313. ▶ Use the Web tab (Automation > Record and Run Settings) to instruct QuickTest to open an SAP Web-based application, or the SAP Enterprise Portal, at the beginning of each record and run session, by specifying its URL. Alternatively, you can instruct QuickTest to record on any open browser. See "Setting Web Record and Run Options" on page 46.
Test Settings Dialog Box (tests only)	Use the Web pane. (File > Settings > Web node) See "Defining Web Settings for Your Test" on page 69.
Custom Active Screen Capture Settings Dialog Box (tests only)	Use the Web section. (Tools > Options > Active Screen node > Custom Level) See the section on the Custom Active Screen Capture Settings dialog box in the <i>HP QuickTest Professional User Guide</i> .
Application Area Settings Dialog Box (components only)	Use the Web pane. (File > Settings > Web node) See "Defining Web Settings for Your Application Area" on page 71.
Web Event Recording Configuration Dialog Box	(Tools > Web Event Recording Configuration) When you load the Add-in for SAP Solutions, the settings in the Web Event Recording Configuration dialog box are automatically customized. You do not need to make any Web event configuration changes.

This chapter includes:

- ▶ Recording Tests on Web-based SAP Applications on page 292
- ▶ Troubleshooting and Limitations - Web-based SAP Support on page 296

Recording Tests on Web-based SAP Applications

Before you begin recording tests on Web-based SAP applications, you can define your required recording settings. This enables you to specify the browser on which you want QuickTest to record, specify any environment variables, and select the required Web options to optimize performance.

For more information, see:

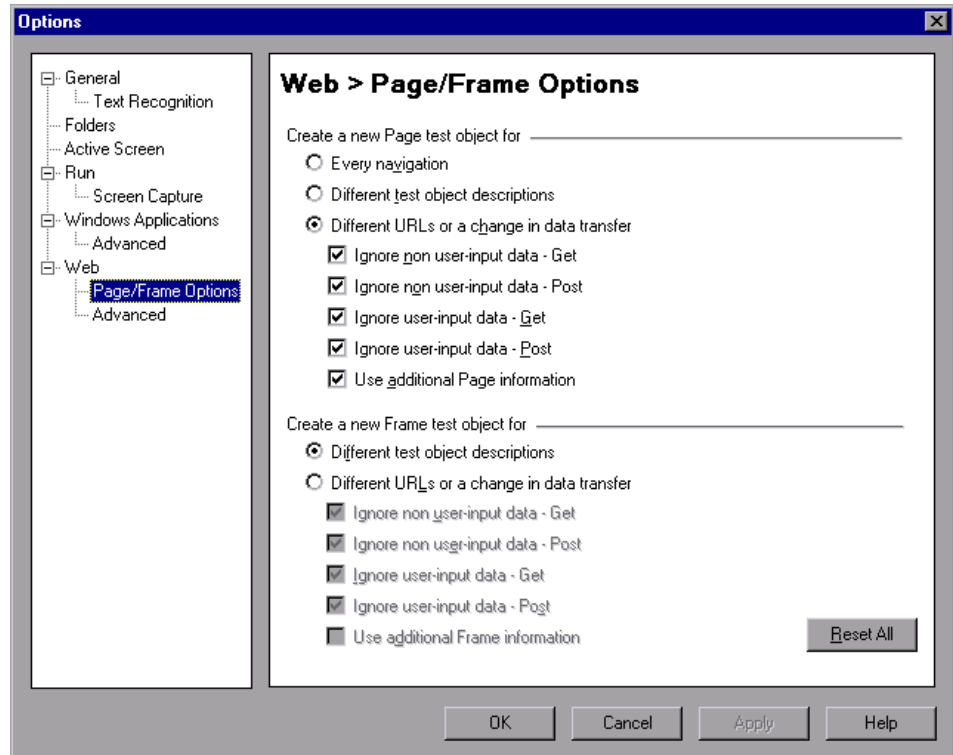
- ▶ "Setting Web Record and Run Options" on page 46
- ▶ "Setting Web Testing Options" on page 52

Setting Web Testing Options for Tests on Web-based SAP Applications

Before you begin to record and run tests, you can configure the settings that are best-suited for testing Web-based SAP applications. Applying these recommended settings helps to optimize QuickTest performance.

Page and Frame Options

In the Web > Page/Frame Options pane (**Tools > Options > Web node > Page/Frame Options node**), select the following options:

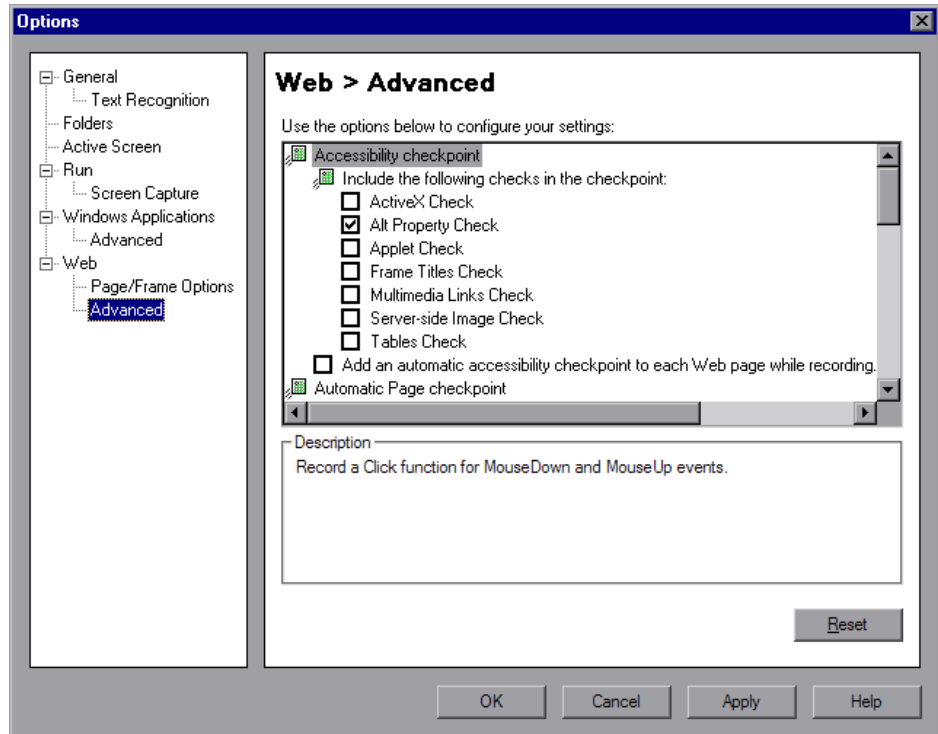


Area	Options
<p>Create a new Page test object for:</p>	<p>Different URLs or a change in date transfer</p> <p>Instructs QuickTest to create a new Page object only when the page URL changes, or if the URL stays the same and data that is transferred to the server changes, according to the data types and transfer methods you select.</p> <p>Make sure that only the following check boxes are selected:</p> <ul style="list-style-type: none"> ▶ Ignore user-input data - Get ▶ Ignore user-input data - Post ▶ Use additional Page information
<p>Create a new Frame test object for:</p>	<p>Different URLs or a change in data transfer</p> <p>This option is selected by default.</p> <p>Instructs QuickTest to create a new Frame object only when the page URL changes, or if the URL stays the same and data that is transferred to the server changes, according to the data types and transfer methods you select.</p> <p>Make sure that all of the check boxes in this section are selected.</p>

These Page and Frame settings are especially suited to testing Web-based SAP applications. For more information on the Web > Page/Frame pane, see "Page and Frame Options" on page 58.

Advanced Web Options

In the Web > Advanced pane (**Tools > Options > Web node > Advanced node**), select the following **Record setting** options:



Area	Options
Record settings	Select the Use standard Windows mouse events check box, as well as the following check boxes: <ul style="list-style-type: none"> ➤ OnClick ➤ OnMouseDown ➤ OnMouseUp This instructs QuickTest to use these standard Windows mouse events instead of browser events.

These recording settings are especially suited to testing Web-based SAP applications. For more information on the Web > Advanced pane, see "Advanced Web Options" on page 62.

Troubleshooting and Limitations - Web-based SAP Support

This section contains general troubleshooting and limitation information about the Web-based SAP add-in, and includes the following sections:

- "SAP Enterprise Portal" on page 296
- "SAP Gui for HTML—Internet Transaction Server (ITS)" on page 297
- "Using the Active Screen " on page 298

SAP Enterprise Portal

- Operations on the **iView** option menu and on objects within the page title bar of SAP Enterprise Portal are recorded as Web operations on the Frame object and not as SAP operations on the iView object.
- Minimized or collapsed iViews may not be recognized correctly.
- In some cases, when more than one browser is open during the test run, QuickTest is unable to correctly identify certain objects.

Workaround: Clear the **Enable Smart Identification** check box for the Browser test objects in the Object Repository dialog box. You may also want to disable the **Enable Smart Identification** option for Browser test objects in the Object Identification dialog box for future test recording.

- ▶ In some cases, a frame in SAP Enterprise Portal may be recognized as a Web Frame object instead of an iView object. In some of these, the frame name is generated dynamically. Because the Web Frame object uses the **name** property to identify the object, you must modify the recorded **name** value to use an appropriate regular expression so that QuickTest will be able to recognize it during the test run.

SAP Gui for HTML—Internet Transaction Server (ITS)

- ▶ When testing your SAP Gui for HTML application on Windows XP, it is recommended to use the Windows Classic theme instead of the Windows XP theme, to improve performance.
- ▶ When using the Object Spy or creating a checkpoint on an object inside an SAP Web table cell, QuickTest may recognize the object as a WebElement (and not as the appropriate SAP Web object), if a click has not yet been performed on the object.

Workaround: Click on the object inside the SAP Web table cell before using the Object Spy or creating a checkpoint on it.

- ▶ Dragging the SAP Gui for HTML table scroll bar is not recorded.

Workaround: You can record scrolling in SAP Gui for HTML tables by clicking the scroll button. Alternatively, use the Step Generator or Expert View to insert a **SAPTable.Object.DoScroll("up")** or **SAPTable.Object.DoScroll("down")** statement in your test.

- ▶ The appearance of toolbar buttons may differ, and toolbar buttons may or may not be displayed, depending on the size of your browser window.

Workaround: Try to maintain the same browser window size and the resulting menu appearance when recording and running your test.

- ▶ When running a test on an ITS frame in an SAP Enterprise Portal iView, the ITS menu sometimes fails to operate properly.

Workaround: Enlarge the iView size and/or increase the **Object Synchronization Timeout** and then run the test again.

Using the Active Screen

- ▶ The Active Screen may not display the entire HTML page captured while recording your test.

Workaround: Resize the Active Screen so that it best fits the HTML page size.

- ▶ When testing an SAP Enterprise Portal application, it is recommended to set advanced authentication for Active Screen access (**Test > Settings > Web**). For more information, see the *HP QuickTest Professional Add-ins Guide*.
- ▶ Avoid using an Active Screen that was captured when a pop-up dialog was open to add an object from the main window to the object repository. Doing this results in an incorrect object hierarchy in the object repository.

20

Enhancing Your SAP Web Test

After you create your test, you can enhance it by adding checkpoints, retrieving output values, and parameterizing values.

This chapter includes:

- ▶ Checking SAP Web Objects and Outputting Values on page 299

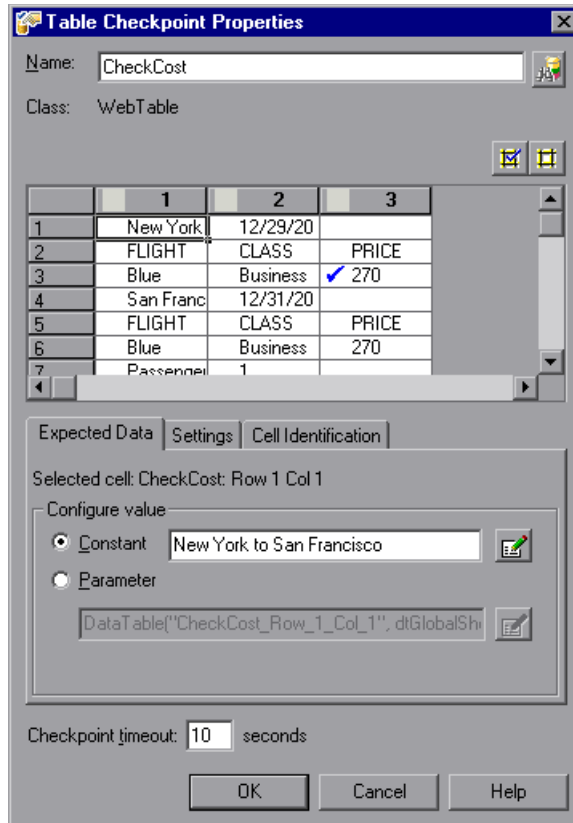
Checking SAP Web Objects and Outputting Values

After you create a test, you can use a variety of options to enhance it, including adding checkpoints and retrieving output values.

For more information on standard, table, text, and bitmap checkpoints, and on output values, see the *HP QuickTest Professional User Guide*.

Checking SAP Web Tables

When working with tests, you check tables in your SAP Web application using the Table Checkpoint Properties dialog box. Table checkpoints are not supported for business components.



Note: If the table object is not open in your SAP Web application when you create the checkpoint, the Table Checkpoint Properties dialog box contains only the Properties tab and the option to select the type of information to check (content or properties) is disabled.

When working with tables in your SAP GUI for HTML application, note that:

- ▶ You can add a table checkpoint while recording or editing your test.
- ▶ You can spool all of the available data from a table into an external file. For more information, see "Spooling Data from an SAP GUI for HTML Application Table" on page 301.
- ▶ If a table has a column header row, it is counted as the first row in the table.
- ▶ If you have not recorded a step on the table object you want to check, but you have an Active Screen capture that displays the table object, you can add a table checkpoint under the following conditions:



- ▶ The Active Screen **Capture level** was set to **Complete** when the object was captured, and
- ▶ The **Active Screen** button is currently selected. (You set the **Capture level** in the Active Screen pane of the Options dialog box (**Tools > Options > Active Screen** node). For more information, see the section on Active Screen options in the *HP QuickTest Professional User Guide*.)

Spooling Data from an SAP GUI for HTML Application Table

If you want to spool all the available data from an SAP GUI for HTML application table into an external file, use the `GetCellData` method to loop through each cell in the table. You can then save the information to an external file.

The following example uses the `GetCellData` method to list the data of each cell in a table of 10 rows and 10 columns:

```
For i=1 to 10
  For j=1 to 10
    Dat=Browser("ITS System Informati").Page("Table control").
      SAPTable("MySAPTable").GetCellData (i, j)
    'Enter lines of code that use the value of the returned Dat variable
  Next
Next
```

For more information on the `GetCellData` method, see the **SAP Web** section of the *HP QuickTest Professional Object Model Reference*.

21

Adding SAP Web Statements to Your Test or Component

After you create your test or component, you can add SAP Web objects, methods, and properties to it.

This chapter includes:

- Working with SAP Web Test Objects on page 303

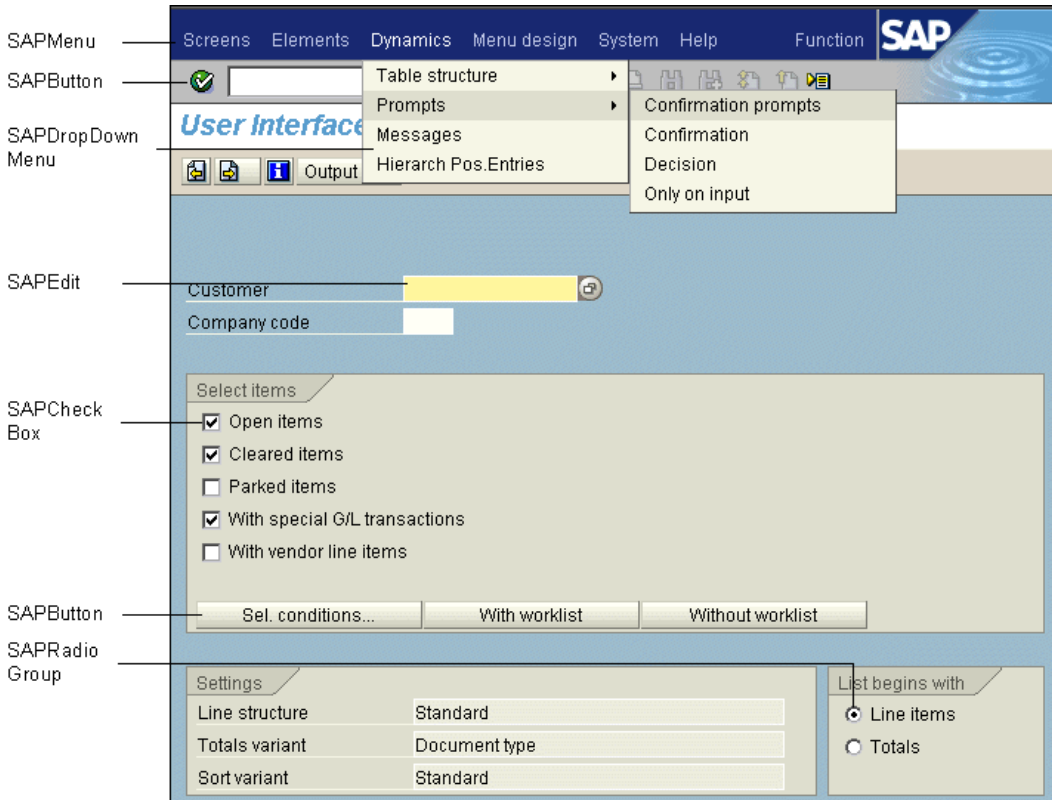
Working with SAP Web Test Objects

QuickTest has a set of SAP Web test object classes that represent objects in your application on which you can record operations, or add to your test or component manually.

Tip: You can use the Object Spy to view the native operations and properties of an object in your application.

SAP Web test objects are customized to make it easy for you to identify and work with the objects in your test. As part of this customization, the methods and properties recorded for these objects are somewhat different from those recorded for comparable Web objects.

The example below shows a window containing several common SAP Web objects:



This section describes how QuickTest identifies the objects in a Web-based SAP application and provides information on each of the following objects:

- SAPButton (see page 305)
- SAPCalendar (see page 305)
- SAPCheckBox (see page 306)
- SAPDropDownMenu (see page 306)
- SAPEdit (see page 306)
- SAPFrame (see page 306)
- SAPiView (see page 306)
- SAPList (see page 307)
- SAPMenu (see page 307)
- SAPNavigationBar (see page 308)
- SAPOKCode (see page 308)
- SAPPortal (see page 309)
- SAPRadioGroup (see page 309)
- SAPStatusBar (see page 309)
- SAPTable (see page 310)
- SAPTabStrip (see page 311)
- SAPTreeView (see page 312)

SAPButton



The SAPButton test object represents SAP GUI for HTML and SAP Enterprise Portal application buttons, including icons, toolbar buttons, regular buttons, buttons with text, and buttons with text and an image.

SAPCalendar



The SAPCalendar test object represents the ICWC Web client calendar control, which enables the user to select dates to appear in the date fields. The test object's main operation is SetDate.

SAPCheckBox



The SAPCheckBox test object represents SAP GUI for HTML and SAP Enterprise Portal application toggle buttons, including check boxes and images that can be pressed and released.

SAPDropDownMenu



The SAPDropDownMenu test object represents menus that are opened by clicking a menu icon within an SAP GUI for HTML or SAP Enterprise Portal application.

SAPEdit



The SAPEdit test object represents SAP GUI for HTML and SAP Enterprise Portal application edit boxes, including single-line edit boxes and multi-line edit boxes (text area).

SAPFrame

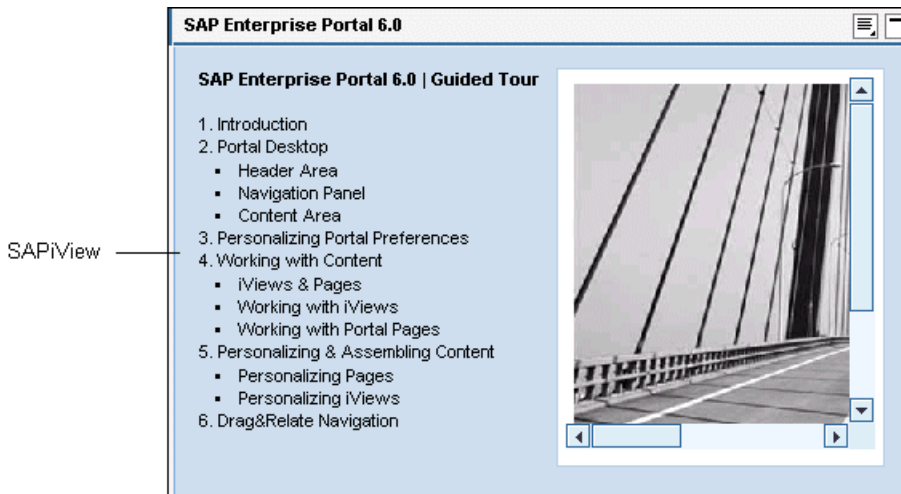


The SAPFrame test object represents SAP GUI for HTML application frames.

SAPiView



The SAPiView test object represents iView frame objects within SAP Enterprise Portal desktops.



SAPList



The SAPList test object represents SAP GUI for HTML and SAP Enterprise Portal application drop-down boxes and multiple selection lists.

A screenshot of an SAPList test object. It shows a rectangular box with a light beige background. At the top left, there is a small tab-like area containing the text "Fill using Dictionary". Below this, the text "Airline carrier" is displayed on the left side of the box. To the right of this text is a drop-down menu with a white background and a small downward-pointing arrow on its right side. The text "Lufthansa" is currently selected and displayed within the drop-down menu.

SAPMenu

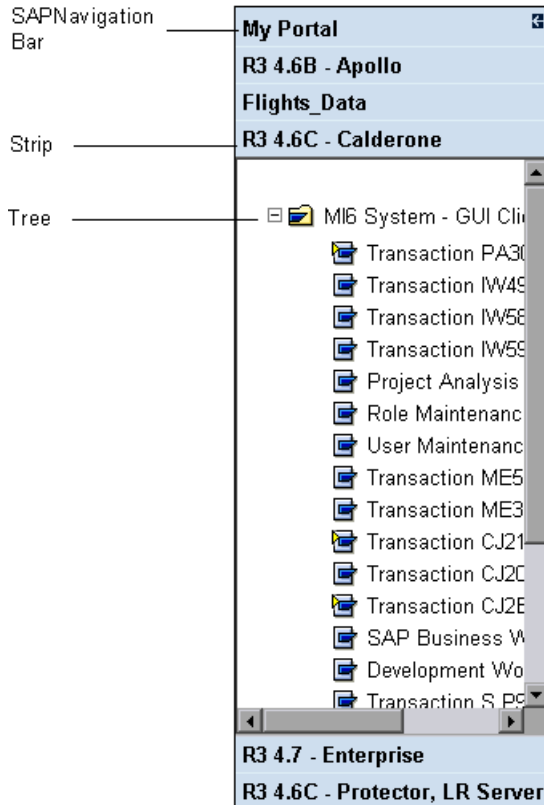


The SAPMenu test object represents SAP GUI for HTML application top-level menus. When you click a menu item, the SAPMenu test object records the full path of the selected item.

SAPNavigationBar



The SAPNavigationBar test object represents an iPanel—the navigation pane displayed in an SAP Enterprise Portal 5.0 desktop. An iPanel is divided into sections, represented by expandable strips. Each strip contains a hierarchical tree of items. When you record an operation on the SAPNavigationBar test object, it records the strip as part of the path, for example, SAPNavigationBar("SAPNavigationBar").Select "R3 4.6C - Calderone;MI6 System - GUI Client Roles;Transaction PA30".



SAPOKCode

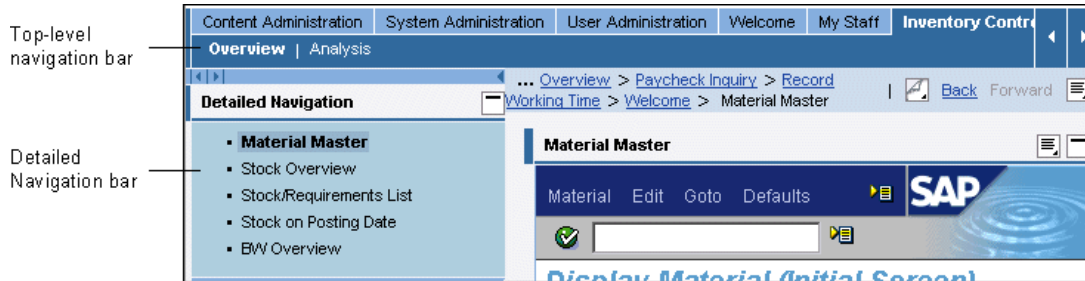


The SAPOKCode test object represents the edit box in a SAP GUI for HTML application in which you enter commands to navigate to the desired transaction.

SAPPortal



The SAPPortal test object represents the SAP Enterprise Portal desktop objects—top-level navigation bar and Detailed Navigation bar.



SAPRadioGroup



The SAPRadioGroup test object represents SAP GUI for HTML and SAP Enterprise Portal application radio button groups.

When possible, QuickTest records radio button selections using the attached text property of the selected radio button. If two or more radio buttons in the group have identical attached text values (or no attached text), QuickTest records the radio button index instead. For example:

'This radio button selection uses the attached text property.
 Browser("System Messages").Page("User Interface_52").SAPRadioGroup("Basic personal").Select "Personal data"

'This radio button selection uses the radio button index.
 Browser("System Messages").Page("User Interface_46").SAPRadioGroup("Address_2").Select "#0"

When writing SAPRadioGroup.Select statements manually in the Expert View, you can use either argument type to identify the radio button.

SAPStatusBar



The SAPStatusBar test object represents a status bar in an SAP GUI for HTML application.

i Standard Order 9596 has been saved

SAPTable



The SAPTable test object represents SAP GUI for HTML application table objects. Each cell can contain an SAP Web or Web object, such as a check box or combo box. The value of this object determines the value in the cell.

A...	Fl...	Depart.city	D...	Destination cit	D...	Flight...	Depart..
AA	17	NEW YORK	JFK	SAN FRANCISCO	SFO		13:30:00
AA	64	SAN FRANCISCO	SFO	NEW YORK	JFK		09:00:00
AZ	555	ROME	FCO	FRANKFURT	FRA		19:00:00
AZ	788	ROME	FCO	TOKYO	TYO		12:00:00
AZ	789	TOKYO	TYO	ROME	FCO		11:45:00
AZ	790	ROME	FCO	OSAKA	KIX		10:35:00

When you perform an operation on an object within a table cell, QuickTest records the changes to the data in the cell, rather than recording the method performed on the object within the cell.

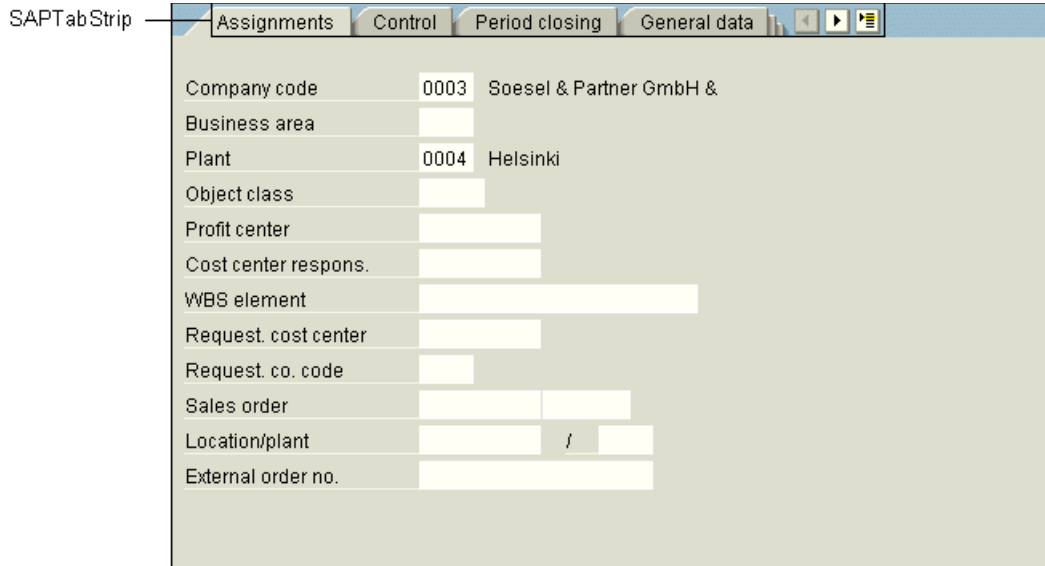
For example, if you select a table cell containing the text **San Francisco** from a list of departure cities (in the **Depart.city** column), QuickTest records a `SetCellData` method for the table cell, indicating the data that was set in the cell. If the object in the cell is later changed to an edit object, QuickTest enters the data (**San Francisco**) into the cell (in the edit object) during the run session. Thus, you do not need to modify your test even when the object inside the cell changes.

Note: You perform operations on a table object and not on the inner objects contained in the table object, for example, a check box or edit box.

SAPTabStrip



The SAPTabStrip test object represents SAP GUI for HTML tab strip objects (objects that enable switching between multiple tabs). You choose the required tab by clicking its title. If a tab is not visible, you can display it by clicking the left or right arrows.



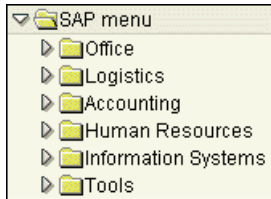
Note: While recording a test, arrow clicks are ignored. However, while running a test, a tab that is not visible will still be selected correctly by QuickTest.

Selecting a tab through the drop-down menu (the rightmost button on the tab strip) is handled exclusively via the **SAPButton.SelectMenuItem** method.

SAPTreeView



The SAPTreeView test object represents SAP GUI for HTML and SAP Enterprise Portal application tree objects.



QuickTest can record selection and activation operations on any item in an SAPTreeView object. Although QuickTest does not record expand and collapse operations, it can select and activate items in an SAPTreeView object during the run session, regardless of whether the tree is expanded or collapsed.

For more information on all the SAP Web test objects, methods, and properties see the **SAP Web** section of the *HP QuickTest Professional Object Model Reference*.

Note: When learning SAP Web pages, the following child objects are automatically filtered out and are not added to the object repository:

- ▶ WebElement
- ▶ WebTable
- ▶ Images with type "Plain Image"

If you want to add an object that is automatically filtered out, you can manually add it by selecting it in the Object Selection dialog box.

22

Setting Up Your SAP GUI for Windows Environment

Before you can start testing your SAP GUI for Windows applications, you must make sure that your server and client are installed and configured with the correct versions and support options. This chapter provides the specific setup information that you need to successfully use the QuickTest Professional Add-in for SAP Solutions. This chapter is relevant only if you test SAP GUI for Windows applications.

This chapter includes:

- ▶ About Setting Up Your SAP Windows Environment on page 314
- ▶ Installing SAP GUI Scripting Support on page 315
- ▶ Checking Package and Patch Versions Installed on the SAP Application Server on page 316
- ▶ Checking the Patch Version Installed on your SAP GUI for Windows Application on page 321
- ▶ Enabling Scripting on the SAP Application (Server-Side) on page 322
- ▶ Enabling Scripting on the SAP Application (Client-Side) on page 326
- ▶ Setting F4 Help to Use Dialog Display Mode on page 330
- ▶ Setting F1 Help to Use Modal Dialog Box Mode on page 332
- ▶ Checking the Connection Speed on the SAP Server on page 333

About Setting Up Your SAP Windows Environment

QuickTest Professional's support for SAP GUI for Windows versions 6.20, 6.40 and 7.10 is based on the SAP GUI Scripting API which is disabled by default.

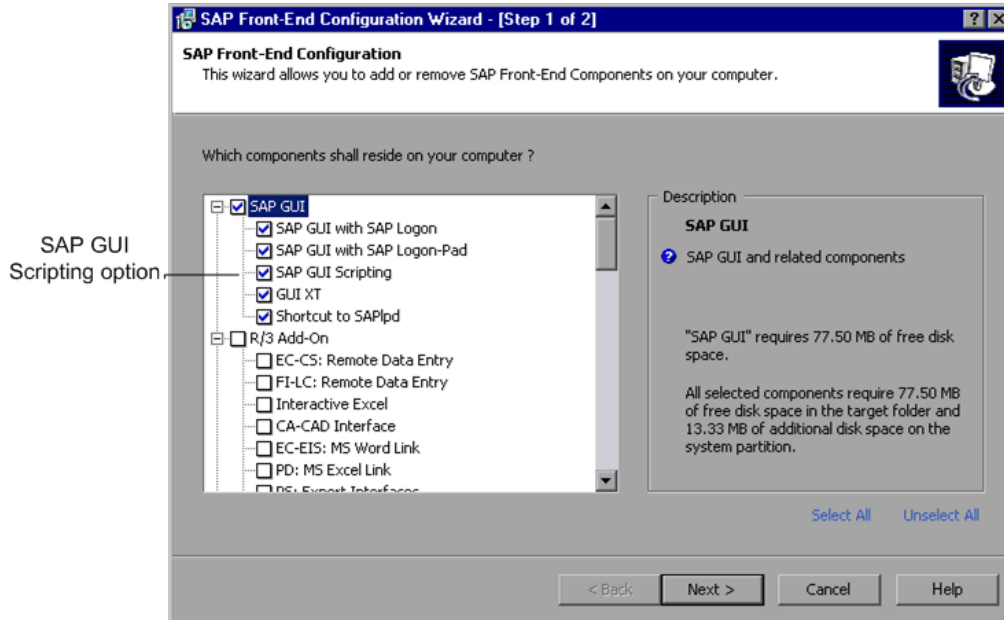
To test your SAP GUI for Windows application using the QuickTest Professional Add-in for SAP Solutions, you must confirm that:

- ▶ The SAP GUI Scripting option is installed.
- ▶ Your server and client have the proper package and patch versions installed.
- ▶ Your server supports the Scripting API.
- ▶ The Scripting API is enabled on both the server and clients.
- ▶ Your client is configured to use the **Dialog** display mode for F4 Help screens and that it is not set to use a **Low speed connection**.
- ▶ The F1 and F4 Help display setting is configured correctly, to support testing the use of the F1 and F4 Help screens in your SAP GUI for Windows application.

Note: If you plan to use the QuickTest-eCATT integration features, you must also install the appropriate support package and configure the eCATT server to work with QuickTest. For more information, see "Configuring eCATT to Work with QuickTest" on page 363.

Installing SAP GUI Scripting Support

When you install your SAP GUI for Windows application, you must select the **SAP GUI Scripting** installation option.



If you did not select this option when you installed the SAP GUI for Windows application, it is essential that you reinstall it and select this option before setting the other configuration options described in this chapter.

Note: SAP provides a range of security mechanisms that allow the administrator to limit the use of SAP GUI Scripting by system, by group, by user, and by scripting functionality. To test SAP GUI for Windows applications, you must ensure that these security mechanisms are not activated for the application you are testing. For more information on the various security options, see the online SAP GUI Scripting Security Guide at the SAP Service Marketplace.

Checking Package and Patch Versions Installed on the SAP Application Server

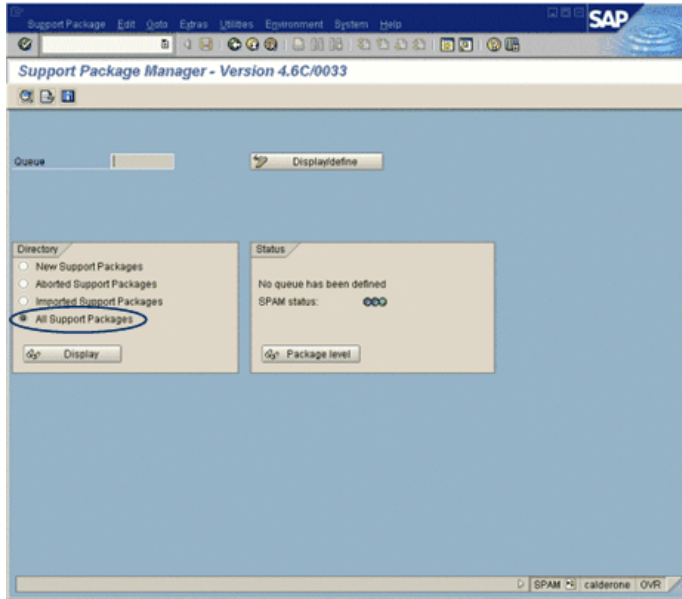
To use the QuickTest Professional Add-in for SAP Solutions, you must confirm that you have the correct support package and kernel patch levels for your software component release. The following table shows the **minimum** required versions and levels. You must have these versions and levels or higher:

Software Component	Release	Support Package	Kernel Patch Level
SAP_APPL	31I	SAPKH31I96	Kernel 3.1I level 650
SAP_APPL	40B	SAPKH40B71	Kernel 4.0B level 903
SAP_APPL	45B	SAPKH45B49	Kernel 4.5B level 753
SAP_BASIS	46B	SAPKB46B37	Kernel 4.6D level 948
SAP_BASIS	46C	SAPKB46C29	Kernel 4.6D level 948
SAP_BASIS	46D	SAPKB46D17	Kernel 4.6D level 948
SAP_BASIS	610	SAPKB61012	Kernel 6.10 level 360

To check the support package:

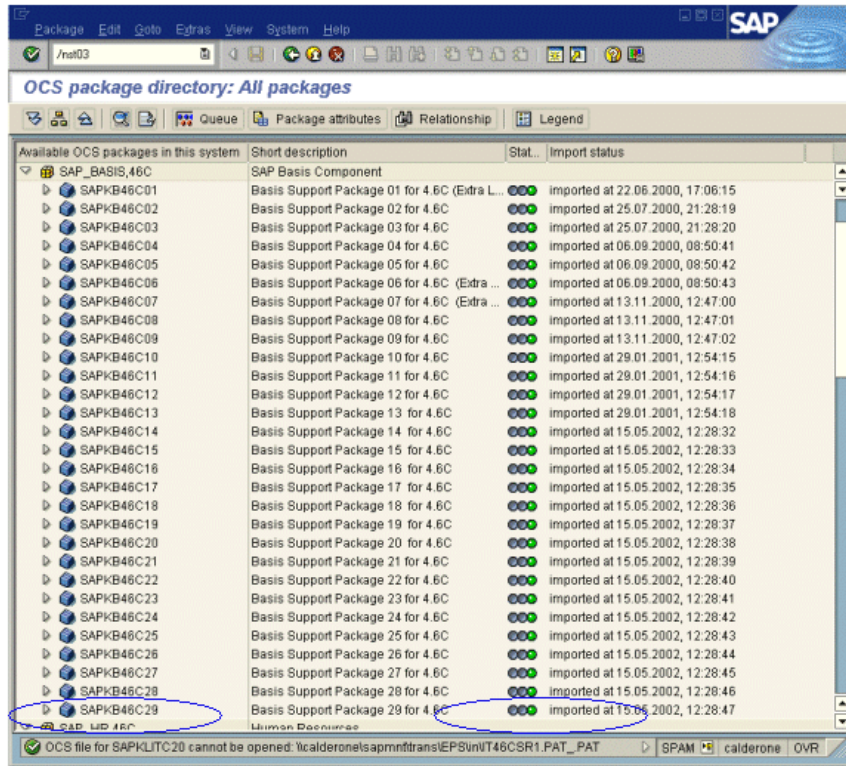
- 1 Log on to your SAP server.

2 Run the SPAM transaction.



3 In the **Directory** section, select **All Support Packages** and click the **Display** button.

- The All packages window opens. Verify that the correct package is installed for the SAP release you are using (see "Checking Package and Patch Versions Installed on the SAP Application Server" on page 316).



If the correct package is installed, a green light icon is displayed in the **Status** column.

If you do not have the required package installed, download and install it.

For more information on downloading and installing the required package, see SAP OSS note # 480149.

To check the kernel patch level:

- 1** Log on to your SAP server.
- 2** Select **System > Status**. The System: Status dialog box opens.

The screenshot shows the 'System: Status' dialog box with the following data:

Usage data			
Client	800	Previous logon	05.11.2004 16:12:38
User	QA01	Logon	07.11.2004 08:53:10
Language	EN	System time	16:37:39
		Time zone	CET 15:37:39

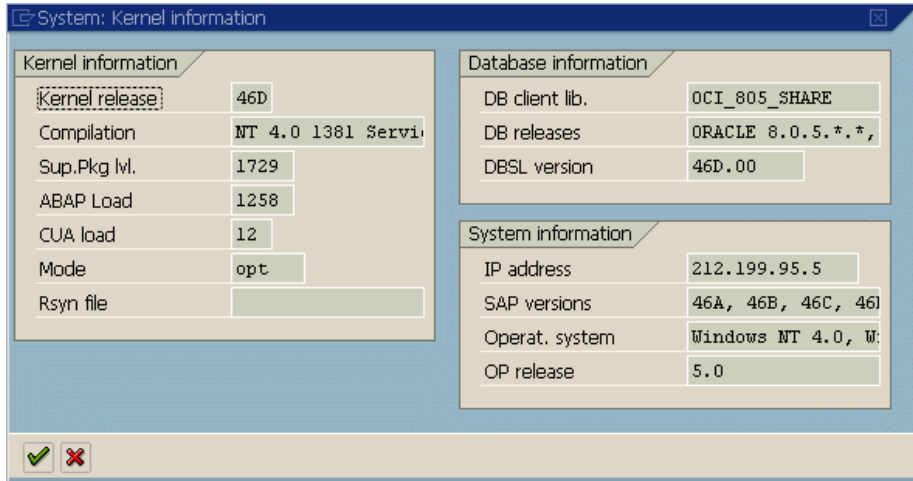
SAP data	
Repository data	
Transaction	PA30
Program (screen)	MP000600
Screen number	2001
Program (GUI)	MP000600
GUI status	DIS
SAP System data	
Component version	R/3 Release 4.6C
Installation number	0120033759
License expiry date	31.12.9999

Host data		Database data	
Operating system	Windows NT	System	ORACLE
Machine type	2x Intel 8	Release	8.1.7.0.0
Server name	calderone_MI6	Name	MI6
Platform ID	560	Host	CALDERONE
		Owner	SAPR3

At the bottom of the dialog box, there is a 'Navigate' button with a green checkmark icon, a green arrow icon, and a red 'X' icon.



- 3 Click the **Other kernel information** button. The System: Kernel information dialog box opens.



- 4 In the **Kernel information** section, check the value of the **Sup. Pkg. lvl.**

If the level is lower than the required level for the SAP release you are using (see "Checking Package and Patch Versions Installed on the SAP Application Server" on page 316), you must download the latest kernel version and upgrade your existing one.


For more information on downloading and installing the required kernel patch level, see SAP OSS note #480149.

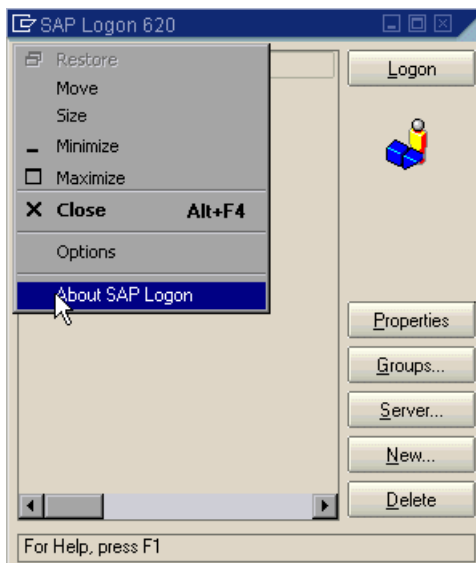
Checking the Patch Version Installed on your SAP GUI for Windows Application

If you want to test your SAP GUI for Windows application with the QuickTest Professional Add-in for SAP Solutions, make sure that the minimum required patch level is installed. For more information on required patch levels, see the *HP QuickTest Professional Product Availability Matrix*, available from the Documentation Library Home page or the root folder of the QuickTest Professional DVD.

Note: If the minimum required patch level is not installed, an error message displays when you try to record on your SAP GUI for Windows application.

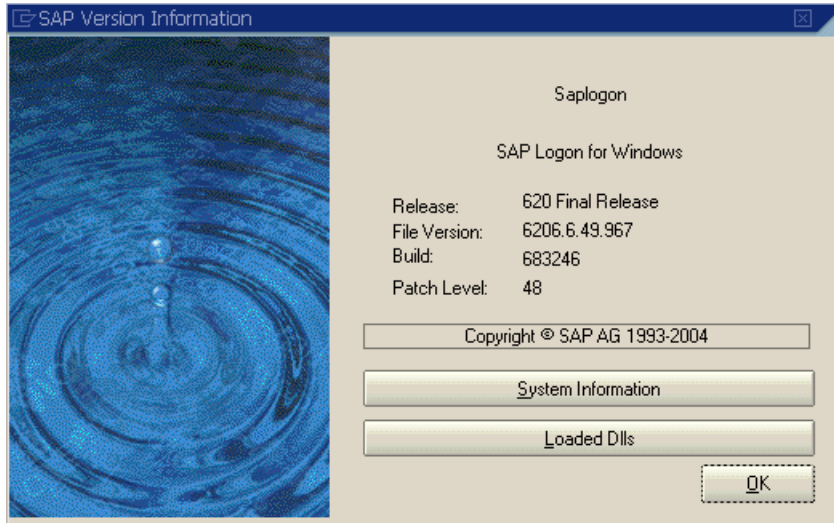
To check the patch level on your SAP GUI for Windows application:

- 1 Open the SAP Logon dialog box and click the  button on the left side of the SAP Logon dialog box's title bar. Then select **About SAP Logon** from the menu.



The SAP Version Information dialog box opens.

- 2 In the SAP Version Information dialog box, confirm that the minimum required patch level is installed.



Enabling Scripting on the SAP Application (Server-Side)

After you confirm that you have the proper support package and kernel patch levels installed, you must enable scripting on your SAP application. By default, scripting is disabled.

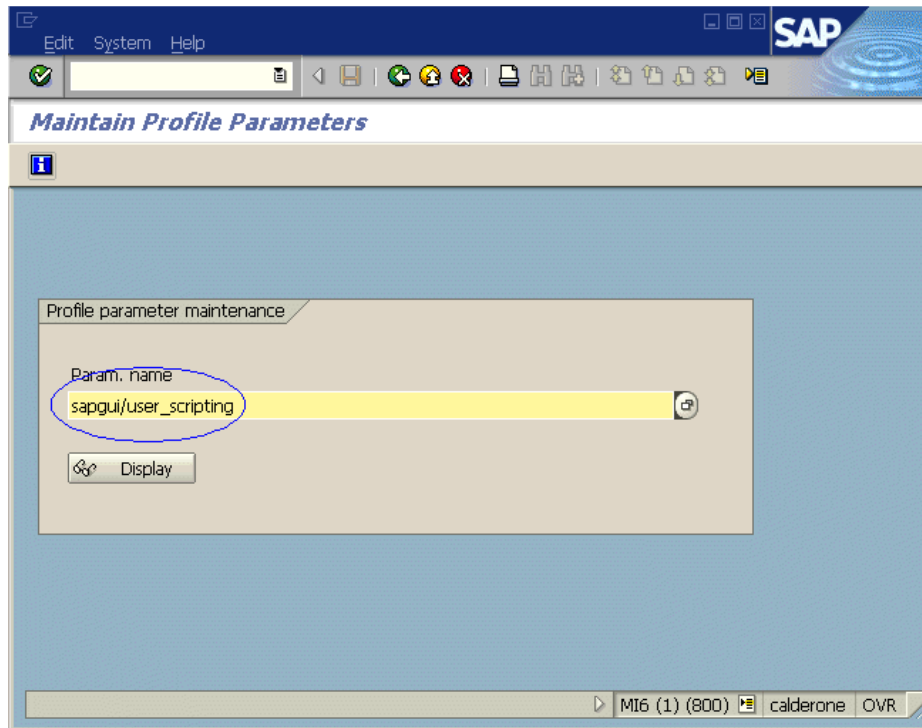
You enable scripting by entering the Maintain Profile Parameters window with administrative permissions and setting the *sapgui/user_scripting* profile parameter to TRUE on the application server.

To enable scripting for all users, set this parameter on all application servers. To enable scripting for a specific group of users, set the parameter only on application servers with the appropriate access restriction settings.

Note: If you connect to a server on which scripting is disabled, an error message displays when you try to record on your SAP GUI for Windows application.

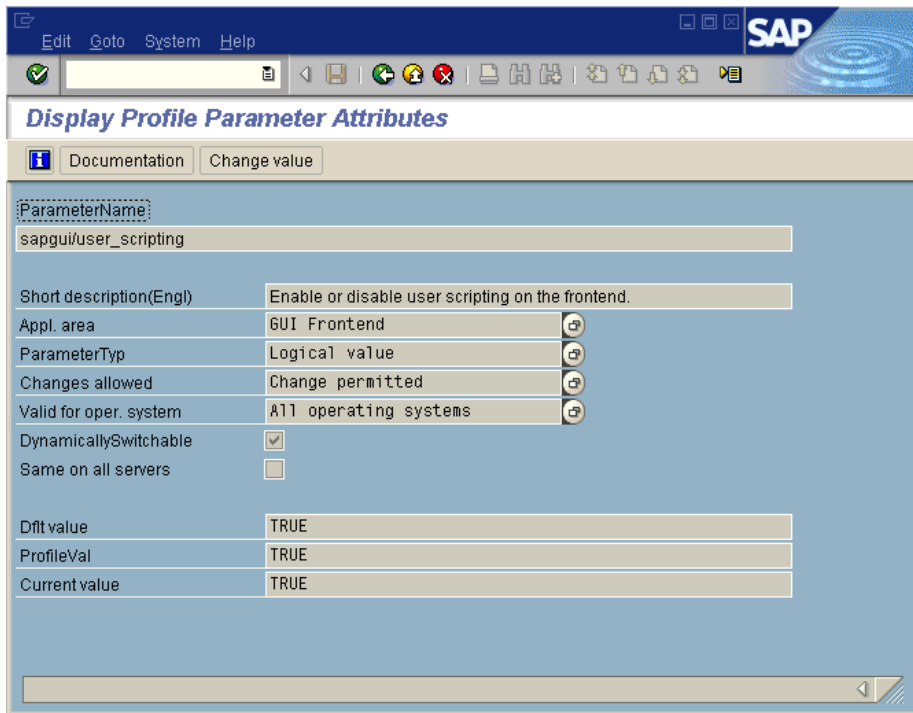
To change the profile parameter:

- 1** Enter `/nrz11` in the OKCode edit box to open transaction **rz11**.
- 2** In the **Param. Name** box of the Maintain Profile Parameters window, enter `sapgui/user_scripting` and click the **Display** button.



Note: If the message **Parameter name is unknown** is displayed in the status bar, your client lacks the required support package (see "Checking the Patch Version Installed on your SAP GUI for Windows Application" on page 321). Download and install the support package that corresponds to the SAP release you are using and then repeat steps 1 and 2.

The Display Profile Parameter Attributes window opens.



- 3** If **ProfileVal** is **FALSE**, you must modify its value. To modify it, click the **Change value** button. The Change Parameter Value window opens.



- 4** Enter **TRUE** in the **New value** box and click the **Save** button.

Note: You must enter **TRUE** in all capital letters. Entering **True** or **true** has no effect.

When you save the change, the window closes and the value of the parameter is displayed as **TRUE**. However, this change takes effect only when you log on to the system. Therefore, before beginning to work with the QuickTest Professional Add-in for SAP Solutions, you must log off and log on again. You may also need to restart the SAP Service from the SAP Console.

If you find that even after restarting the SAP Service from the SAP Console and logging on again to the client, your change to the ProfileVal parameter was not saved, you may have an outdated kernel version. In this case, either restart the application server or download and import the required kernel patch, as specified below.

Release	Kernel Version	Patch Level
6.10	6.10	391
6.20	all versions	all levels
6.40	all versions	all levels
7.10	all versions	all levels

For more information and download guidelines, see SAP OSS note # 480149.

Enabling Scripting on the SAP Application (Client-Side)

To test SAP GUI for Windows applications with QuickTest Professional, you must confirm that scripting is enabled on the SAP GUI for Windows client.

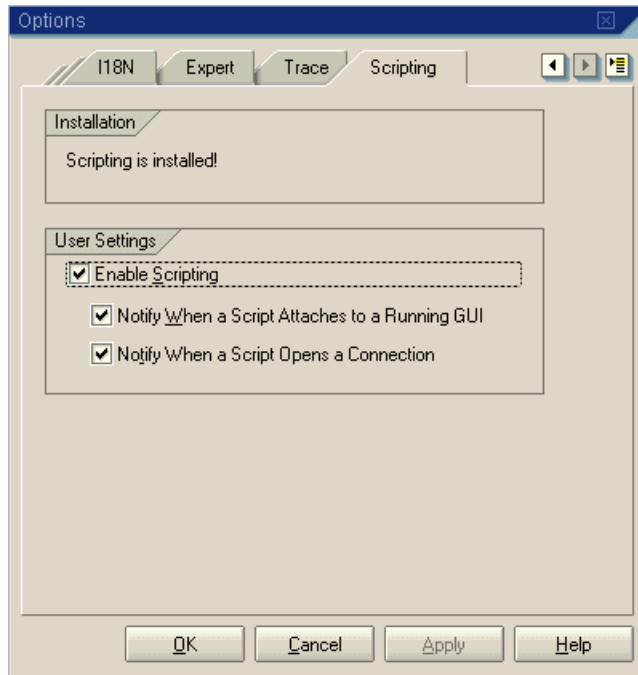
It is also recommended to disable warning messages in the SAP GUI for Windows environment when working with QuickTest Professional.

To ensure that scripting is enabled on the SAP GUI for Windows client:

- 1 Log on to your SAP server.
- 2 Click the **Customize Local Layout** SAP toolbar button and then select **Options**. The Options dialog box opens.

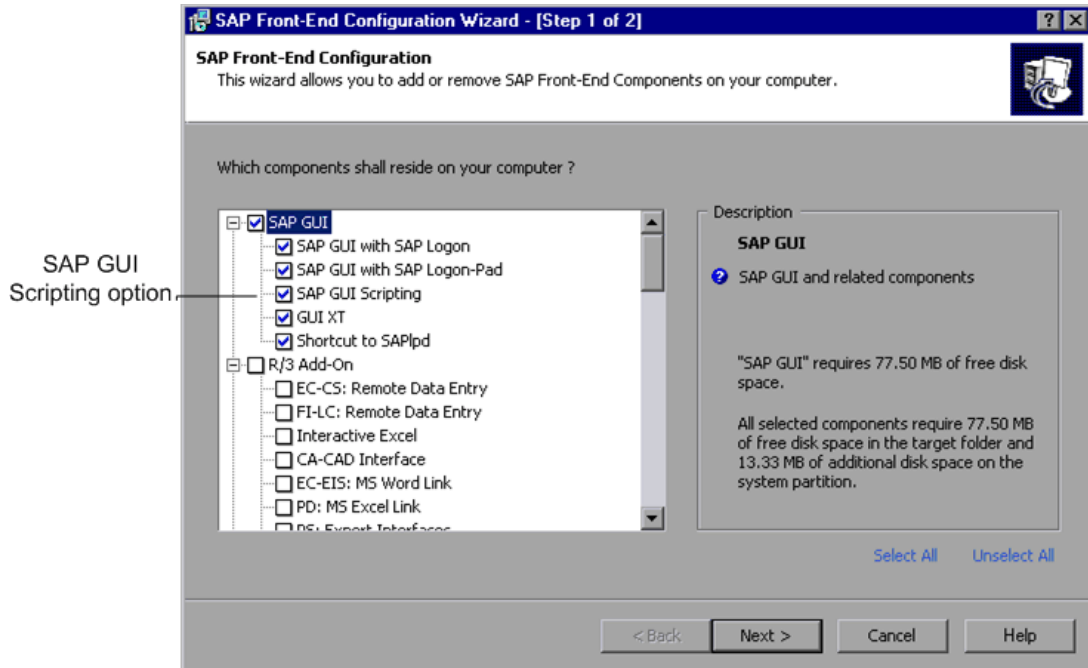


3 Click the **Scripting** tab.



4 Confirm that the **Enable Scripting** check box is selected. If the **Enable Scripting** check box is cleared, select it.

- 5 If the **Scripting is installed** message is not displayed in the **Installation** area, or the **Enable Scripting** check box is disabled, then the **SAP GUI Scripting** option is not installed. Reinstall your SAP GUI for Windows application and be sure to select the **SAP GUI Scripting** check box.



Eliminating Warning Messages

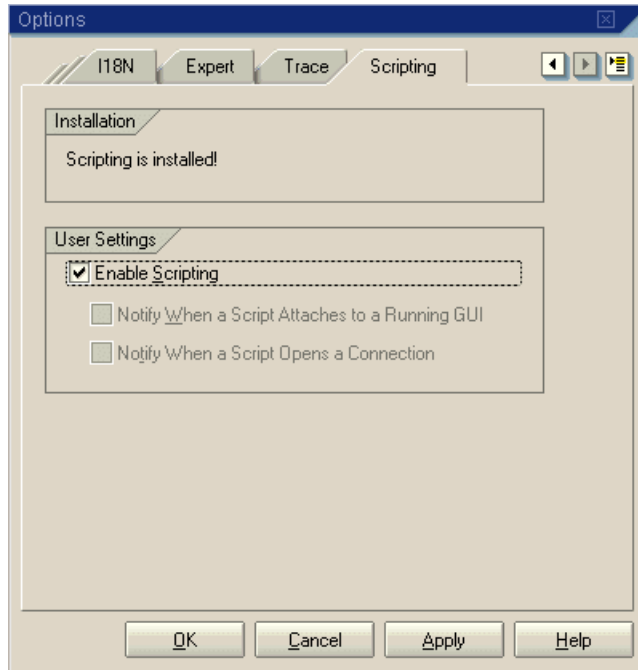
By default, you regularly receive two warning messages when using QuickTest Professional with an SAP GUI for Windows application:

- ▶ When QuickTest Professional connects to the Scripting API, the following warning message is displayed: A script is trying to attach to the gui.
- ▶ When QuickTest Professional opens a new connection using the Scripting API, the following warning message is displayed: A script is opening a connection to system <system_name>.

It is recommended to disable these warning messages in the SAP GUI for Windows application when working with QuickTest Professional.

To eliminate the display of warning messages:

- 1 Log on to your SAP server.
- 2 Click the **Customize Local Layout** SAP toolbar button and then select **Options**. The Options dialog box opens.
- 3 Click the **Scripting** tab.



- 4 Clear the **Notify When a Script Attaches to a Running GUI** and **Notify When a Script Opens a Connection** check boxes and click **OK**.

Setting F4 Help to Use Dialog Display Mode

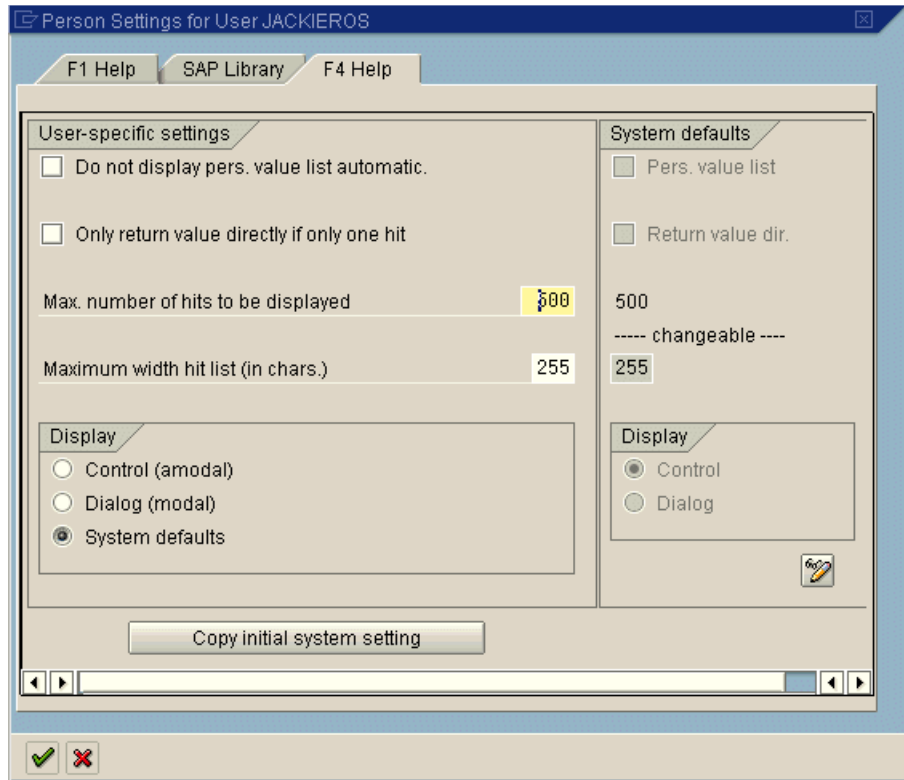
When the SAP GUI for Windows application uses the SAP GUI Scripting API (Enable Scripting option), it cannot load the F4 Help screens in **Control** mode. Therefore, you must ensure that your client is set to load F4 Help screens in **Dialog** mode.

Note: This is a per-user setting. You must set this option on each client that you want to test using the QuickTest Professional Add-in for SAP Solutions. Alternatively, the system default can be changed by the SAP system administrator.

To set F4 Help to use Dialog mode:

- 1 Log on to your SAP server.
- 2 Select **Help > Settings** from the SAP menu bar. Your Person Settings dialog box opens.

3 Click the **F4 Help** tab.



4 In the **System defaults** section on the right side of the tab, view the **Display** setting. This setting indicates the default server setting for all clients.

- ▶ If the **Display** setting is **Dialog**, then you can set your **Display** settings in the **User-specific settings** section on the left side of the tab to **System defaults** or **Dialog (modal)**.
- ▶ If the **Display** setting is **Control** (as in the example above), then you must change the **Display** setting in the **User-specific settings** section on the left side of the tab to **Dialog (modal)**.



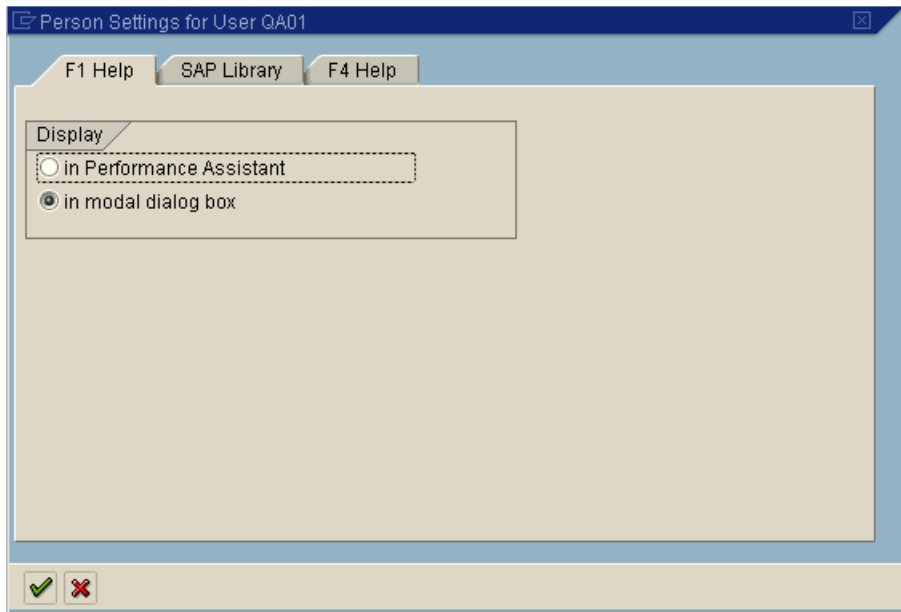
5 Click the **Copy (Ctrl + S)** button to save your changes and close the dialog box.

Setting F1 Help to Use Modal Dialog Box Mode

The F1 Help in your SAP GUI for Windows application can be displayed using either the Performance Assistant or as a modal dialog box. QuickTest Professional can record the displaying of the F1 Help only if the modal dialog box option is selected. If you want to include F1 Help access in your tests, you should select the **in modal dialog box** option.

To set F1 Help to use modal dialog box mode:

- 1 Log on to your SAP server.
- 2 Select **Help > Settings** from the SAP menu bar. Your Person Settings dialog box opens.
- 3 Click the **F1 Help** tab.



- 4 In the **Display** section, select **in modal dialog box**.



- 5 Click the **Copy (Ctrl + S)** button to save your changes and close the dialog box.

Checking the Connection Speed on the SAP Server

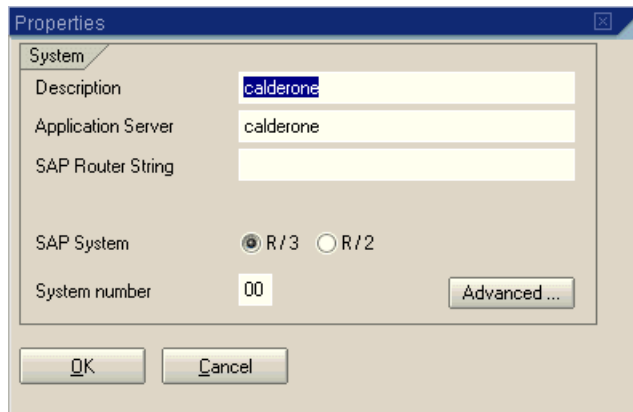
When you log on to SAP using the **Low speed connection** option to communicate with the server, the SAP server does not send sufficient information for QuickTest to properly record and run tests. (QuickTest displays an error message if the **Low speed connection** option is selected.) Therefore, confirm that this option is not selected for the server to which you are connecting before recording and running QuickTest tests.

For more information, see SAP OSS note #587202.

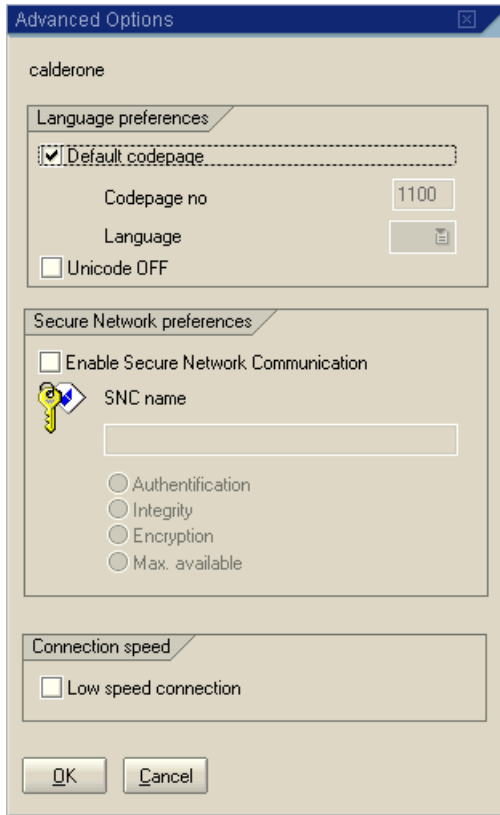
Note: Depending on the SAP GUI for Windows version you are working with, the dialog boxes shown in this section may or may not appear the same as those on your screen. However the instructions in this section are the same for all supported SAP GUI for Windows versions.

To check the connection speed setting on your SAP GUI for Windows client:

- 1** Open the SAP Logon dialog box and select the server to which you want to connect.
- 2** According to the version you are using, either click the **Properties** button or right-click a server and select **Properties**. The Properties dialog box for the selected server opens.



- 3 Click the **Advanced** button. The Advanced Options dialog box opens.



- 4 In the **Connection speed** section, confirm that the **Low speed connection** check box is cleared.
- 5 Repeat steps 1 to 4 for each server you want to use in conjunction with QuickTest.

23

Using the Add-in for SAP Solutions on SAP GUI for Windows Applications

You can use the QuickTest Professional Add-in for SAP Solutions to test SAP GUI for Windows objects (controls).

The QuickTest Professional Add-in for SAP Solutions has been certified by SAP AG.

For details on supported SAP GUI for Windows environments, see the **Add-in for SAP Solutions** section of the *HP QuickTest Professional Product Availability Matrix*, available from the Documentation Library Home page or the root folder of the QuickTest Professional DVD.

The Add-in for SAP Solutions provides test objects, methods, and properties that can be used when testing objects in SAP GUI for Windows applications. For more information, see the **SAP GUI for Windows** section of the *HP QuickTest Professional Object Model Reference*.

When the Add-in for SAP Solutions is loaded, QuickTest can learn objects and run steps on both Web-based and Windows-based SAP applications. For information on recording and running tests and components on Web-based SAP applications, see "Using the Add-in for SAP Solutions on Web-based SAP Applications" on page 289.

The following table summarizes basic information about the QuickTest SAP GUI for Windows support and how it relates to some commonly-used aspects of QuickTest.

General Information	
Add-in Type	This functions like a Windows-based add-in when testing SAP GUI for Windows applications. Much of its functionality is the same as other Windows-based add-ins. See "Testing Windows-Based Applications" on page 87.
Checkpoints and Output Values	<ul style="list-style-type: none"> ▶ See the sections describing checkpoints and output values in the <i>HP QuickTest Professional User Guide</i> (for tests), and the <i>HP QuickTest Professional for Business Process Testing User Guide</i> (for components). ▶ See "Enhancing Your SAP Windows Test" on page 413. ▶ See "Supported Checkpoints and Output Values Per Add-in" on page 769.
Other	The Add-in for SAP Solutions recognizes special SAP Windows objects such as OKCode edit boxes, table and grid controls, list, and tree controls, and toolbars. See "Adding SAP Windows Statements to Your Test or Component" on page 431.
Prerequisites	
Other	Confirm that you have properly configured your SAP server and client. See "Setting Up Your SAP GUI for Windows Environment" on page 313.
Setting Preferences	
Options Dialog Box	Use the SAP pane. (Tools > Options > SAP node) See "Configuring Testing Options for SAP GUI for Windows Applications" on page 347.
Record and Run Settings Dialog Box (tests only)	Use the SAP tab. (Automation > Record and Run Settings) See "Setting Record and Run Settings for SAP GUI for Windows Tests" on page 342.

<p>Custom Active Screen Capture Settings Dialog Box (tests only)</p>	<p>Use the SAP GUI for Windows section. (Tools > Options > Active Screen node > Custom Level)</p> <p>See the section on the Custom Active Screen Capture Settings dialog box in the <i>HP QuickTest Professional User Guide</i>.</p>
<p>Application Area Settings Dialog Box (components only)</p>	<p>Use the Applications pane. (File > Settings > Applications node)</p> <p>See the section on defining Application Settings for your application area in the <i>HP QuickTest Professional for Business Process Testing User Guide</i>.</p>

This chapter includes:

- ▶ Considerations for Working with the Add-in for SAP Solutions on page 338
- ▶ Understanding QuickTest and the SAP GUI Scripting API on page 339
- ▶ Setting Record and Run Settings for SAP GUI for Windows Tests on page 342
- ▶ Configuring Testing Options for SAP GUI for Windows Applications on page 347
- ▶ Understanding Low-Level or Analog Mode Recording on SAP GUI for Windows on page 359
- ▶ Using Standard Windows Recording Capabilities on page 359
- ▶ Understanding QuickTest-eCATT Integration on page 360
- ▶ Configuring eCATT to Work with QuickTest on page 363
- ▶ Working with eCATT in Standalone Mode on page 365
- ▶ Working with eCATT in Integrated Mode on page 390
- ▶ Troubleshooting and Limitations - SAP Windows on page 408

Considerations for Working with the Add-in for SAP Solutions

When recording and running tests or components on SAP GUI for Windows applications, consider the following:

- ▶ When working in tests, the Record and Run Settings dialog box in QuickTest enables you to specify a server and client to open at the beginning of every test record and run session. The servers available in the dialog box are the same as those available in the SAP Logon Pad and SAP Logon dialog box.
- ▶ When you record a component on an SAP GUI for Windows session, the Record and Run Settings dialog box is not available. Instead, you need to open the SAP session manually or include statements in your component that connect to the SAP server (using the SAPGuiUtil test object).
- ▶ You can also record specific operations in your SAP GUI for Windows Application in Standard Windows Recording mode, if required. For more information, see "Using Standard Windows Recording Capabilities" on page 359.
- ▶ As you record a test or component on your SAP GUI for Windows application, QuickTest records the operations you perform. QuickTest works directly with the SAP GUI Scripting API to record your operations. Therefore, although QuickTest records a step for each operation you perform, it adds the steps to your test only when API events are sent to QuickTest (when information is sent to the SAP server).

For more information on the SAP GUI Scripting API events, see your SAP documentation.

- ▶ When you select a test step in QuickTest, the corresponding object is highlighted in the Active Screen (unless you chose not to capture Active Screen information when you recorded your test). However, the values of the object properties stored with the Active Screen are the values of the properties at the time that the steps were added to the test (when you performed the step that sent information to the SAP server). These values may potentially be different from the values of the properties at the time that the selected step was actually performed. For more information on Active Screen capture levels, see the section on the Custom Active Screen Capture Settings dialog box in the *HP QuickTest Professional User Guide*.







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

Understanding QuickTest and the SAP GUI Scripting API

QuickTest works directly with the SAP GUI Scripting API to record your operations. Therefore, QuickTest adds steps to your test or component only when API events are sent to the server. This means that while recording a test or component, you may perform several operations on your application before the corresponding steps are added. When you perform a step that sends information to the server, QuickTest inserts steps with the relevant SAP Windows objects in the Keyword View (tests and components) and adds corresponding statements in the Expert View (tests only).

Example 1

Suppose you record the steps of filling in a Price Simulation for Material form. You select the three check boxes in the form (**Incl. cash discount**, **Delivery costs**, and **Effective price**) and click **Continue**. When you click the **Continue** button, information is sent to the SAP server, and the steps in which you select the check boxes and click the **Continue** button are added to your test at once. In the Keyword View, the process described above is displayed as follows.

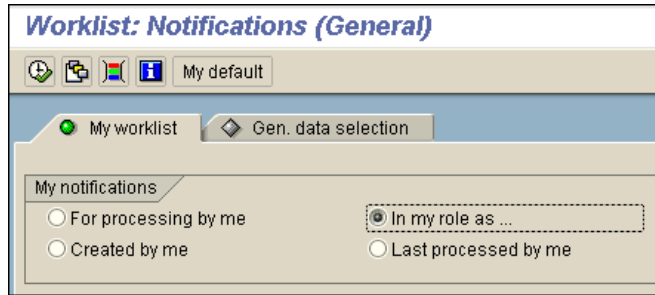
 Price Simulation for Material			
 Incl. cash discount	Set	"ON"	Set the state of the "Incl. cash discount" check box to "ON".
 Delivery costs	Set	"ON"	Set the state of the "Delivery costs" check box to "ON".
 Effective price	Set	"ON"	Set the state of the "Effective price" check box to "ON".
 Effective price	SetFocus		Set the focus on the "Effective price" check box.
 Continue (Enter)	Click		Click the "Continue (Enter)" button.

QuickTest records these steps in the Expert View as follows:

```
SAPGuiSession("Session").SAPGuiWindow("Price Simulation for
Material").SAPGuiCheckBox("Incl. cash discount").Set "ON"
SAPGuiSession("Session").SAPGuiWindow("Price Simulation for
Material").SAPGuiCheckBox("Delivery costs").Set "ON"
SAPGuiSession("Session").SAPGuiWindow("Price Simulation for
Material").SAPGuiCheckBox("Effective price").Set "ON"
SAPGuiSession("Session").SAPGuiWindow("Price Simulation for
Material").SAPGuiCheckBox("Effective price").SetFocus
SAPGuiSession("Session").SAPGuiWindow("Price Simulation for
Material").SAPGuiButton("Continue (Enter)").Click
```

Example 2

Suppose you select a radio button in the **My worklist** tab of your SAP GUI for Windows application. This radio button is labeled **In my role as...**



QuickTest uses the SAP GUI component type (41) to identify the object as an SAPGuiRadioButton object. It creates an SAPGuiRadioButton test object with the name **In my role as...** and records the following properties and values as the description for the radio button.

Type	Property	Value
ABC	guicomponenttype	41
ABC	name	MEL_ROLE
ABC	text	In my role as ...

Note: The **guicomponenttype** and **name** property values are supplied by the SAP GUI Scripting API.

QuickTest also records that you performed a Set method to turn ON the radio button.

QuickTest displays your step in the Keyword View like this:

Worklist: Notifications	Resize	135,25	Resize the "Worklist: Notifications" window to 135 by 25 characters.
In my role as ...	Set		Select the "In my role as ..." radio button.
In my role as ...	SetFocus		Set the focus on the "In my role as ..." radio button.

QuickTest displays your step in the Expert View like this:

```
SAPGuiSession("Session").SAPGuiWindow("Worklist: Notifications").
    SAPGuiRadioButton("In my role as...").Set
```

When you run a test or component, QuickTest identifies each object in your application by its test object class and its *description*—the set of identification properties and values used to uniquely identify the object. In the above example, during the run session, QuickTest looks up the description for the SAPGuiRadioButton object with the name **In my role as...** by searching the object repository. QuickTest finds the following description:

```
guicomponenttype = 41
name = MEL_ROL
text = In my role as...
```

QuickTest then looks in the application for an SAPGuiRadioButton object that matches the above description. When it finds the object, it performs the Set method on it to change the value of the field to ON (selects the radio button).

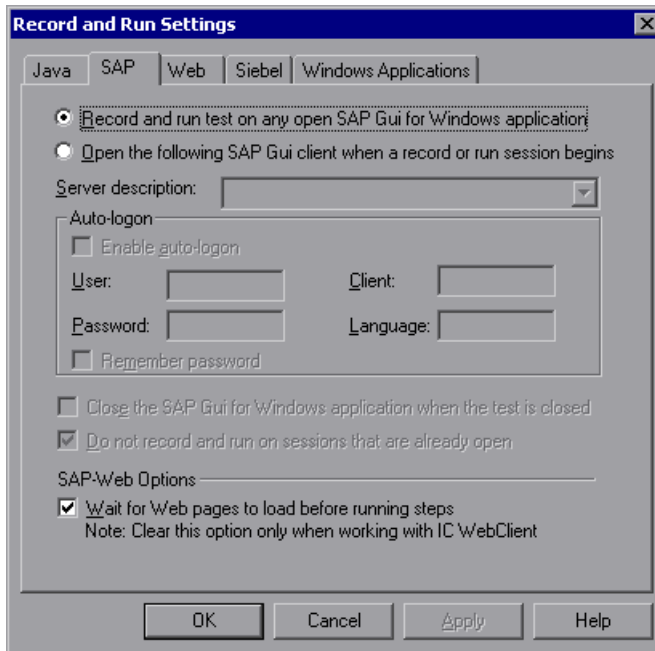
Note: The QuickTest Professional Add-in for SAP Solutions provides an alternative recording mechanism for specific SAP GUI for Windows objects that do not have built-in test object support. For more information, see "SAP GUI for Windows Alternative Recording Mechanism" on page 442.

For more information on the test object model, the object repository, and the Object Spy, see the *HP QuickTest Professional User Guide*.

Setting Record and Run Settings for SAP GUI for Windows Tests

You can use the SAP tab of the Record and Run Settings dialog box to instruct QuickTest to connect to a particular SAP server and open your SAP GUI for Windows application each time you begin a recording session for a test. Alternatively, you can instruct QuickTest to record on any open SAP GUI for Windows application.

If you do not modify the Record and Run settings before you begin recording, the Record and Run Settings dialog box opens automatically when you begin recording a new test (by clicking **Record** (or choosing **Automation > Record**). You can also open it by choosing **Automation > Record and Run Settings**.



If you load only the QuickTest Professional Add-in for SAP Solutions and the Web Add-in, then only the SAP, Web, and Windows Applications tabs are displayed in the Record and Run Settings dialog box. If other add-ins are loaded, the corresponding tabs are also displayed (as shown above).

You can use the SAP tab to instruct QuickTest to connect to a specified SAP server and open your SAP GUI for Windows application using specified user settings. Alternatively, you can instruct QuickTest to record and run the test on any open SAP GUI for Windows application. If you select to connect to a specific server, you can specify details that will enable QuickTest to automatically log on to the server each time a record or run session begins (instead of recording the log on steps).

You can also use application details environment variables to specify these parameters. For more information, see "Defining Application Details Environment Variables" on page 346.

The SAP tab includes the following options:

Option	Description
<p>Record and run tests on any open SAP GUI for Windows application</p>	<p>Instructs QuickTest to use any open SAP GUI for Windows application to record and run the test.</p> <p>This option supports sessions opened using the SAP Logon dialog box or the SAP Logon Pad.</p>
<p>Open the following SAP Gui client when a record or run session begins</p>	<p>Instructs QuickTest to connect to the specified server.</p>
<p>Server description</p>	<p>Indicates the server to which you want to connect. The Server description box lists the servers available in the SAP Logon Pad or the SAP Logon dialog box.</p> <p>To add a server to the list in the Record and Run Settings dialog box, close the Record and Run Settings dialog box, define an appropriate entry using your SAP Logon dialog box, and then reopen the Record and Run Settings dialog box.</p>

Option	Description
Enable auto-logon	<p>Instructs QuickTest to open the specified SAP GUI for Windows application using the specified logon details.</p> <p>Enabled only when Open the following SAP Gui client when a record or run session begins is selected.</p>
User	<p>The user name used to log on to the specified server.</p> <p>Enabled only when Enable auto-logon is selected.</p>
Password	<p>The password for the specified user name.</p> <p>Enabled only when Enable auto-logon is selected.</p>
Client	<p>The client number.</p> <p>Enabled only when Enable auto-logon is selected.</p>
Language	<p>The language that you want the specified SAP GUI for Windows application to display.</p> <p>Enabled only when Enable auto-logon is selected.</p>
Remember password	<p>Saves the password information in this dialog box so that you do not have to enter it each time you begin to record or run the test.</p> <p>Enabled only when Enable auto-logon is selected.</p>

Option	Description
<p>Close the SAP GUI for Windows application when the test is closed</p>	<p>Instructs QuickTest to close the SAP GUI for Windows session specified in the Record and Run Settings dialog box when the test is closed.</p> <p>Any other SAP GUI for Windows session that was opened before, during, or after the run session is not affected.</p> <p>The Session cleanup option in the SAP pane of the Options dialog box (Tools > Options > SAP node) overrides this option. For more information, see "Configuring Testing Options for SAP GUI for Windows Applications" on page 347.</p>
<p>Do not record and run on sessions that are already open</p>	<p>Instructs QuickTest not to record or run tests on any SAP GUI for Windows sessions that were already open prior to the start of the record or run session. This is to ensure that steps are not inadvertently recorded on other SAP GUI for Windows sessions that may also be running on the same computer.</p>
<p>Wait for Web pages to load before running steps (This option is only for SAP Web applications)</p>	<p>Instructs QuickTest to wait for Web pages to synchronize completely before starting the test run.</p> <p>Note: This option is selected by default. The option should be cleared only when working with IC WebClient.</p>

Note: In addition to saving all values set in the Record and Run Settings dialog box with the test, the values you enter for the **User**, **Client**, **Password**, and **Language** in the **Auto-logout** area of the dialog box are saved with the selected server. If you select the same server in the **Server description** box for a new test, the saved values are automatically displayed in the auto-logout area.

For more information on the Record and Run Settings dialog box, see "Using the Record and Run Settings Dialog Box" on page 38.

Defining Application Details Environment Variables

For tests, you can use application details environment variables to specify the applications you want to use during the record and run session. These variables can also be used in external library files for automation scripts.

If you define any of these application details environment variables, they override the values in the **Server description**, **User**, **Password**, **Client**, and **Language** boxes in the SAP tab of the Record and Run Settings dialog box. For more information, see "Setting Record and Run Settings for SAP GUI for Windows Tests" on page 342.

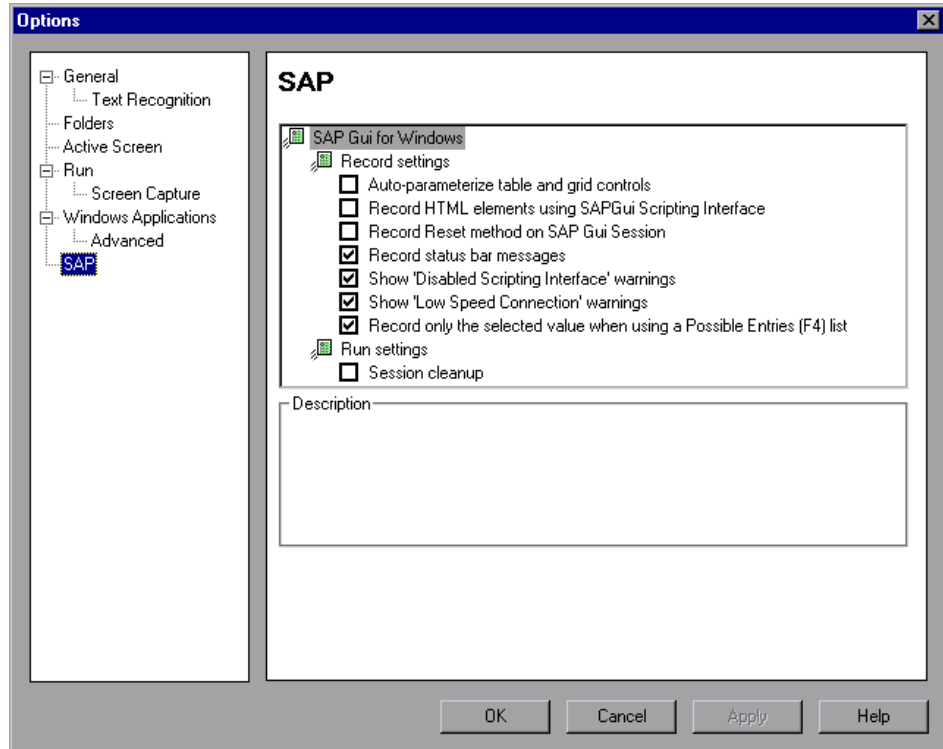
Use the variable names listed in the table below to define SAP application details:

Option	Variable Name	Description
Server description	SAP_SERVER_ENV	The description of the server to which you want to connect.
User	SAP_USERNAME_ENV	The user name used to log on to the specified client number.
Password	SAP_PASSWORD_ENV	The encrypted password for the specified user name.
Client	SAP_CLIENT_ENV	The client number.
Language	SAP_LANGUAGE_ENV	The language that you want the specified SAP GUI for Windows application to display.

For more information on defining and working with environment variables, see the *HP QuickTest Professional User Guide*.

Configuring Testing Options for SAP GUI for Windows Applications

The SAP pane of the Options dialog box (**Tools > Options > SAP** node) enables you to configure how QuickTest records and runs tests and components on SAP applications.



Note: The SAP pane is available only when the QuickTest Professional Add-in for SAP Solutions is installed and loaded.

The options in this pane apply only to steps performed on SAP GUI for Windows applications.

The SAP pane contains the following options:

Option	Description
Auto-parameterize table and grid controls	While recording tests, automatically captures the data you set in table and grid cells and stores it in a new data sheet in the Data Table. QuickTest inserts an Input statement into your test, which refers to the new data sheet. Using this option enables you to set values of multiple cells in a single test step and easily parameterize the cell values. For more information, see "Using the Auto-Parameterize Option to Parameterize Table and Grid Cell Values" on page 351.
Record HTML elements using SAPGui Scripting interface	Specifies whether QuickTest should use the SAP GUI Scripting API when recording HTML elements within SAP applications, or use the built-in Web support to record these HTML elements. You can use this option to handle synchronization issues that may arise from recording on Web elements inside an SAP GUI for Windows session. Changes to this option take effect only after you start recording a new test or component.
Record Reset Method on SAP Gui Session	Records a Reset method as the first step when recording a test or component. When the test or component is run, the first step resets the application session so that it starts at the initial SAP screen. This ensures that the test or component starts with the same application state each time it is run. This option is applicable only if the Open the following SAP Gui client when a record or run session begins and the Auto-logon options are both selected in the SAP tab of the Record and Run Settings dialog box.
Record status bar messages	Records a SAPGuiStatusbar.Sync step each time a status bar message is displayed in the SAP application. Note: This option is selected by default.

Option	Description
<p>Show 'Disabled Scripting Interface' warnings</p>	<p>Instructs QuickTest to display warnings if the SAP GUI Scripting API is disabled on the SAP application. If this is the case, you cannot record or run steps until you enable the SAP GUI Scripting API. For information on enabling the SAP GUI Scripting API, see "Enabling Scripting on the SAP Application (Client-Side)" on page 326.</p> <p>Note: This option is selected by default.</p>
<p>Show 'Low Speed Connection' Warnings</p>	<p>Instructs QuickTest to display warnings if the connection speed to the server is set to Low speed connection.</p> <p>Note: This option is selected by default.</p> <p>If this option is selected, one of the following occurs if the connection speed is low:</p> <ul style="list-style-type: none"> ▶ If the session was opened by QuickTest when recording started (as defined in the Record and Run Settings dialog box), the error message appears and the recording stops. ▶ If the session was opened by the user before recording started, the error message appears and recording continues in Standard Windows Recording mode. For information on Standard Windows Recording mode, see "Using Standard Windows Recording Capabilities" on page 359. <p>If this option is not selected, one of the following occurs if the connection speed is low:</p> <ul style="list-style-type: none"> ▶ If the session was opened by QuickTest when recording started (as defined in the Record and Run Settings dialog box), the error message does not appear and the recording stops. ▶ If the session was opened by the user before recording was begun, the error message does not appear and recording continues in Standard Windows Recording mode. For information on Standard Windows Recording mode, see "Using Standard Windows Recording Capabilities" on page 359. <p>The connection speed can be checked using the SAP client. For more information, see "Checking the Connection Speed on the SAP Server" on page 333.</p>

Option	Description
<p>Record only the selected value when using a Possible Entries (F4) list</p>	<p>Specifies that only the selected value is recorded when using a Possible Entries list. Any other actions performed on any windows opened after pressing F4 (or after clicking the icon in the specific field) are ignored, and only the actual change made to the field is recorded.</p> <p>Note: An event is received only on the field in focus when F4 was pressed, and not on all the populated fields in the screen. For this reason it may be preferable not to select this option when recording.</p>
<p>Session cleanup</p>	<p>Instructs QuickTest to close all SAP GUI for Windows sessions opened by QuickTest during the current run session when the test is closed. This includes all SAP GUI for Windows sessions that were invoked from the Record and Run Settings dialog box, as well as any sessions that may have been invoked during the run session using a SAPGuiUtil statement or the Open New Session button in the SAP GUI for Windows application that was being recorded.</p> <p>SAP GUI for Windows sessions that were opened during a previous run session, or opened manually before or during the current run session are not affected.</p> <p>This option overrides the Close the SAP GUI for Windows application when the test is closed option in the SAP tab of the Record and Run Settings dialog box. For more information, see "Setting Record and Run Settings for SAP GUI for Windows Tests" on page 342.</p>

Using the Auto-Parameterize Option to Parameterize Table and Grid Cell Values

When working with tests, QuickTest records a SetCellData statement, by default, each time you modify the value of a cell in a table or grid. If you want to modify the values of several cells in a single table or grid, and then parameterize your test so that different values are entered into the cells each time your test action runs, you can do this by parameterizing each statement individually, or by enabling the **Auto-parameterize table and grid controls** option.

When this option is selected, QuickTest automatically captures all values you set for a particular table or grid during a recording session and stores them in a special data sheet in the Data Table. QuickTest inserts a single **SAPGuiTable.Input**, **SAPGuiGrid.Input**, or **SAPGuiAPOGrid.Input** statement into your test, which refers to this new data sheet. Before running the test, you can easily modify the values or add additional sets of data to the data sheet for each action iteration.

To record in auto-parameterize mode:

- 1 Select **Tools > Options** and click the **SAP** node.
- 2 Select the **Auto-parameterize table and grid controls** option.
- 3 Begin a recording session.
- 4 Set the value of one or more cells in a table or grid.
- 5 Press the ENTER key or perform another operation that sends data to the SAP server.

For information on auto-parameterization and the Input statement, see:

- "Understanding How QuickTest Records in Auto-Parameterize Mode" on page 352
- "Parameterizing Cell Values in the Input Data Sheet" on page 355
- "Working with Auto-Parameterization - Tips and Guidelines" on page 356
- "Entering Data in Rows Requiring Scrolling" on page 358

Understanding How QuickTest Records in Auto-Parameterize Mode

In tests, when you record with the **Auto-parameterize table and grid controls** option and you perform an operation that sends data to the SAP server after setting table or grid cell values, QuickTest:

- Creates a new data sheet to represent the table or grid. Each data sheet is a sub-sheet of the action in which the table or grid operations were recorded. The data sheet name is always the action name followed by a period (.) and the internal name of the table or grid. For example: Action1.FLIGHT_TABLE

- Adds a column to the data sheet for each table or grid column in which you record. (Columns in which you did not set any cell data are not added to the data sheet.)

The name of the column in the data sheet is generally the same as the name of the column in your application.




If a column in the application does not have a header, or more than one column header has the same name, QuickTest inserts a column with a name in the format: `__<index>`, where `<index>` represents the column number according to its location when you record the Input step.

- Inserts the values you set during the recording session into the appropriate cells in the data sheet. Each row in which you entered data is represented by a row in the data sheet. Place-holder (empty) rows are added for rows above the rows in which you recorded. For example, if you set data in rows 2, 4, and 7, seven rows are added to the data sheet. The cells in rows 1, 3, 5, and 6 do not contain any data.
- Inserts an additional **end row** where the value of the first cell in the row is `.END.`

	Airline_carrier	Flight_number
1	AB	
2	AB	
3		553
4	END	

End Row —

- Inserts an Input *DataSheetName* statement (followed by a SelectCell statement) into your test.

 SAP	Table control tc spfli	Input	"Action1.Table control tc spfli_3"	Add a sheet named "Ac
 SAP	Table control tc spfli	SelectCell	1,"Airline"	Select the cell in row 1,
 SAP		SendKey	ENTER	Press the ENTER keyb

QuickTest records these steps in the Expert View as follows:

```
SAPGuiSession("Session").SAPGuiWindow("SAP").SAPGuiTable("Table control tc spfli").Input "Action1.Table control tc spfli_3"
```

```
SAPGuiSession("Session").SAPGuiWindow("SAP").SAPGuiTable("Table control tc spfli").SelectCell 1,"Airline"
```

```
SAPGuiSession("Session").SAPGuiWindow("SAP").SendKey ENTER
```

The Input statement instructs QuickTest to enter values from the data sheet into the table or grid corresponding to the data sheet name, similar to an automatically parameterized statement referring to a special sheet in the Data Table.

Suppose you update values in a table control containing airline flight information. You update some airline codes, add state and country names to some of the departure and destination cities, update one of the destination airport codes, and update some of the departure times. The edited table in your application may look something like this:

Ai...	Fli...	Depart.city	D...	Destination cit	D...	Flight ti...	Departur
AB	17	NEW YORK CITY	LGA	SAN FRANCISCO, CA	SFO		13:30:0
AB	64	SAN FRANCISCO, CA	SFO	NEW YORK	JFK		09:35:0
AZ	553	ROME, ITALY	FCO	FRANKFURT	FRK		19:00:0
AZ	788	ROME, ITALY	FCO	TOKYO, JAPAN	TYO		12:00:0
AZ	789	TOKYO, JAPAN	TYO	ROME, ITALY	FCO		11:45:0
AZ	790	ROME, ITALY	FCO	OSAKA	KIX		10:35:0

QuickTest inserts the following Input statement in your test to represent the data input:

```
SAPGuiSession("Session").SAPGuiWindow("SAP R/3").SAPGuiTable("SPFLI").
    Input "Action1.SPFLI"
```

Note: If you record on a table or grid that scrolls using the ENTER key rather than the PAGEDOWN key, you may need to manually add the ScrollMethod optional argument. For more information, see "Entering Data in Rows Requiring Scrolling" on page 358.

The corresponding data sheet in your Data Table looks like this:

	Airline_carrier	Depart.city	Dep. airport	Destination_cit	Dest. airport	Departure
1	AB	NEW YORK CITY	LGA	SAN FRANCISCO		
2	AB	SAN FRANCISCO, CA				09:35:00
3		ROME, ITALY			FRK	
4		ROME, ITALY		TOKYO, JAPAN		
5		TOKYO, JAPAN		ROME, ITALY		
6		ROME, ITALY				
7	END					
8						

There are six rows in the data sheet, because data was modified in the first six rows of the table or grid in the application. Note that the data sheet does not contain columns for the **Flight Number** and **Flight time** columns, because no values were modified in those columns during the recording session.

Parameterizing Cell Values in the Input Data Sheet

When working in tests, after you record an Input statement to create an input data sheet, you can modify the values to be used in the run session, and you can create multiple sets of table or grid cell data to be used in different iterations of an action.

As described above, when you record the Input statement, QuickTest records the values you set in the appropriate rows and columns in the input data sheet for that table or grid. Below the data it adds an end row (shaded in blue) with the text `.END` in the first cell of the row. This row indicates the end of the first set of data for the table or grid. This set of data and its corresponding end row represents a single *data set*.

To supply different data values for each action iteration, you add new data sets. You add a new data set for a table or grid by entering the values in the appropriate rows and columns below the previous end row. To indicate the end of the new data set, copy and paste the end row from the first set of data to the row below the new set of data. You can include a different number of rows in each data set.

Note: The Input statement can run successfully only if it can find the end row. Therefore, the first cell of the end row must contain only the text `.END`. You can enter text into other cells in that row, if needed. For example, you can enter a number in the second cell of the end row to indicate the iteration number corresponding to that set of data.

Because the input data sheets are added as a sub-sheet of the current action, the Input statement uses the data set corresponding to the current action iteration. For example, if you set the action to run on all iterations and your action sheet includes five rows of data, then your input data sheet should also include five data sets (and five `.END` rows).

The input data sheet below contains three sets of data. The first set contains data for the top three rows of the table or grid. The second set contains data for the top two rows of the table or grid. The third set contains data for rows 2-5. The blank first row (row 8 in the data sheet), indicates that no data should be entered or modified in the first row of the table or grid.

Note that a number was manually entered into the second cell of each **END** row to make it easier to identify the action iteration to which each data set corresponds.

Manually added numbers indicate the iteration that corresponds to each data sheet

	Airline_carrier	Flight_number	Depart.city	Dep._airport
1	AB		NEW YORK CITY	LGA
2	AB		SAN FRANCISCO, CA	
3		553	ROME, ITALY	
4	END	1		
5	AA		551 NYC	LGA
6		552	ORLANDO, FL	
7	END	2		
8				
9	QR		123 LOUISVILLE, KY	SDF
10	ST		101 MADISON, WI	
11	YY		110 BALTIMORE, MD	BWI
12		552	ORLANDO, FL	
13	END	3		

Working with Auto-Parameterization - Tips and Guidelines

Consider the following tips and guidelines when using the **Auto-parameterize table and grid controls** option:

- QuickTest inserts an Input statement and a new input data sheet each time information including modified table or grid cell data is sent to the server. If you set data in the cells of a particular table or grid both before and after sending information to the server, you will have more than one input data sheet (and more than one Input statement) representing the same table or grid.

Note:

It is recommended to enter data only in the visible rows of the table or grid while recording, especially if scrolling results in sending information to the server. You can add additional rows to the recorded data set while editing your test.

It is also recommended to perform sorting, calculations, and other such operations either before beginning or after you finish entering data in a table or grid.

- The end of each data set in the input table or grid must be indicated by an end row with only the text .END in the first cell of the row.
- You can enter additional text, such as comments or an iteration number, in other cells of the .END row.
- You can include a different number of rows in each data set.

If you enter data for rows that require scrolling to display them in your application, you may need to modify your Input statement. For more information, see "Entering Data in Rows Requiring Scrolling" on page 358.

- When recording, QuickTest adds a column to the input data sheet only for table or grid columns in which you set data. You can add additional columns from your table or grid to the data sheet while editing your test. Double-click the column header in the data sheet to rename it. Enter the name of your table or grid column. If the table or grid column name has spaces, replace the spaces with underscores.
- In general, the columns in your data sheet can be in any order, as long as the column names match the column names in your table or grid. However, if you record data in a column without a column header name or if more than one column in the table or grid has the same header name, QuickTest adds a column to the data sheet in the format: `_
_<index>`, where `<index>` indicates the number of the column in the table or grid when you record the Input statement, for example, `__1` or `__2`. You can also use this format for columns in the data sheet if the column header names in your table or grid may change from iteration to iteration.
- To use multiple sets of data from an input data sheet, you must have at least one other Data Table parameter in your action that is set to use **Current action sheet (local)**. Also, confirm that the action is set to run multiple iterations in the Run tab of the Action Properties dialog box.
- The number of data sets in your input data sheet should match the number of rows in the corresponding action data sheet.

If your input data sheet contains fewer data sets than the number of rows in the action sheet, no data will be inserted in the table or grid during those action iterations. For example, if the action runs five iterations, and your input data sheet contains only four data sets, during the fifth iteration no data will be entered into the table or grid when the Input statement runs.

If your data sheet contains more data sets than the number of rows in the action sheet, those data sets will not be used.

Entering Data in Rows Requiring Scrolling

When working in tests, QuickTest inserts a new Input statement and creates a new input data sheet each time you send information to the server that includes table or grid cell data. Therefore, if scrolling results in sending data to the server, it is recommended to add data only to visible cells during the recording session. If you want to enter data into additional rows during the run session, you can add those rows to the data sheet manually while editing your test.

If you create an input data set for rows that are not visible on the table or grid in your application, then QuickTest must scroll the table or grid during the run session to insert the data for those rows. If you create an input data set for a row that needs to be added to the table or grid, QuickTest must send a command to add the row. By default, QuickTest sends a `PAGEDOWN` command if the rows in the data sheet exceed those currently displayed in the application. If QuickTest needs to use the `ENTER` key to add additional rows to the table or grid, then you need to manually add the optional *ScrollMethod* argument (with the value `ENTER`) to your Input statement before running your test. For example:

```
SAPGuiSession("Session").SAPGuiWindow("Create Standard").  
    SAPGuiTable("SAPMV45ATCRTL_V_ERF_").  
    Input "Action1.All items", ENTER
```

Understanding Low-Level or Analog Mode Recording on SAP GUI for Windows

When working in tests, if you are unable to record on an object in the normal recording mode, or if you want to record mouse clicks and keyboard input with the exact x- and y-coordinates, you can record on those objects using low-level or analog recording (select **Automation > Low Level Recording** or **Automation > Analog Recording** during a recording session).

When recording in one of these modes, your steps are added to your test (or to the analog file) as you record them rather than when information is sent to the server. If you begin recording in low-level or analog mode, do not switch back to the normal recording mode until you perform a step that results in communication with the SAP server. Switching between one of these modes and the normal recording mode before the server communication, may result in your steps being recorded twice (once in low-level/analog mode and once in normal mode).

For more information on low-level and analog recording, see the *HP QuickTest Professional User Guide*.

Using Standard Windows Recording Capabilities

In tests, there are some operations in SAP GUI for Windows applications that open standard Windows controls. To record steps on these controls while recording a test on an SAP GUI for Windows application, you need to switch to Standard Windows Recording mode. If you do not switch to Standard Windows Recording mode, then nothing is recorded in your test when you perform operations on these controls.



To switch to Standard Windows Recording mode while recording a test in an SAP GUI for Windows application, either click the **Standard Windows Recording Mode** button in the Testing toolbar, or select **Automation > Standard Windows Recording**.



To record steps as SAP GUI for Windows objects again, click the **Standard Windows Recording Mode** button in the Testing toolbar, or select **Automation > Standard Windows Recording** to deselect the option.

Note: If you switch to Standard Windows Recording mode after performing an operation on a standard Windows control, in some cases this may cause both QuickTest and the SAP application to become unresponsive. To avoid this, make sure that you switch to Standard Windows Recording mode before you use the standard Windows control in your SAP application.

Understanding QuickTest-eCATT Integration

In addition to Quality Center, HP's Web-based test management tool, you can also store and manage QuickTest tests in the SAP Extended Computer Aided Test Tool (eCATT).

QuickTest Professional Add-in for SAP Solutions is integrated with the SAP Extended Computer Aided Test Tool (SAP eCATT). This integration into SAP eCATT, via SAP GUI for Windows 6.20, has been certified against the SAP Web AS 6.20. Now, using SAP eCATT and QuickTest Professional, customers can run quality tests in environments that span beyond Windows and SAP environments including complex, multi-platform, highly-integrated composite, legacy, and proprietary enterprise applications.

You can use the **SAP eCATT integration** feature only if you already have the SAP GUI for Windows software installed on your computer, including support for RFC libraries. You add support for RFC libraries by selecting the **Unicode RFC Libraries** check box (under **Development Tools**) during the SAP installation.

Note: You cannot use SAP eCATT to manage business components or scripted components.

Managing Tests in eCATT

After creating tests for your SAP application, you can store and manage them in a test management tool. Depending on your needs, you may choose to use HP Quality Center, or SAP eCATT.

For more information on integrating with Quality Center, see the *HP QuickTest Professional User Guide* and your Quality Center documentation.

You must configure your eCATT server to work with QuickTest. After the server is configured, you can connect to eCATT from QuickTest (standalone mode) and you can connect to QuickTest from eCATT (integrated mode). From QuickTest or from eCATT you can create QuickTest tests, store tests and associated resource files in the eCATT database, edit tests, run tests, and review run results. You can also call and pass values from an eCATT test script to a QuickTest test.

Notes:

eCATT support is available only when the SAP Front End software is installed on your computer (including support for Unicode), both the **Add-in for SAP Solutions** and **SAP eCATT integration** components are installed, and the SAP Add-in is loaded. For more information, see "Loading QuickTest Add-ins" on page 28.

You cannot connect to both eCATT and Quality Center simultaneously.

Understanding eCATT Testing Modes

You can work with tests stored in eCATT in standalone or integrated mode. The current mode is indicated in the QuickTest title bar.

- ▶ **Standalone mode.** The test is stored in eCATT, but was opened from within the QuickTest interface.
- ▶ **Integrated mode.** The test was opened for editing or running from within the eCATT interface.

The table below describes the basic differences between these two modes.

	Standalone Mode	Integrated Mode
eCATT-QuickTest connection	Connect to eCATT from QuickTest using the eCATT Connection dialog box.	eCATT automatically establishes the eCATT-QuickTest connection.
Available QuickTest features	All QuickTest features are available. You can open and work with any test in eCATT or in the file system.	You can work only with the currently open test. File > Open, File > New, and the Recent files list options are disabled. If you select File > Save As , QuickTest warns you that it will disconnect from eCATT and switch QuickTest to standalone mode.
Resource files	When you open the test, you can also edit and save all the test's resource files, including those stored in eCATT.	When you open the test, test resources stored in eCATT are opened in read-only mode.
Save location	Tests and uploaded files are automatically saved to the local package (\$TMP) in eCATT.	You can save tests to any package (including non-local packages).
eCATT dependence	Although QuickTest is connected to eCATT, you can work and navigate in eCATT independently.	eCATT is locked while the test is open in QuickTest. To release eCATT, close QuickTest.
Run results	All run results are stored in the file system. They cannot be accessed from your eCATT log list.	Run results are stored to the network drive you specify in the eCATT pane of the Options dialog box. You can access the run results from the eCATT log.

For more information on working in standalone mode, see "Working with eCATT in Standalone Mode" on page 365.

For more information on working in integrated mode, see "Working with eCATT in Integrated Mode" on page 390.

Configuring eCATT to Work with QuickTest

Before you can use the eCATT-QuickTest integration features available with the QuickTest Professional Add-in for SAP Solutions, you (or an eCATT system administrator) must install the appropriate support package and configure the eCATT server to work with QuickTest.

To enable eCATT-QuickTest integration, an SAP user with administrative privileges must:

- ▶ Update the eCATT server with the appropriate support package. Contact SAP or your SAP representative to receive the necessary support package.
- ▶ Set external tool parameters in the ECCUST_ET table.
- ▶ Apply the necessary roles or profiles to each user who wants to work with QuickTest and eCATT.

For information on specific SAP server version and support package requirements, see the *HP QuickTest Professional Product Availability Matrix*, available from the Documentation Library Home page or the root folder of the QuickTest Professional DVD.

Setting External Tool Parameters in the ECCUST_ET Table

To enable eCATT to communicate with QuickTest, you must set certain values for the external tool parameters in the ECCUST_ET table. You perform this procedure only once in the system.

To set parameter values in the ECCUST_ET Table:

- 1** Navigate to transaction **se17**. The General Table Display window opens.
- 2** In the **Table Name** box, enter **ECCUST_ET** and press **ENTER**.
- 3** The Display Table **ECCUST_ET** window opens and displays an empty table with the required parameter names.

4 Enter the values exactly as shown below:

TOOL_NAME	QuickTest Professional
PROG_ID	MERCURY.ECATTAGENT
TOOL_DESC	QuickTest Professional
TOOL_DATABASE	QUICKTEST DATABASE
TOOL_RUN_DB	QUICKTEST RUNTIME DATABASE
TOOL_NO_PWD	X
TOOL_NO_DB	X

Note: You can also use the function module **SET_EXTERNAL_TOOL** to create entries in the customizing table. For more information, see your eCATT documentation.

Applying Necessary Roles or Profiles to eCATT-QuickTest Users

To perform all of the operations described in the following chapters, you must have permission to do the following:

- Run eCATT scripts
- Edit eCATT scripts
- Work with an external tool (QuickTest) in integrated mode
- Connect to eCATT from an external tool (QuickTest) in standalone mode

Each of these tasks requires special roles or profiles. Before you begin working with the QuickTest-eCATT integration, you should confirm with your system administrator that the user name you use is assigned the necessary roles or profiles to perform the above tasks. For example, to work with QuickTest in standalone mode, you must be assigned the role **S_ECET** or the profile **SAP_ECET** in the eCATT system.

For more information, contact your system administrator or see your SAP and eCATT documentation.

Working with eCATT in Standalone Mode

You can connect to an eCATT database from QuickTest. This is called *standalone mode*. When you work in standalone mode, you have access to all standard QuickTest features.

This section includes:

- "Working in Standalone Mode" on page 366
- "Connecting to and Disconnecting from eCATT" on page 367
- "Connecting QuickTest to eCATT" on page 367
- "Saving Tests to eCATT in Standalone Mode" on page 370
- "Understanding the Save QuickTest Test to eCATT Dialog Box" on page 372
- "Opening Tests from an eCATT Database in Standalone Mode" on page 374
- "Understanding the Open QuickTest Test from eCATT Dialog Box" on page 375
- "Uploading Files to eCATT" on page 377
- "Understanding the eCATT Upload File Dialog Box" on page 382
- "Understanding the Save External File to eCATT Dialog Box" on page 383
- "Downloading Files from eCATT" on page 384
- "Setting Options for Working with eCATT" on page 385
- "Configuring the eCATT Trace File" on page 387
- "Passing Values Between eCATT Test Scripts and QuickTest Tests" on page 388
- "Running a Test Stored in an eCATT Database in Standalone Mode" on page 389

Working in Standalone Mode

When you open QuickTest with the **SAP** Add-in loaded, you can connect to eCATT, store tests in the eCATT database, open existing tests from the eCATT database, and upload or download files to or from eCATT. You can store a test's external resource files in eCATT. For example, you can store shared object repository files, Data Table files, library files, environment variable files, and recovery files in your eCATT database. QuickTest provides a special set of eCATT-specific options that enable you to control certain elements of the eCATT-QuickTest integration.

You can also perform many of these operations from the eCATT interface (integrated mode).

You can also download resource files already stored in eCATT and save them in the file system.

For more information on working in integrated mode, see "Working with eCATT in Integrated Mode" on page 390.

For more information on eCATT testing modes, see "Understanding eCATT Testing Modes" on page 361.

Note: You can pass values from an eCATT test script to a QuickTest test, or vice versa, in the form of QuickTest test parameters.

If you want to create tests or actions that you can use for different purposes or in different scenarios based on the data supplied to them, you can take advantage of the **Automatically parameterize steps using Test Parameters** option (in the General Tab of the Options dialog box). This option instructs QuickTest to automatically parameterize all the operation arguments in the steps of one or more actions in your test, at the end of a QuickTest recording session. You can then supply the values for these test parameters from eCATT.

For more information on parameters, see the *HP QuickTest Professional User Guide*.

Connecting to and Disconnecting from eCATT

From QuickTest, you can connect or disconnect to or from eCATT at any time during the testing process. However, you should not disconnect QuickTest from eCATT while a QuickTest test that is stored in eCATT is open or while QuickTest is using a shared resource stored in eCATT (such as a shared object repository or Data Table file).

Connecting QuickTest to eCATT

You connect to eCATT using the eCATT Connection dialog box (**Tools > eCATT Connection**).

The screenshot shows the 'eCATT Connection' dialog box. The title bar reads 'eCATT Connection'. The 'Server description' dropdown is set to 'PIPELINE - eCATT'. Under the 'Server logon' section, the 'User' dropdown is set to 'MERCURY', the 'Password' field contains 'xxxxxxxx', the 'Client' field contains '800', and the 'Language' dropdown is set to 'EN'. There are two unchecked checkboxes: 'Reconnect on startup' and 'Save password for reconnection on startup'. A 'Connect...' button with a green plus icon is located to the right of the checkboxes. At the bottom are 'OK', 'Cancel', and 'Help' buttons.

The eCATT Connection dialog box contains the following options:

Option	Description
Server description	The eCATT server to which you want to connect. The Server description box lists the servers available in the SAP Logon Pad or the SAP Logon dialog box. To add a server to the list in the eCATT Connection dialog box, close the dialog box, define an appropriate entry using your SAP Logon dialog box, and then reopen the eCATT Connection dialog box.
User	The user name used to log on to the specified server.
Password	The password for the specified user name.
Client	The client number.
Language	The language that you want to use.
Reconnect on startup	Instructs QuickTest to automatically reconnect to the eCATT server the next time you open QuickTest.
Save password for reconnection on startup	Instructs QuickTest to save your password for reconnection on startup. If you select Reconnect on startup , but do not select this option, you are prompted to enter it each time QuickTest opens. Enabled only when Reconnect on startup is selected.
Connect	Connects QuickTest to eCATT.

The eCATT icon is displayed in the QuickTest status bar to indicate that QuickTest is currently connected to an eCATT server.



Tip: To open the eCATT Connection dialog box, double-click the **eCATT** icon in the QuickTest status bar.

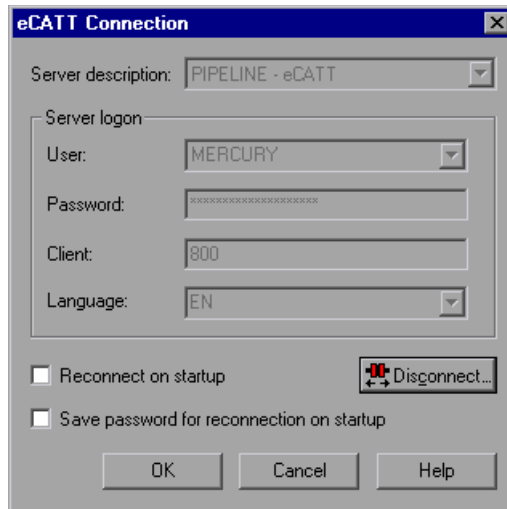
Disconnecting QuickTest from eCATT

When you are finished working with eCATT, you can disconnect from it.

Note: If a test or shared file (such as a shared object repository or Data Table file) that is stored in eCATT is open when you disconnect from eCATT, then QuickTest informs you that the test will be closed.

To disconnect QuickTest from eCATT:

- 1 Select **Tools > eCATT Connection**. The eCATT Connection dialog box opens.



- 2 Click **Disconnect** to disconnect QuickTest from the selected server.
- 3 Click **OK** to close the eCATT Connection dialog box.

Saving Tests to eCATT in Standalone Mode

When QuickTest is connected to an eCATT server in standalone mode, you can create new tests in QuickTest and save them directly to your eCATT database. You can also open, edit, and save existing tests that are stored in eCATT, and you can save existing tests with a new name in the eCATT database or in the file system.

Note: When saving a test to eCATT in standalone mode, it is automatically saved to the local package (**\$TMP**) in eCATT.

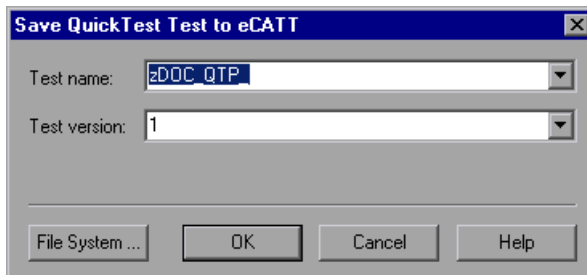
To save a test to an eCATT database in standalone mode:

- 1 From QuickTest, connect to an eCATT server. QuickTest connects to eCATT in standalone mode. For more information, see "Connecting QuickTest to eCATT" on page 367.



- 2 In QuickTest, click **Save** or select **File > Save** to save the test.

The Save QuickTest Test to eCATT dialog box opens. If you defined a **New test prefix** in the eCATT pane of the QuickTest Options dialog box, the Save QuickTest Test to eCATT dialog box displays the defined prefix. For more information, see "Setting Options for Working with eCATT" on page 385.



For more information on the Save QuickTest Test to eCATT dialog box, see "Understanding the Save QuickTest Test to eCATT Dialog Box".

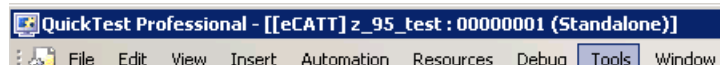
Note: The Save QuickTest Test to eCATT dialog box opens when QuickTest is connected to eCATT in standalone mode. To save a test directly in the file system while you are connected to eCATT, click the **File System** button to open the Save QuickTest Test dialog box.

- 3** In the **Test name** box, enter a valid name for the test. Use a descriptive name that will help you easily identify the test. Ensure that the test name begins with a prefix that matches your eCATT server naming conventions. For example, your eCATT server may require all test names to begin with the letter z.
- 4** In the **Test version** box, enter a version number. The version number can be any number that you choose. For example, each time you open and modify a test, you can increment the version number by 1, instead of overwriting the existing version of the test if you want to keep a record of all versions of a test. The test name and version number together form a unique ID for the test.
- 5** Click **OK** to save the test and close the dialog box. Note that the words **Saving**, and then **Uploading**, are displayed in the QuickTest status bar. When QuickTest completes the save process, the status bar displays the word **Ready**.

When the save process is complete, the QuickTest title bar displays the test information in the following format:

[eCATT] *TestName: VersionNumber (Mode)*

For example:

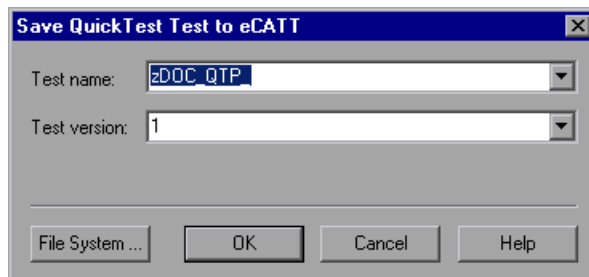


Understanding the Save QuickTest Test to eCATT Dialog Box

The Save QuickTest Test to eCATT dialog box enables you to save a QuickTest test in the eCATT database. This dialog box opens when you are connected to eCATT in standalone mode and you select to save a new test, or to save an existing test with a new name (Save As).

Note:

- ▶ This dialog box is similar to the Save External File from eCATT dialog box, and displays similar user interface elements.
 - ▶ When saving a test to eCATT in standalone mode, it is automatically saved to the local package (**\$TMP**) in eCATT.
-



The Save QuickTest Test to eCATT dialog box includes the following options:

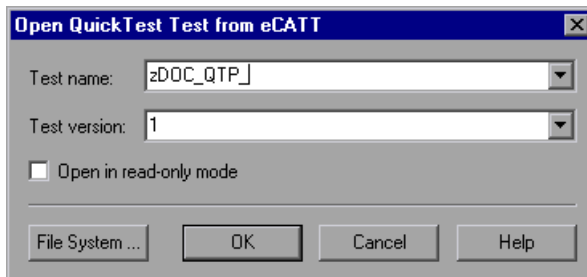
Option	Description
Test name	<p>The name for the test. Use a descriptive name that will help you easily identify the test. Ensure that the test name begins with a prefix that matches your eCATT server naming conventions. For example, your eCATT server may require all file names to begin with the letter Z.</p> <p>When the Save Test in eCATT dialog box opens, it displays the default test prefix in the Test name box. You can define or modify this prefix in the eCATT pane of the Options dialog box. For more information, see "Setting Options for Working with eCATT" on page 385.</p>
Test version	<p>The version number of the test. The version number can be any number that you choose. You can use the version number option as a kind of manual version control. For example, each time you open and modify a test, you can increment the version number by 1, instead of overwriting the existing version of the test if you want to keep a record of all versions of a test. The test name and version number together form a unique ID for the test.</p>
File System	<p>Opens the Save QuickTest Test dialog box, enabling you to save the currently open test anywhere in the file system.</p>

Opening Tests from an eCATT Database in Standalone Mode

When QuickTest is connected to an eCATT server in standalone mode, you can open and edit QuickTest tests that are saved in your eCATT database. You can also edit and save all of a test's resources (external files associated with the test), even if they are stored in eCATT.

To open a test from an eCATT database:

- 1 Connect to an eCATT server. For more information, see "Connecting QuickTest to eCATT" on page 367.
- 2 In QuickTest, click **Open** or select **File > Open** to open the test. The Open QuickTest Test from eCATT dialog box opens.



For more information on Open QuickTest Test from eCATT dialog box, see "Understanding the Open QuickTest Test from eCATT Dialog Box".

Note: The Open QuickTest Test from eCATT dialog box opens only when QuickTest is connected to an eCATT server. To open a test directly from the file system while you are connected to eCATT, click the **File System** button to open the Open Test dialog box.

- 3 In the **Test name** box, enter a valid test name or select one from the list of recently opened eCATT tests. Do not specify a folder path or other location.
- 4 In the **Test version** box, select a version number from the list.

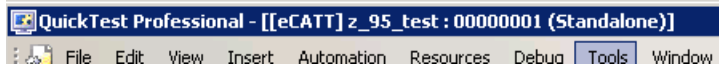
As QuickTest downloads and opens the test, the operations it performs are displayed in the status bar.

- 5 If you want to open the test in read-only mode, select the **Open in read-only mode** check box.

When the test opens, the QuickTest title bar displays the test information in the following format:

[eCATT] *TestName: VersionNumber (Mode)*

For example:

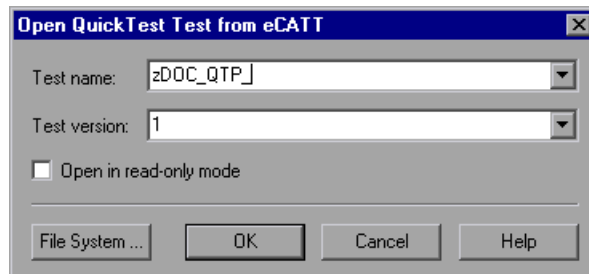


Understanding the Open QuickTest Test from eCATT Dialog Box



The Open QuickTest Test from eCATT dialog box enables you to open a QuickTest test from the eCATT database. This dialog box opens when you select **File > Open** or click the **Open** button while connected to eCATT in standalone mode.

Note: This dialog box is similar to the Open External File from eCATT dialog box, except that this dialog box contains the **Open in read-only mode** option.



The Open QuickTest Test from eCATT dialog box includes the following options:

Option	Description
Test name	The name of the test you want to open. When the Open Test from eCATT dialog box opens, it displays the most recently opened eCATT test in the Test name box.
Test version	The version number of the test you want to open.
Open in read-only mode	Opens the test in read-only mode. You can run the test and save the results, but you cannot modify the test or any external resources associated with the test.
File System	Opens the Open QuickTest Test dialog box, enabling you to open a test from anywhere in the file system.

Opening Tests from the Recent Tests List

When working in standalone mode, you can open eCATT tests from the recent tests list in the **File** menu. If you select a test located in an eCATT database, but QuickTest is currently not connected to eCATT or to the correct eCATT server for the test, the eCATT Connection dialog box opens.

Enter the **Password** and click **OK**.

The eCATT Connection dialog box also opens if you choose to open a test that was last edited on your computer using a different eCATT user name. You can either log on using the displayed **User** or you can click **Cancel** to prevent the opening of the selected test and remain logged in with your current user name.

Note: You must disconnect from eCATT before opening a Quality Center test from the recent tests list.

Uploading Files to eCATT

When you save a QuickTest test in eCATT, it is also recommended to store all associated resource files in eCATT so that any user who opens the test from eCATT will have access to all the test's resource files.

Like test names, all test resource files stored in eCATT must begin with a valid prefix. You can set the default prefix for files in the eCATT pane of the QuickTest Options dialog box. For more information, see "Setting Options for Working with eCATT" on page 385.

For files that you create outside QuickTest, such as Data Table files, library files, and environment variable files, you upload the files to eCATT using the **eCATT Upload File** option. You can also use this dialog box to upload existing shared object repository or recovery files from the file system to eCATT.

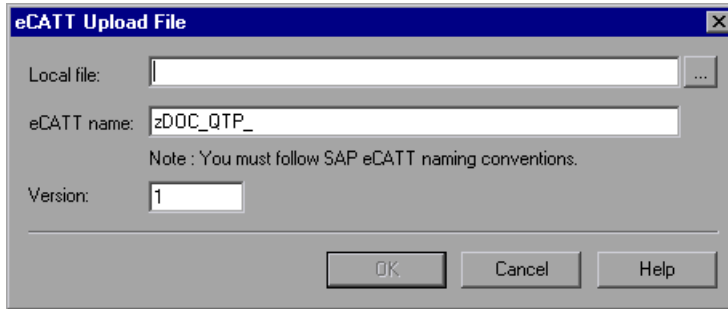
When you create a new shared object repository or recovery file, you can create the file as you normally would in QuickTest and then save the file directly to eCATT.

Note: When uploading a file to eCATT in standalone mode, it is automatically saved to the local package (**\$TMP**) in eCATT.

To upload external resource files from the file system to eCATT:

- 1** Create and save the file in the file system.
- 2** Connect to eCATT. For more information, see "Connecting QuickTest to eCATT" on page 367.

- 3 Select **Tools > eCATT Upload File** option. The eCATT Upload File dialog box opens.



For information on the eCATT Upload File dialog box, see "Understanding the eCATT Upload File Dialog Box".

- 4 Browse or enter the file path of the **Local file** you want to upload.
- 5 Specify the **eCATT name** and **Version** number you want to assign to the uploaded file.
- 6 Associate the uploaded file with your test in the appropriate QuickTest dialog box. For more information on associating Data Table files, library files, environment variable files, and shared object repositories with your test, see the *HP QuickTest Professional User Guide*.

To create a new shared object repository file and store it in eCATT:

- 1 Open a blank test.
- 2 Select **Resources > Object Repository Manager**.
- 3 Select **File > Save**. The Save External File to eCATT dialog box opens.
- 4 In the **File name** field, enter the name you want to use for the shared object repository according to the naming conventions of the eCATT server. For example, if your eCATT server requires all filenames to begin with z ,save the file in the following format: **z<filename>**. For example: zSOR_dwdm
- 5 In the **File version** field, enter the version number you want to use for the shared object repository.

- 6 If a warning message opens, click **Yes** to create the new object repository file in eCATT.

For more information on creating object repository files, see the *HP QuickTest Professional User Guide*.

To copy or export an object repository to eCATT:

- 1 Open the test whose object repository you want to copy or export.

Notes when exporting objects from a local object repository:

- You must select the action whose object repository you want to export.
- The object repository name must contain at least 14 characters.

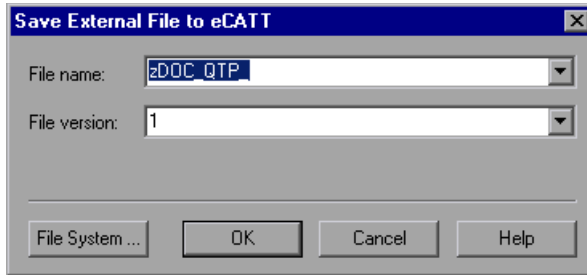
- 2 Select one of the following:

- **Resources > Object Repository Manager** to open the shared Object Repository Manager.
- **Resources > Object Repository** to open the local Object Repository for the selected action.

- 3 Do one of the following:

- In the shared **Object Repository Manager**, select **File > Save As** to save a copy of the object repository file with a new name in eCATT. The Save External File to eCATT dialog box opens. Proceed to step 4.
- In the **Object Repository dialog box**, select **File > Export Local Objects** to export the object repository to a shared object repository file in eCATT.

The Save External File to eCATT dialog box opens.



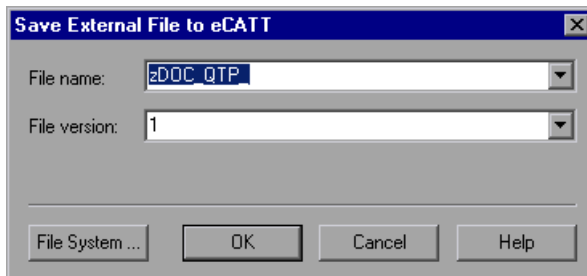
For information on the Save External File to eCATT dialog box, see "Understanding the Save External File to eCATT Dialog Box".

- 4 Enter the **File name** and **File version** for the shared object repository.
- 5 Click **OK** to save the file.

For more information on exporting and saving object repository files, see the *HP QuickTest Professional User Guide*.

To create a new recovery file in eCATT:

- 1 Select **Resources > Recovery Scenario Manager**. The Recovery Scenario Manager opens.
- 2 Click the **New Scenario** button. The Recovery Scenario Wizard opens. Follow the instructions in the wizard to create a new scenario. When you are finished, the scenario is displayed in the Recovery Scenario Manager. If you want to add more scenarios to the new scenario file, repeat step 2. When you are ready to save the scenario file, click **Save**. The Save External File to eCATT dialog box opens.



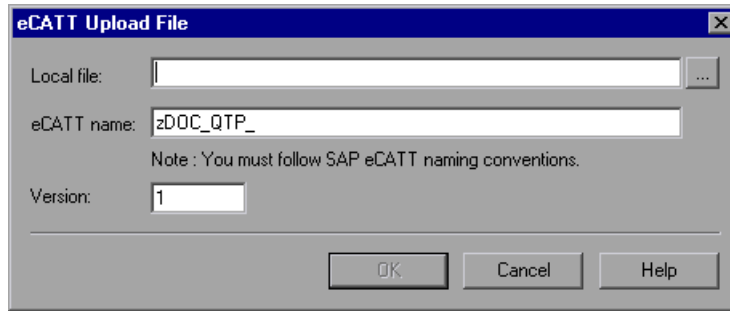
For information on the Save External File to eCATT dialog box, see "Understanding the Save External File to eCATT Dialog Box".

- 3** Enter the **File name** and **File version** for the recovery file.
- 4** Click **OK** to save the file.

For more information on creating and saving recovery files, see the *HP QuickTest Professional User Guide*.

Understanding the eCATT Upload File Dialog Box

You use the eCATT Upload File dialog box to store a test's external resource files in eCATT.

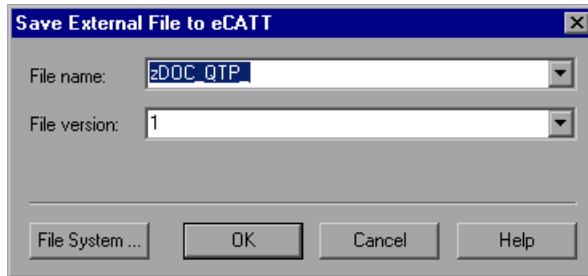


The eCATT Upload File dialog box includes the following options:

Option	Description
Local file	The complete path of the file you want to upload. You can enter file path or browse to the file.
eCATT name	<p>The name under which to store the file in eCATT. Ensure that the file name begins with a prefix that matches your eCATT server naming conventions. For example, you may have to prefix all file names with the letter Z.</p> <p>When the eCATT Upload File dialog box opens, it displays the default file prefix in the eCATT name box. You can define or modify this prefix in the eCATT pane of the QuickTest Options dialog box. For more information, see "Setting Options for Working with eCATT" on page 385.</p>
Version	The version number of the file. The version number can be any number that you choose. For example, each time you open and modify a file, you can increment the version number by 1, instead of overwriting the existing version of the file if you want to keep a record of all versions of a file. The file name and version number together form a unique ID for the file.

Understanding the Save External File to eCATT Dialog Box

You use the Save External File to eCATT dialog box to save a test's resource files directly to eCATT.

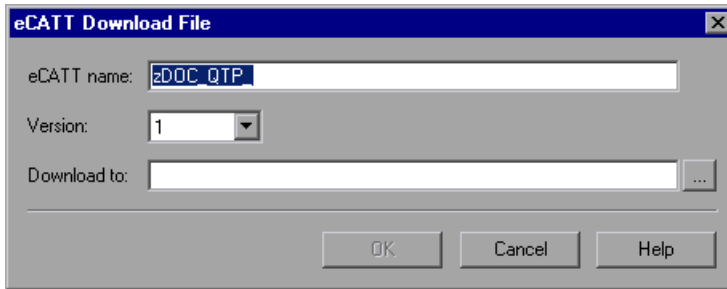


The Save External File to eCATT dialog box includes the following options:

Option	Description
File name	<p>The name under which to store the file in eCATT. Ensure that the file name begins with a prefix that matches your eCATT server naming conventions. For example, you may have to prefix all file names with the letter Z.</p> <p>When the eCATT Upload File dialog box opens, it displays the default file prefix in the eCATT name box. You can define or modify this prefix in the eCATT pane of the QuickTest Options dialog box. For more information, see "Setting Options for Working with eCATT" on page 385.</p>
File Version	<p>The version number of the file. The version number can be any number that you choose. For example, each time you open and modify a file, you can increment the version number by 1, instead of overwriting the existing version of the file if you want to keep a record of all versions of a file. The file name and version number together form a unique ID for the file.</p>

Downloading Files from eCATT

If you upload a file to eCATT and then associate that file with a test as a resource file, the resource file is automatically downloaded each time you open the test. You can also download files stored in eCATT and save them in the file system using the eCATT Download File dialog box (**Tools > eCATT Download File**).

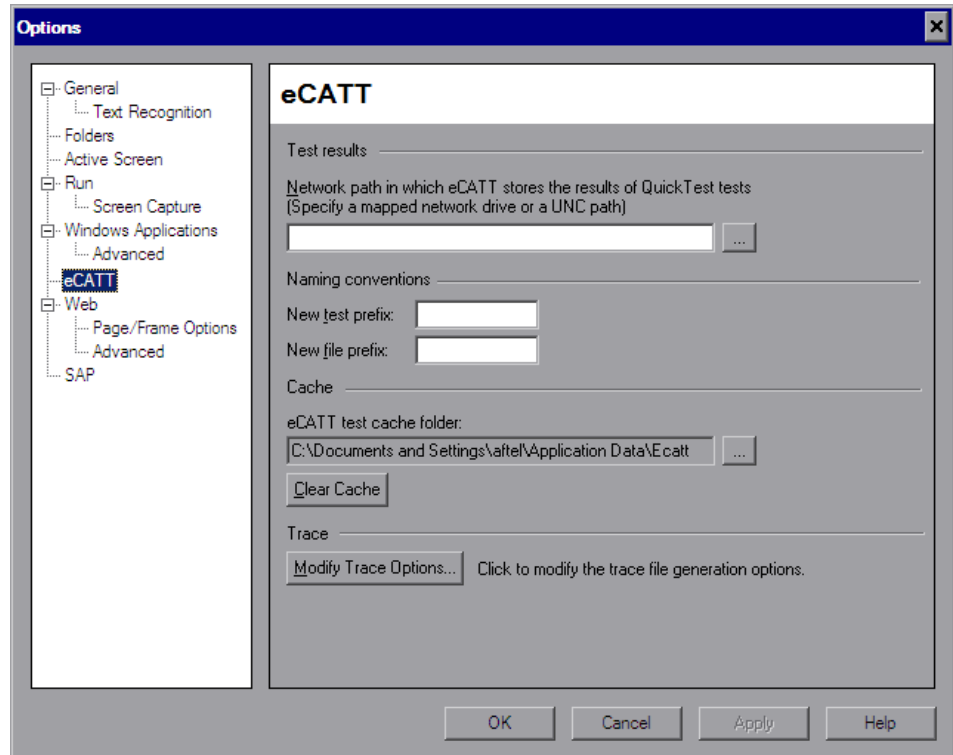


The eCATT Download File dialog box includes the following options:

Option	Description
eCATT name	The name of the file stored in eCATT. When the eCATT Download File dialog box opens, it displays the default file prefix in the eCATT name box. You can define or modify this prefix in the eCATT pane of the Options dialog box. For more information, see "Setting Options for Working with eCATT" on page 385.
Version	The version number of the file to download.
Download to	The complete path and file name of the location to which you want to download the file. You can enter or browse to the folder path.

Setting Options for Working with eCATT

The eCATT pane of the Options dialog box enables you to configure how QuickTest behaves when you are connected to eCATT.



Note: The eCATT pane is available only when the QuickTest Professional Add-in for SAP Solutions is installed and loaded.

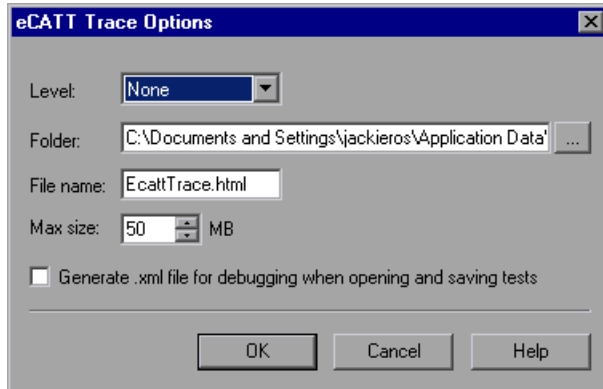
The eCATT pane contains the following options:

Option	Description
Run results	The location in which run results are stored when the test is run from eCATT. This folder must be a mapped network drive or a path in Universal Naming Convention (UNC) format.
New test prefix	The prefix that is displayed by default in the Save QuickTest Test in eCATT dialog box.
New file prefix	The prefix that is displayed by default when specifying a file to store in eCATT.
eCATT test cache folder	<p>The location in which a test from eCATT is temporarily stored when it is open for editing or running in QuickTest.</p> <p>Default location = C:\Documents and Settings\<username>\Application Data\eCATT</username></p> <p>Note: RunCache and EditCache folders are automatically created under the specified folder.</p>
Clear Cache	Deletes all files from the eCATT test cache folder.
Modify Trace Options	Opens the eCATT Trace Options dialog box, which enables you to configure if and how QuickTest generates the QuickTest-eCATT communication trace log (used for troubleshooting communication errors).

Configuring the eCATT Trace File

You can instruct QuickTest to generate a QuickTest-eCATT communication trace file each time eCATT runs a QuickTest test to troubleshoot communication errors.

You use the eCATT Trace Options dialog box to configure whether and how QuickTest generates the trace file.



The eCATT Trace Options dialog box contains the following options:

Option	Description
Level	<p>The level of detail to include in the trace file that is created when eCATT runs a QuickTest test.</p> <p>None. (default) No trace file is created.</p> <p>Low. The trace file lists any eCATT-QuickTest communication errors.</p> <p>Medium. The trace file includes eCATT-QuickTest communication errors and information on other major operations that result in eCATT-QuickTest communication.</p> <p>High. The trace file includes all available information related to eCATT-QuickTest communications.</p>

Option	Description
Folder	The folder path for storing the trace file. Required if a trace file level other than None is specified in the Level option. Default location = C:\Documents and Settings\<username>\Application Data\Ecatt\Trace</username>
File name	The file name for the trace file. Default = EcattTrace.html
Max size	The maximum file size you want to allow for the trace file.
Generate .xml file for debugging when opening and saving tests	Generates an .xml file each time you open or save a test that is stored in eCATT. The files are saved in Open and Save folders under the trace folder. Note: Selecting this option results in slower response times for editing and saving tests in QuickTest. In general, you should select this option only when instructed to do so to debug eCATT connectivity issues.

Passing Values Between eCATT Test Scripts and QuickTest Tests

You can pass values from an eCATT test script to a QuickTest test, or vice versa, using QuickTest test parameters.

Passing values to QuickTest involves:

- ▶ Defining QuickTest test parameters and using them in your QuickTest test. For more information, see the *HP QuickTest Professional User Guide*.
- ▶ Calling a QuickTest Test and Specifying Arguments from eCATT (see page 399)

Running a Test Stored in an eCATT Database in Standalone Mode

When you run the test from QuickTest (standalone mode), the run results are stored in the location you specify in the file system. You cannot access these results from eCATT.

Running Tests from QuickTest

When working with QuickTest in standalone mode, you run a test stored in an eCATT database just like any other QuickTest test.

To run a test stored in eCATT (in Standalone mode):

- 1** In QuickTest, click the **Run** button or select **Automation > Run**. The Run dialog box opens.
- 2** Accept the default results folder or browse to select another one.

Notes:

- The default results folder is created under the folder where the cache (local) copy of your test is stored. You set the location of your **eCATT test cache folder** in the eCATT pane of the QuickTest Options dialog box.
- When running tests in standalone mode, no eCATT log is created. For more information on the eCATT log for QuickTest run sessions, see "Viewing Results of a QuickTest Test Run in Integrated Mode" on page 406.

To run the test and overwrite the previous run session results, select the **Temporary run results folder (overwriting older temporary results)** option.

Note: QuickTest stores temporary run session results for all tests in <System Drive>:\%Temp%\TempResults. The path in the text box of the **Temporary run results folder (overwriting older temporary results)** option is read-only and cannot be changed.

- 3 Click **OK**. The Run dialog box closes and QuickTest begins running the test.

When the run session ends, the Run Results window opens (unless the **View results when run session ends** check box is cleared in the Run pane of the QuickTest Options dialog box). For more information on running QuickTest tests and analyzing run results, see the *HP QuickTest Professional User Guide*.

Working with eCATT in Integrated Mode

You can connect to QuickTest from eCATT. This is called *integrated mode*. When you work in integrated mode, only QuickTest features related to the eCATT test are available in QuickTest. When you run tests in integrated mode, your run session results are accessible in the eCATT log.

When you log on to an eCATT server that has been configured to integrate with QuickTest, you can view, edit, and run QuickTest tests that are stored in eCATT. You can also use the standard eCATT commands to copy, rename, and delete QuickTest tests, just as you would with any other file stored in eCATT.

When you open a QuickTest test from eCATT, QuickTest opens in integrated mode. In this mode, you can use all QuickTest features that are associated with the open test. You cannot open another test or save the open test with another name.

You can run a test in integrated mode in any of the following ways. You can use the **Run** option in QuickTest. You can use the **Execute Test Script** (F8) option for a selected QuickTest test in eCATT.

You can also execute an eCATT test script (or *blob*—Binary Large Object) that calls a QuickTest test. Creating eCATT scripts that call QuickTest tests is useful if you want to pass or retrieve values to or from a QuickTest test. For more information on configuring eCATT to work with QuickTest, see "Configuring eCATT to Work with QuickTest" on page 363.

You can also work with tests stored in eCATT from the QuickTest interface (standalone mode). For more information on working in standalone mode, see "Working with eCATT in Standalone Mode" on page 365. For more information on eCATT testing modes, see "Understanding eCATT Testing Modes" on page 361.

This section includes:

- ▶ "Performing Basic Test Management Operations from eCATT" on page 391
- ▶ "Transferring Data To and From QuickTest Tests Using Test Parameters" on page 399
- ▶ "Running a Test Stored in an eCATT Database in Integrated Mode" on page 403

Performing Basic Test Management Operations from eCATT

From eCATT, you can perform basic QuickTest test management operations, such as displaying or opening existing QuickTest tests that are stored in eCATT, making copies of tests, renaming tests, deleting tests, and creating new QuickTest tests.

For more information, see:

- ▶ Displaying or Editing a QuickTest Test from eCATT, below
- ▶ Copying QuickTest Tests from eCATT, on page 394
- ▶ Renaming and/or Changing Packages for QuickTest Tests from eCATT, on page 395
- ▶ Deleting QuickTest Tests from eCATT, on page 397
- ▶ Creating QuickTest Tests from eCATT, on page 397

Displaying or Editing a QuickTest Test from eCATT

You can select to display any existing QuickTest test that is stored in eCATT. When you open the test, QuickTest opens in integrated and read-only mode.

Alternatively, you can select to open the QuickTest test for editing. When the test opens in integrated mode, you can use many QuickTest options. For example, you can edit the test and run the test from the QuickTest interface. However, any external resource files (for example, shared object repository files or external Data Table files) open in read-only mode. Resource files that are saved with the test (for example, a local repository or the test's local Data Table file) are editable. To edit external resource files, open the test in standalone mode.

When you display or open a QuickTest test in integrated mode, you can work only with the open test. You cannot open another test or save the open test with another name.

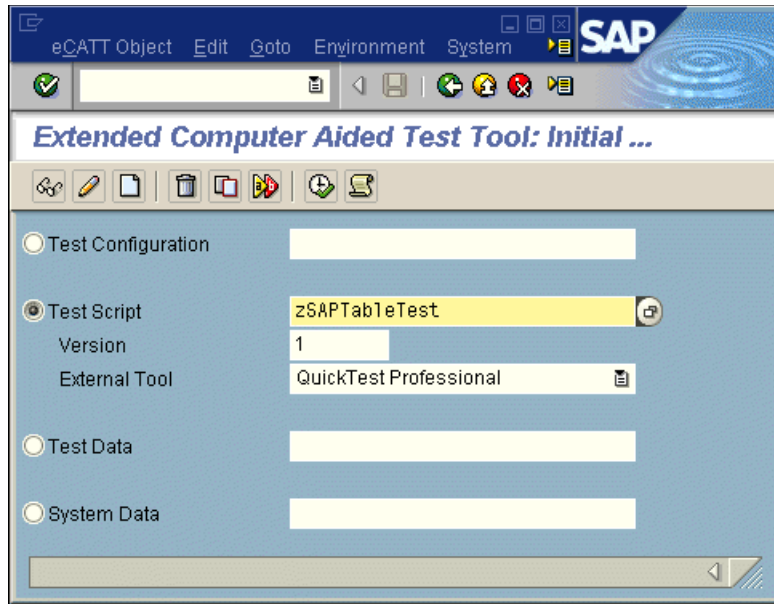
Note: If you select the QuickTest **File > Save As** menu command when working in integrated mode, QuickTest displays a warning message indicating that you can save a test with a new name in the file system, but doing so disconnects QuickTest from eCATT and switches QuickTest to standalone mode.

For more information on integrated and standalone modes, see "Understanding eCATT Testing Modes" on page 361.

To display or open a QuickTest Test from eCATT:

- 1 Log on to eCATT.
- 2 In the eCATT initial window, select **Test Script**.
- 3 In the **Test Script** box, enter the name of the test.
- 4 In the **Version** box, enter the test version number.

5 Select QuickTest Professional as the External Tool.



6 To display the test in read-only mode, click the **Display Object** button. The Display Test Script window opens.



To open the test for editing, click the **Change Object** button. The Change Test Script window opens.

7 Click the **Script (Call External Tool)** button (SHIFT + F12). If QuickTest is not already open, it opens. If the test has external resource files and/or if you chose to open the test in read-only (Display) mode, QuickTest reminds you that the resources and/or test will open in read-only mode.

8 Click **OK** on the message boxes. QuickTest displays the test.

9 If you opened the test for editing, you can use most QuickTest options. For more information, see the *HP QuickTest Professional User Guide*.

10 Close QuickTest to return to eCATT.

Copying QuickTest Tests from eCATT

You can create a copy of a QuickTest test that is stored in the eCATT database and store it with a different name.

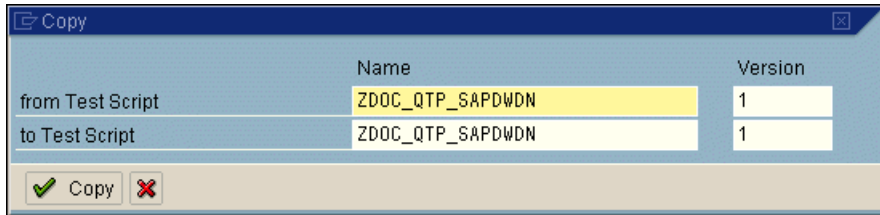
To copy a QuickTest Test:

1 Repeat steps 1 to 5 of "Displaying or Editing a QuickTest Test from eCATT" on page 392 to specify information about the test you want to copy.



2 Click the **Copy Object** button.

The Copy dialog box opens.



	Name	Version
from Test Script	ZDOC_QTP_SAPDWDN	1
to Test Script	ZDOC_QTP_SAPDWDN	1

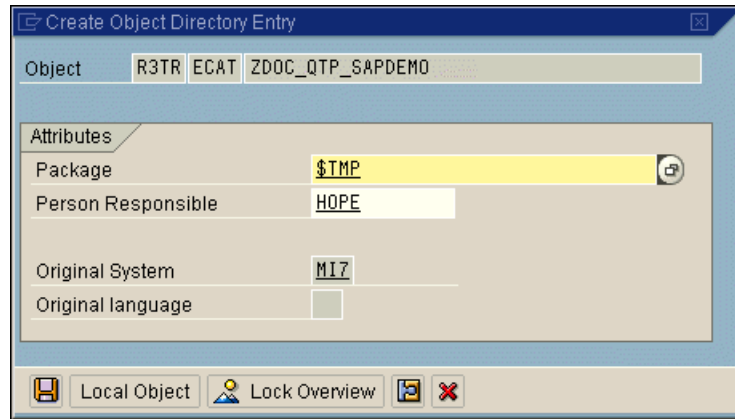
3 The **from Test Script** box displays the current QuickTest test **Name** and **Version** number.

4 In the **to Test Script** box, enter the **Name** and **Version** of the new copy of the test.

5 Click **Copy** to save the copy of the test.

► If you modified only the version number, proceed to step 8.

- ▶ If you entered a new test name, the Create Object Directory Entry dialog box opens.



- 6 In the **Package** box, specify the package in which you want to store the copy of the test. Modify other edit boxes as necessary.

Note: If the test has external resource files, they are stored separately in the **\$TMP** (local) package.



- 7 Click **Save** to save your settings and copy the test.
- 8 The eCATT initial window reopens and displays the name and version of the new copy.

Renaming and/or Changing Packages for QuickTest Tests from eCATT

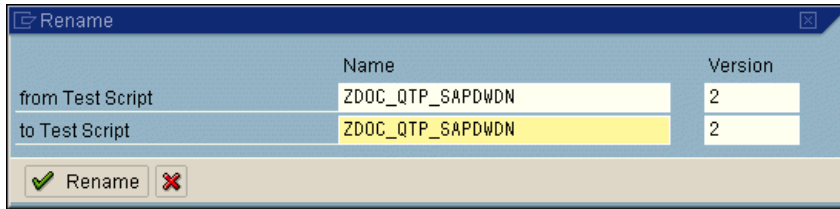
You can rename a QuickTest test that is stored in the eCATT database and store it with another name and/or store it in a different package.

To rename and/or change packages for a QuickTest Test:

- 1 Repeat steps 1 to 5 of "Displaying or Editing a QuickTest Test from eCATT" on page 392 to specify information about the test you want to rename or move.



2 Click the **Rename Object** button. The Rename dialog box opens.

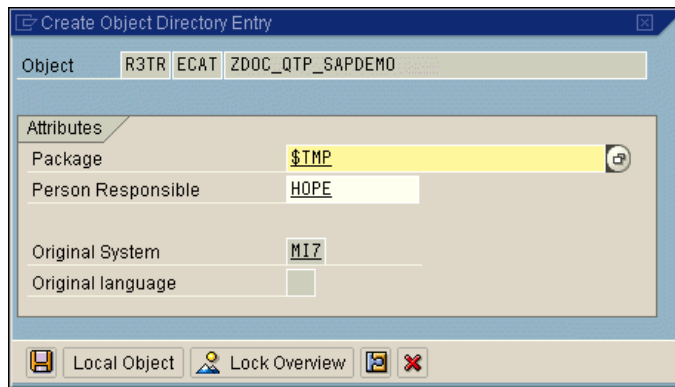


3 The **from Test Script** box displays the current QuickTest test **Name** and **Version** number.

4 In the **to Test Script** box, enter the new **Name** and/or **Version** for the test.

5 Click **Rename**.

- If you modified only the version number, proceed to step 8.
- If you entered a new test name, the Create Object Directory Entry dialog box opens.



6 In the **Package** box, specify the package in which you want to store the test. Modify other edit boxes as necessary.

Note: If the test has external resource files, they are stored separately in the **\$TMP** (local) package.



7 Click **Save** to save your settings and rename the test.

8 The eCATT initial window reopens and displays the name and version of the renamed test.

Deleting QuickTest Tests from eCATT

You can delete a QuickTest test from the eCATT database.

To delete a QuickTest test from the eCATT database:

1 Repeat steps 1 to 5 of "Displaying or Editing a QuickTest Test from eCATT" on page 392 to specify information about the test you want to delete.



2 Click the **Delete Object** button. A message box opens.

3 Click **Yes** to confirm that you want to delete the test. The test is deleted from the database.

Creating QuickTest Tests from eCATT

You can create a new QuickTest from eCATT.

To create a new QuickTest test:

1 Repeat steps 1 to 5 of "Displaying or Editing a QuickTest Test from eCATT" on page 392 to specify information about the test you want to create.



2 Click the **Create Object** button. The Create Test Script window opens. Ensure that the **General Data** tab is selected within the **Attributes** tab.

3 In the **Title** box, enter a title for your eCATT test script. The title is a short description of your test script.

4 Enter or select a **Component**.

Test Script ZDOC_QTP_SAPDEMO Version 1

Attributes

General Data Versioning Data Extras Restrictions

Header Data

Title Test that the SAP demo behaves as expected

External Tool QuickTest Professional

Package

Person Responsible HOPE Type B

Component CA Cross-Application Components

5 Click the **Script** button. QuickTest opens with a blank test.

6 Create the test in QuickTest. For information on creating tests in QuickTest, see the other chapters in this guide and the *HP QuickTest Professional User Guide*.



7 In QuickTest, click **Save**. The eCATT Create Object Directory Entry dialog box opens.

Create Object Directory Entry

Object R3TR ECAT ZDOC_QTP_SAPDEMO

Attributes

Package \$TMP

Person Responsible HOPE

Original System M17

Original language

Local Object Lock Overview

8 In the **Package** box, specify the package in which you want to store the test. Confirm that the other edit boxes contain correct values.

Note: If the test has external resource files, they are stored by default in the **\$TMP** (local) package. If you select another package for the test, you must manually move any external resource files to the same package.



- 9 Click **Save** to close the dialog box and save the test. QuickTest is restored in integrated mode and displays the saved test for additional editing.
- 10 When you are finished with the test, close QuickTest to return to eCATT.

Transferring Data To and From QuickTest Tests Using Test Parameters

You can pass values from an eCATT test script to a QuickTest test, or vice versa, using QuickTest test parameters.

Passing values to QuickTest involves:

- ▶ Defining QuickTest test parameters and using them in your QuickTest Test. For more information, see the *HP QuickTest Professional User Guide*.
- ▶ Calling a QuickTest Test and Specifying Arguments from eCATT (see below).

Calling a QuickTest Test and Specifying Arguments from eCATT

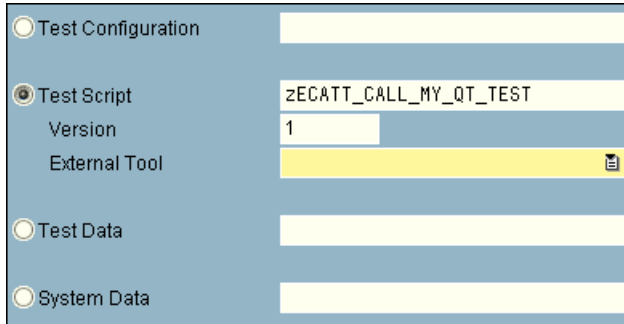
To send values to your QuickTest input arguments, you must run your test via a call from an eCATT test script.

After you have defined input and output arguments for your QuickTest test, you can insert a call to that test from an eCATT script and specify argument values for the input arguments.

To create an eCATT test script that calls and sends argument values to a QuickTest test:

- 1 Log on to eCATT.
- 2 In the eCATT initial window, select **Test Script**.

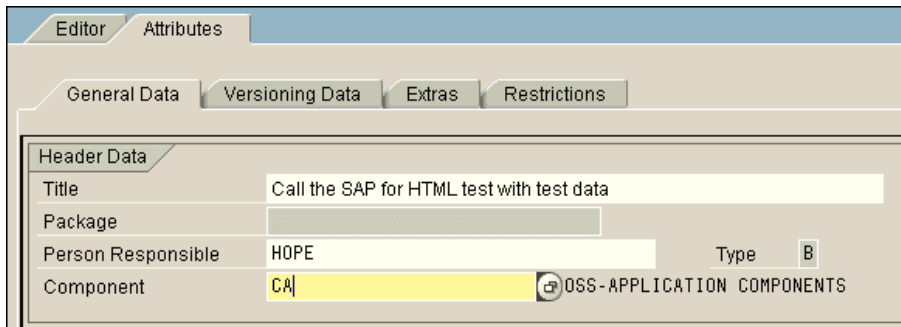
- 3 Enter a test script name and version number. Do not specify an **External Tool**.



<input type="radio"/> Test Configuration	
<input checked="" type="radio"/> Test Script	zECATT_CALL_MY_QT_TEST
Version	1
External Tool	
<input type="radio"/> Test Data	
<input type="radio"/> System Data	



- 4 Click the **Create Object** button. The Create Test Script window opens. Ensure that the **General Data** tab is selected within the **Attributes** tab.
- 5 In the **Title** box, enter a title for your eCATT test script. The title is a short description of your test script.
- 6 Enter or select a **Component**.



Editor		Attributes	
General Data		Versioning Data	
Extras		Restrictions	
Header Data			
Title	Call the SAP for HTML test with test data		
Package			
Person Responsible	HOPE	Type	B
Component	CA	OSS-APPLICATION COMPONENTS	

- 7 Click the **Editor** tab. The eCATT command editor is displayed.
- 8 If the **Command Interface** is not displayed, click the **Parameter<->Command Interface** toggle button to display the **Command Interface**.
- 9 Click the **Pattern** button in the Create Test Script toolbar. The Insert Statement window opens.



- 10** In the **Command** box, select **REFEXT** to call an external test. The window changes to display the options corresponding to an external call.

Command	REFEXT
Test Script (External Tool)	
Interface	
Version	1

- 11** In the **Test Script (External Tool)** box, enter the name of the QuickTest test you want to call.



- 12** Click the **Continue (Enter)** button. eCATT enters the default interface value in the **Interface** box. Modify this value if necessary.

- 13** If you want to call a test version other than version 1, enter the version number of the test you want to call in the **Version** box.

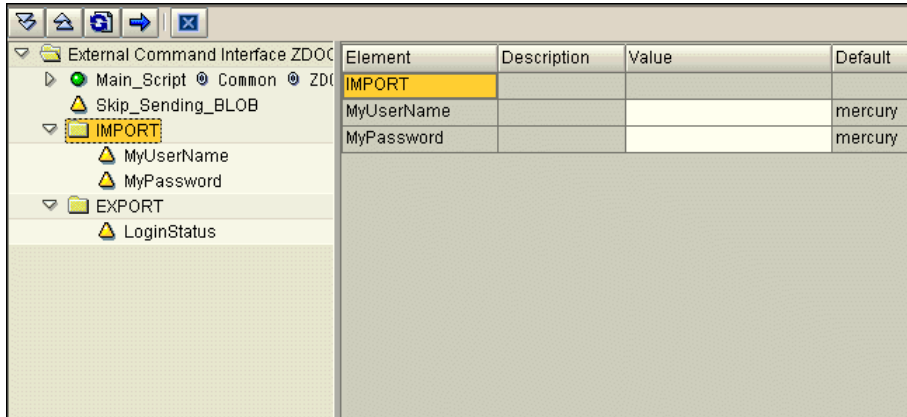


- 14** Click the **Continue (Enter)** button again to add the test to the **Command Interface**.

Command Interface	Description	Group	Ref. Ty...
ZDOC_QTP_SAPDWDN_1	External Script zDOC_QTP_SA...	REFEXT	E

- 15** Double-click the row header for the test in the **Command Interface**. The External Command Interface table is displayed.

- 16 Double-click the **Import** folder. If input test arguments have been defined for the test, they are displayed in the far-right grid.



Tip: You can also double-click the **Export** folder to view the output arguments defined for the test.

- 17 Set the value of each input argument in the **Value** column of the far-right grid.
- 18 Save the test script.

Tip: You can enter the name of an eCATT parameter from the eCATT script as the value of a QuickTest input parameter.

Running a Test Stored in an eCATT Database in Integrated Mode

When you run the test from eCATT (integrated mode), the run results are stored in the network drive specified for eCATT run results (as defined in the eCATT pane of the QuickTest Options dialog box). Although the run results are not stored in eCATT, you can access the results from the eCATT log.

For information on running tests in standalone mode, see "Running a Test Stored in an eCATT Database in Standalone Mode" on page 389.

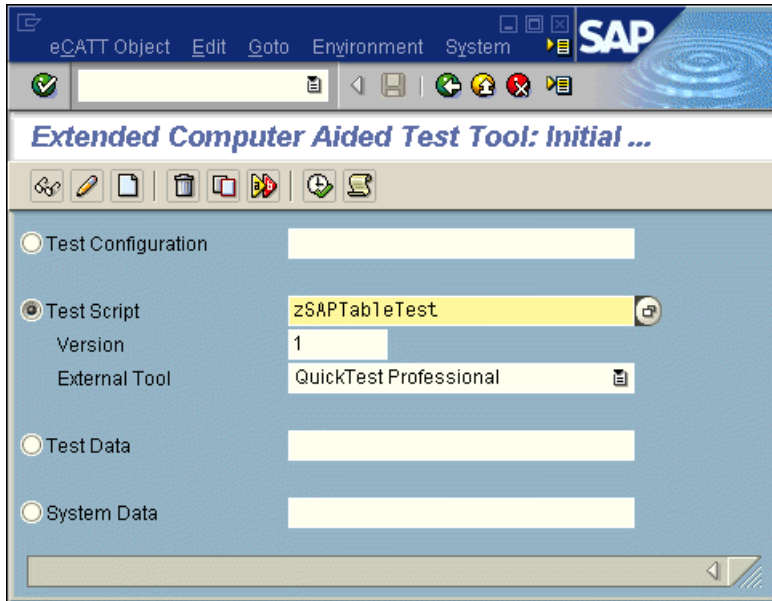
Running QuickTest Tests from eCATT Test Scripts (Integrated Mode)

To run a QuickTest test from eCATT, you can:

- ▶ Display the test in QuickTest and use the standard QuickTest **Run** option in QuickTest. For more information on displaying the test, see "Displaying or Editing a QuickTest Test from eCATT" on page 392. For information on running a test using the QuickTest Run option, see the *HP QuickTest Professional User Guide*.
- ▶ Create and run an eCATT test script that calls a QuickTest test. For information on creating eCATT test scripts that call QuickTest tests, see "Passing Values Between eCATT Test Scripts and QuickTest Tests" on page 388. For information on running eCATT test scripts, see your eCATT documentation.
- ▶ Use the **Execute Test Script** (F8) option for a selected QuickTest test in eCATT. For more information, see the procedure below.

To run a QuickTest test from eCATT using the Execute Test Script option:

- 1 Log on to eCATT.
- 2 In the eCATT initial window, enter the test name in the **Test Script** box and the version number in the **Version** box. Select **QuickTest Professional** as the **External Tool**.





- 3 Click the **Execute Test Script (F8)** button. The Start Options window opens.

Section	Property	Value
Error behavior	Error behavior	S No Termination, Continue with Next Script Command
	System Data	
System Data	Test System	
	Log Display	<input checked="" type="checkbox"/>
	Archiving	<input type="checkbox"/>
	Close RFC connection	<input type="checkbox"/>
TCD	Start Mode for Command TCD	N Process in Background, Synchronous Local
	Start Mode Overwrites Mode If TCD Is Run	<input checked="" type="checkbox"/>
SAPGUI	Proc Mode SAPGUI	N Optimized Performance
	Error Mode for SAPGUI	N Standard (Terminate on Any Error)
	Stop When	N Do Not Stop
	Close GUIs	N Close Generated Sessions After Script
External Tool	Mode for Ext. Tool	N Normal
	User Name	
	Password	*****

- 4 Select the options you want to use for the run session. For example, select the **Log Display** check box if you want the test log to display when the run session ends and select **A With Surface of External Tool** in the **Mode for Ext. Tool** box, if you want QuickTest to be displayed during the run session. For information on other options in this window, see your eCATT documentation.



- 5 Click the **Execute** button to start the run session. If you selected the **Log Display** check box in the Start Options window, then the eCATT log is displayed when the run session ends.

```

0000001291 Test Scpt zSAPTableTest Version 1 - SECATT [Without Interruption]
MI7 800 HOPE E 620 PIPELINE Windows NT ORACLE 02.11.2003 11:07:15
  zSAPTableTest Version 1 External Script zSAPTableTest (00000001)
    The test case has been active for [02:34 min] minutes
    
```

Viewing Results of a QuickTest Test Run in Integrated Mode

You can view the results of a QuickTest test that was run from eCATT in one of three ways:

- **In the QuickTest Run Results window.** Because all tests run from eCATT are stored in a network drive, you can open QuickTest (or the QuickTest Run Results Viewer application), and browse to the desired run results. You can view the folder in which the results are saved in the **UNCPathToLocalLog** line of the eCATT log.

```

0000001291 Test Scpt zSAPTableTest Version 1 - SECATT [Without Interruption]
MI7 800 HOPE E 620 PIPELINE Windows NT ORACLE 02.11.2003 11:07:15
  zSAPTableTest Version 1 External Script zSAPTableTest (00000001)
    EXT_SCRIPT 00000000 EXECUTED WITH QuickTest Professional XML-DATA-02
      External Command Interface (00000001)
        UNCPathToLocalLog = L:\QuickTest\Ver_6_5\Sample_Tests_4_Docs\zSAPTableTest\00000001\0000001291
    
```

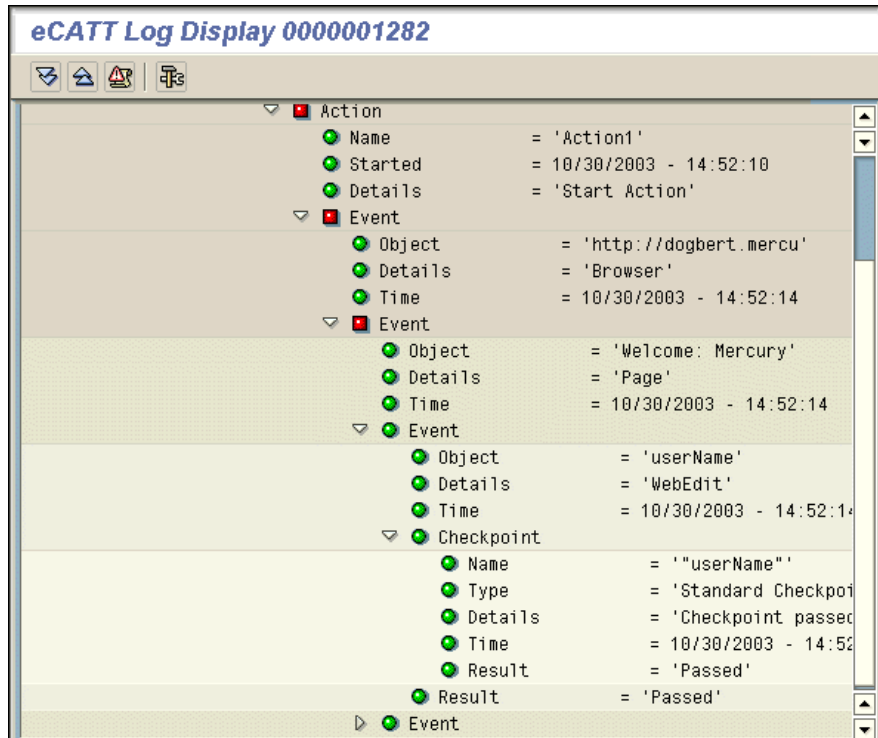
For more information on opening and analyzing run session results in QuickTest, see the *HP QuickTest Professional User Guide*.

- **Via the generated XML Report.** Each time you run a QuickTest test from eCATT, an **.xml** file is generated. This file contains all details of the run session. To view the file, click the line containing the text: **XML-DATA** in the eCATT log.

```

0000001291 Test Scpt zSAPTableTest Version 1 - SECATT [Without Interruption]
MI7 800 HOPE E 620 PIPELINE Windows NT ORACLE 02.11.2003 11:07:15
  zSAPTableTest Version 1 External Script zSAPTableTest (00000001)
    EXT_SCRIPT 00000000 EXECUTED WITH QuickTest Professional XML-DATA-02
      External Command Interface (00000001)
    
```

- **In the eCATT Log Display.** You can expand the log to view the results of the test or the results of a specific event, such as a checkpoint.



Troubleshooting and Limitations - SAP Windows

This section contains general troubleshooting and limitation information about the SAP Windows add-in, and includes the following sections:

- "Creating and Running Testing Documents" on page 408
- "Working with SAP Windows Controls" on page 409
- "Test Objects, Methods, and Properties" on page 410
- "Using the Active Screen" on page 411
- "SAP Scripting API" on page 411

Creating and Running Testing Documents

- Running a test on HTML elements embedded in an SAP Gui for Windows application may result in an "Object is disabled" error. This may happen if the HTML control is not ready for the test run.

Workaround: Add a **Sync** statement such as **SAPGuiSession.Sync** or a **Wait** statement to the script in order to run the test successfully.

- By default, the recording and running of steps on HTML elements embedded in an SAP Gui for Windows application is performed using the QuickTest Professional Web Add-in. In some cases, steps recorded using the Web Add-in are inserted into the script before SAP Add-in steps that use the SAP Scripting API.

Workaround: Use the option of recording HTML elements embedded in SAP Gui application using the SAP Scripting Interface. To do so, stop recording, select the **Record HTML elements using SAPGui Scripting interface** check box in the SAP pane of the Options dialog box (**Tool > Options > SAP** node). Then close and reopen the test and then begin recording again. For more information, see the *HP QuickTest Professional Add-ins Guide*.

- ▶ When you insert a copy of an action or when you insert a call to an action, but select **Use a local, editable copy** in the **Parameter data** section of the Insert Call to Action dialog box, QuickTest copies the action's data sheet to the test. However, if the called or copied action includes an **SAPGuiTable.Input**, **SAPGuiGrid.Input**, or **SAPGuiAPOGrid.Input** statement, the corresponding input data sheet is not copied to the Data Table with the action.

Workaround: Insert and run **Datatable.AddSheet** and **Datatable.ImportSheet** statements to import the sheet referenced by the action's **Input** method. Ensure that the name of the data sheet exactly matches the name specified in the corresponding **Input** statement.

- ▶ In the SAP Enterprise Portal environment, occasional synchronization problems may occur during the test run when alternating between SAP Web and SAP Windows environments.

Workaround: Add a **WaitProperty** or **Wait** statement between the Web steps and the Windows steps.

- ▶ The QuickTest Professional Add-in for SAP Solutions connects to your SAP Logon or SAP Logon Pad application for recording and running tests on SAP Gui for Windows sessions. If you use both SAP Logon and SAP Logon Pad processes on your desktop, QuickTest Professional will connect to the latest process that was launched.
- ▶ Use the SAP tab of the Record and Run Settings dialog box to instruct QuickTest to open your SAP Gui for Windows application. Do not use the Windows Applications tab of the dialog box for this purpose.

For more information, see the *HP QuickTest Professional Add-ins Guide*.

Working with SAP Windows Controls

- ▶ Separate toolbar controls (ones that are not part of a grid or other object) are supported by the SapGuiToolbar test object (**GuiComponentType** is 202), and the Object Spy recognizes them because they are separate objects.

Note that tree controls do not have associated toolbars. Toolbars displayed on top of tree controls are recognized as separate toolbars, and are therefore supported as described above.

- ▶ Toolbars inside grid controls are supported by the SapGuiToolbar test object (GuiComponentType is 204). However, the Object Spy does not recognize these toolbars because they are part of the grid. You cannot add these toolbars to the object repository using the **Add to repository** option from the Active Screen or the **Add Objects** option in the Object Repository dialog box. To add these toolbars to the object repository, record on them.
- ▶ Toolbars inside other controls (such as a toolbar within a text area control) are not supported.
- ▶ Microsoft Office controls within the SAP window are not supported.
- ▶ If you record the step of pressing an F4 key, and that key press results in setting new values for multiple fields, a step is recorded only for the field from which the F4 key was pressed, and therefore, only that field will be populated during the run.
- ▶ The SAP Editor control is not supported.
- ▶ QuickTest fails to run steps on SAP tree nodes that contain the ";" character.

Test Objects, Methods, and Properties

- ▶ When using the SAPGuiTable **Input** method, check the scrolling mode of the current table. If you parameterize a table with a Data Table sheet that contains more rows in the sheet than are displayed in the table's current view, QuickTest tries to scroll down the table while running the test, to insert more rows from the data sheet. QuickTest supports two ways of scrolling rows in tables—by pressing the ENTER key, or by pressing the PAGEDOWN key. By default, the Add-in for SAP Solutions tries PAGEDOWN if needed. You can configure the required mode using the second argument of the **Input** method.

For more information, see the *HP QuickTest Professional Add-ins Guide*.

- ▶ Right-click operations are not supported for the **SAPGuiTextArea** object.
- ▶ Drag-and-drop operations in the SAP Gui for Windows application are disabled when QuickTest is open.

Using the Active Screen

- ▶ Active Screen images are based on captured screen bitmaps. Therefore, objects that are not visible in the SAP GUI for Windows view are not part of the Active Screen image. You cannot add objects to the script from the Active Screen if they were not in the captured view.
- ▶ Drop-down menus are not captured in the Active Screen. Active Screen technology captures the data after the menu is closed and the menu item is selected.
- ▶ While recording, QuickTest Professional captures one Active Screen image for several steps. QuickTest Professional records steps only when the SAP GUI for Windows client sends information to the SAP back-end server. When this occurs, all steps that were performed between the previous communication and the current one are added to the script. The last screen that was sent to the server is captured by the Active Screen for all steps recorded during that communication.
- ▶ When recording on Web elements inside SAP GUI for Windows applications, HTML images are not captured.
- ▶ Adding objects to the object repository (using the **View/Add Object** option, or creating checkpoint or output value steps) from an Active Screen created from a step recorded on a Web element inside a SAP GUI for Windows application generates an incorrect object hierarchy in the object repository.

SAP Scripting API

- ▶ For security reasons, the SAP scripting API prevents the recording of passwords. When you record the operation of inserting a password in a password box, QuickTest records a **Set** statement using asterisks (****) as the method argument value.

Workaround: Record the password normally during the recording session. After the recording session, modify the password step to use the **SetSecure** method, and enter the encrypted password value or parameterize the value.

For more information, see the **SAP Windows** section of the *HP QuickTest Professional Object Model Reference* (**Help > QuickTest Professional Help > Object Model Reference > SAP Windows**).

- The QuickTest Professional Add-in for SAP Solutions does not automatically record standard Windows dialog boxes used by your SAP GUI for Windows application (such as the Open File and Save As dialog boxes). This is because the SAP scripting API does not support these dialog boxes. This may also occur when using SAP Gui for Windows with GuiXT.

Workaround: Change to Standard Windows Recording mode (select **Automation > Standard Windows Recording** or click the **Standard Windows Recording** toolbar button) to record on these objects. Alternatively, use low-level recording to record on these objects or use programmatic descriptions to run steps on these objects.

Note: If you switch to **Standard Windows Recording** mode *after* performing an operation on a standard Windows control, in some cases this may cause both QuickTest and the SAP application to become unresponsive. To avoid this, make sure that you switch to **Standard Windows Recording** mode *before* you perform the operation that opens the standard Windows control in your SAP application.

24

Enhancing Your SAP Windows Test

After you create your test, you can enhance it by adding checkpoints, retrieving output values, parameterizing values, and inserting SAP Windows objects, methods, and properties.

Note: All of the information in this chapter is relevant for tests and scripted components only.

This chapter includes:

- Considerations for Enhancing SAP Windows Tests on page 413
- Checking SAP Windows Objects and Outputting Values on page 414
- Outputting SAP Windows Property and Table Cell Values on page 420

Considerations for Enhancing SAP Windows Tests

- You must have the actual table or grid open to the appropriate level or view to insert a new table checkpoint while editing your test or component. This is true even if your Active Screen (tests only) contains a capture of the table or grid.
- In general, it is not necessary to open the table or grid in the application to edit an existing checkpoint. However, if you want to modify the row range for the checkpoint, the actual table or grid must be open to the appropriate level or view.

- ▶ When creating a checkpoint on simple table controls (tests only), QuickTest captures all rows and columns of the table (whether or not they are visible). You can choose to create your checkpoint on a specific range of rows in the Define Row Range dialog box.
- ▶ When creating a checkpoint on ActiveX grid controls (these generally have toolbars), QuickTest captures the data from all columns and all rows in the grid in the table checkpoint. If you do not need to check data from all rows in your grid, you can specify the rows you want to include in the checkpoint in the Define Row Range dialog box. You can also increase or decrease the number of rows included in the checkpoint at a later time.
- ▶ When inserting a table checkpoint (tests only), consider how other steps performed on the table may affect the checkpoint.

Example 1: If you have a step in your test that clicks the **Total** toolbar button on a grid control, that click refreshes all data in the table. The refresh could potentially cause a table checkpoint on a cell in the table to fail.

Example 2: If you click a toolbar button in a grid control that adds rows to your table before creating a table checkpoint, the extra rows are captured as part of the grid checkpoint (if you capture all rows). Therefore, confirm that the same rows are displayed during the run session.

- ▶ For more information on standard, table, text, and bitmap checkpoints, and on output values, see the *HP QuickTest Professional User Guide*.

Checking SAP Windows Objects and Outputting Values

After you create a test, you can use a variety of options to enhance it, including adding checkpoints and retrieving output values.

Adding a Table Checkpoint

When working with tests, you can check the contents and properties of simple table controls and ActiveX grid controls in your SAP GUI for Windows application by inserting table checkpoints. You can add a table checkpoint while recording or editing a test. Table checkpoints are not supported when working with business components.

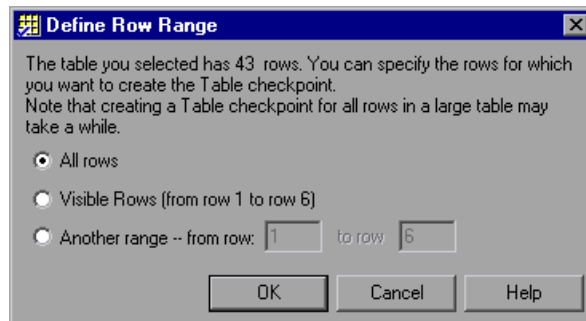
To add a table checkpoint while recording:

- 1** Select **Insert > Checkpoint > Standard Checkpoint** or click the **Insert Checkpoint or Output Value** toolbar button. The QuickTest window is minimized and the mouse pointer turns into a pointing hand.
- 2** Click the table or grid you want to check. If the location you select is associated with more than one object, the Object Selection dialog box opens. For more information, see the *HP QuickTest Professional User Guide*.



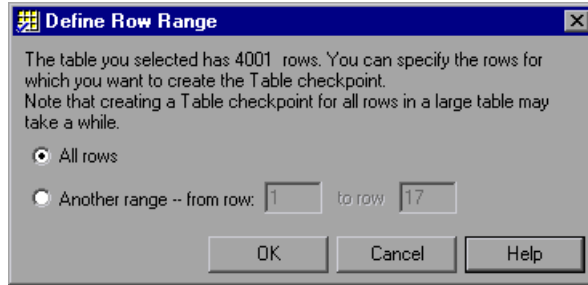
- 3** Select an **SAPGuiTable**, **SAPGuiGrid**, or **SAPGuiAPOGrid** object from the displayed object tree and click **OK**. The Define Row Range dialog box opens. The Define Row Range dialog box differs depending on whether a table object, a grid, or an APO grid object is selected.

- ▶ **Table controls.** The Define Row Range dialog box for table controls opens as follows:



Note: The total number of rows indicated in the first sentence of the dialog box for table controls is only an approximation. This is because only the data from visible rows is actually available for SAP Windows table controls.

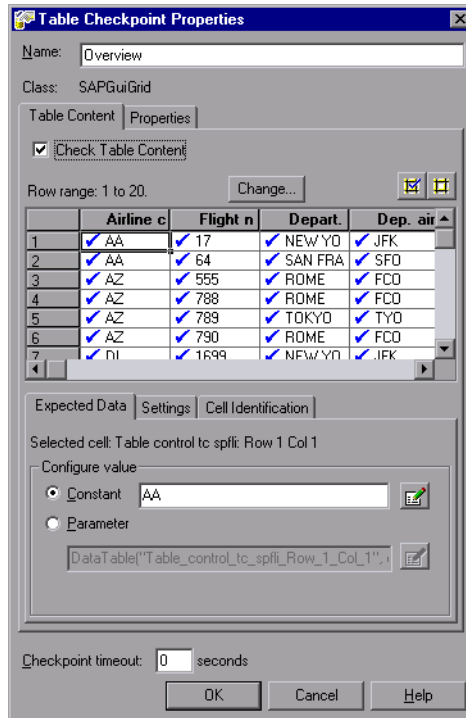
- ▶ **Grid and APOGrid controls.** The Define Row Range dialog box for grid controls and APOGrid controls opens as follows:



Note: The total number of rows indicated in the first sentence of the dialog box for grid controls is exact. The **Visible Rows** option is not available when checking grid controls.

- 4 Select the range of rows you want to include in your checkpoint. You can include all the rows in the table, only the visible rows (for SAP GUI for Windows table controls only), or another range that you specify.

5 Click **OK**. The Table Checkpoint Properties dialog box opens.



Note: If you selected **All rows** or you specified a large row range in the Define Row Range dialog box, it may take a few moments for the Table Checkpoint Properties dialog box to open.

Tips:

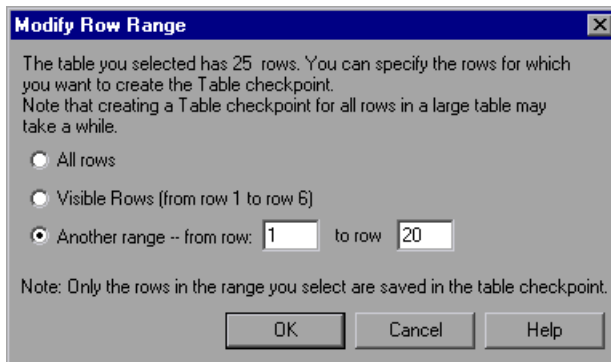
- You can rename the table in the **Name** field.
- The Table Checkpoint Properties dialog box has a **Change** button that enables you to modify the number of rows captured for the checkpoint. For more information, see "Modifying a Table Checkpoint" on page 418.

Modifying a Table Checkpoint

When working in tests, you can change the expected data, settings, and cell identification options for an existing table checkpoint. You can also change the rows that are included in the checkpoint.

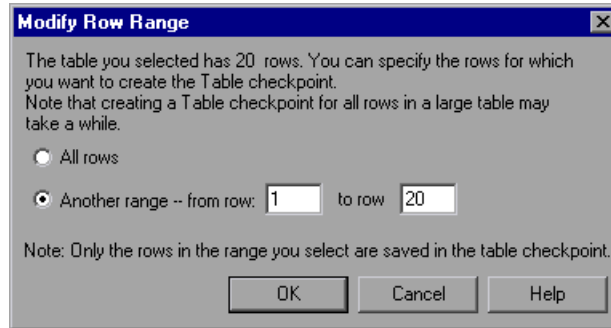
To modify the number of rows in an existing table checkpoint:

- 1 Open the SAP GUI for Windows application containing the table or grid you want to check and display the table or grid in the application.
- 2 In the Keyword View or Expert View, right-click the table checkpoint that you want to modify and select **Checkpoint Properties**. The Checkpoint Properties dialog box opens.
- 3 In the Table Content tab, click the **Change** button. The Modify Row Range dialog box opens. The Modify Row Range dialog box differs depending on whether a table object, grid, or APOGrid object is selected.
 - **Table controls.** The Modify Row Range dialog box opens as follows:



Note: The total number of rows indicated in the first sentence of the dialog box for table controls is only an approximation. This is because only the data from visible rows is actually available for SAP Windows table controls.

- **Grid and APOGrid controls.** The Modify Row Range dialog box for grid controls and APOGrid controls opens as follows:



Note: The total number of rows indicated in the first sentence of the dialog box for grid controls is exact. The **Visible Rows** option is not available when checking values for grid controls.

- 4 Select the range of rows you want to include in your checkpoint. You can include all the rows in the table or grid, only the visible rows (for SAP table controls only), or another range that you specify.
- 5 Click **OK**. The Modify Row Range dialog box closes, and the Table Checkpoint Properties dialog box displays the rows you specified in the Modify Row Range dialog box.
 - If your modified row range includes new rows, QuickTest captures the current values of the new rows from the open table in your SAP GUI for Windows application.

- ▶ If your modified row range includes some or all of the rows that were already included in the checkpoint, the expected values of those cells are not changed. This enables you to modify the row range without losing parameterization, regular expressions, or other changes you may have made to the expected cell values in your checkpoint. Therefore, you cannot use the Modify Row Range dialog box to update the expected values of an existing table checkpoint. To update the expected values of your checkpoint, use the **Update Run** option. For more information, see the section on updating checkpoints in the *HP QuickTest Professional User Guide*.
- ▶ 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.

Outputting SAP Windows Property and Table Cell Values

You can retrieve object property values (tests or components) or table cell values (tests only) during the run session. You can subsequently use these *output values* as input. This enables you to use data retrieved during a run session in other parts of your test.

Outputting Object Property Values of SAP Windows Objects

You can output the property values of objects in your application while recording or editing your test. You output the property values of SAP Windows objects just as you do for any other application.

For example, consider a purchase order transaction. You design a test in which you place a new order and then view the status of the new order. Each time you run the test, the application generates a unique purchase order ID for the new order. To view the status of a purchase order, you must enter the purchase order ID that was generated when the order was created. You cannot know the purchase order ID before you run the test.

To solve this problem, you create an output value for the unique number that is generated when you create a new order and store it in an OrderID column in the Data Table. Then you parameterize the step that specifies an order ID to view. You use the Order ID Data Table column (created in the output value step) as the parameter column.

When you run the test, QuickTest retrieves the unique order ID that was generated for the new order and inserts it in the run-time Data Table as an output value. When the test reaches the step for setting the value of the **Order ID** edit box in the View Order Status window, QuickTest inserts the unique order ID from the run-time Data Table into the **Order ID** edit box.

For more information on output values, see the *HP QuickTest Professional User Guide*.

Outputting Table Cell Values

When working with tests, you can output the contents of simple table controls and ActiveX grid controls in your SAP GUI for Windows application by inserting output values. You can insert an output value for a table or grid cell while recording or editing a test. Table output values are not supported for business components.

When inserting an output value for simple table controls, QuickTest captures all rows and columns of the table (whether or not they are visible), in the SAP GUI for Windows application. You can choose to create your output value on a larger range of rows in the Define Row Range dialog box.

When creating an output value for ActiveX grid controls (these generally have toolbars), QuickTest captures the data from all columns and all rows in the grid. If you do not need to output data from all rows in your test, you can specify the rows you want to include in the output value in the Define Row Range dialog box. You can also increase or decrease the number of rows included in the Table Output Value Properties dialog box at a later time.

Tip: You can also spool all of the available data from a table into an external file. For more information, see "Spooling Data from a Table" on page 430.

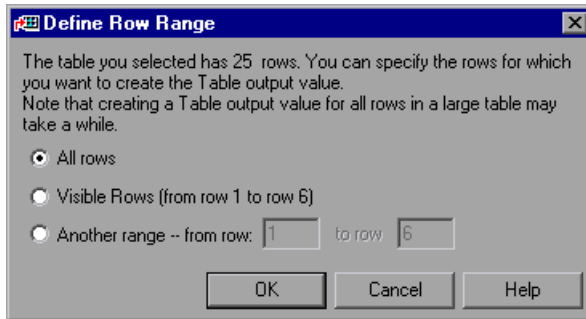
To create an output value for a table cell while recording:



- 1** In the Keyword View or Expert View, select **Insert > Output Value > Standard Output Value**. Alternatively, click the arrow next to the **Insert Checkpoint or Output Value** toolbar button and select **Standard Output Value**. The QuickTest window is minimized and the mouse pointer turns into a pointing hand.
- 2** Click the table or grid for which you want to insert an output value. If the location you select is associated with more than one object, the Object Selection dialog box opens. For more information, see the *HP QuickTest Professional User Guide*.
- 3** Select an **SAPGuiTable**, **SAPGui Grid**, or **SAPGuiAPOGrid** object from the displayed object tree and click **OK**. The Define Row Range dialog box opens.

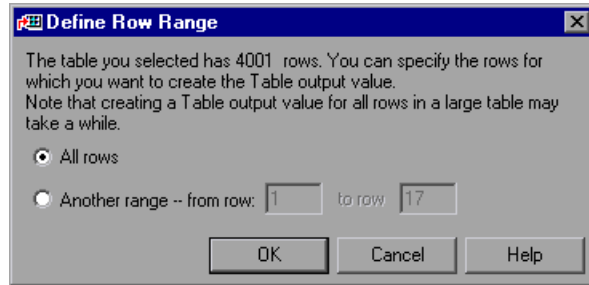


- **Table controls.** The Define Row Range dialog box opens as follows



Note: The total number of rows indicated in the first sentence of the dialog box for table controls is only an approximation. This is because only the data from visible rows is actually available for SAP Windows table controls.

- **Grid and APOGrid controls.** The Define Row Range dialog box for grid controls and APOGrid controls opens as follows:

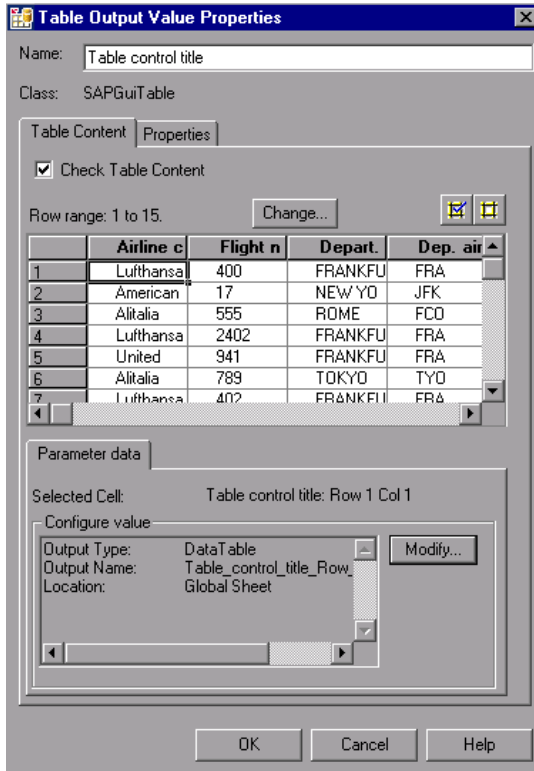


Note: The total number of rows indicated in the first sentence of the dialog box for grid controls is exact.

The **Visible Rows** option is not available when outputting values for grid controls.

- 4 Select the range of rows from which you want to output values. You can include all the rows in the table or grid, only the visible rows (for SAP table controls only), or another range that you specify.

- 5 Click **OK**. The Table Output Value Properties dialog box opens.




Note: If you selected **All Rows** or you specified a large row range in the Define Row Range dialog box, it may take a few moments for the Table Output Value Properties dialog box to open.

- 6 If required, you can modify the output value name in the **Name** box.
- 7 Specify your preferences for the cells you want to output. For more information on the Table Output Value Properties dialog box, see the *HP QuickTest Professional User Guide*.

Note: The Table Output Value Properties dialog box also has a **Change** button that enables you to modify the number of rows captured for the output value. For more information, see "Modifying a Table Output Value" on page 429.

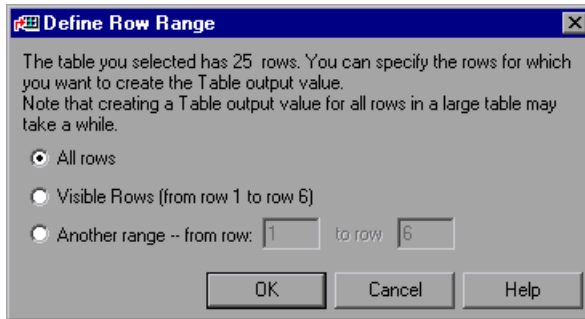
- 8** Click the **Modify** button if you want to change the output type and/or settings for the selected value. The Output Options dialog box opens and displays the current output type and settings for the value. For more information on the Output Options dialog box, see the *HP QuickTest Professional User Guide*.
- 9** Click **OK** to close the dialog box. QuickTest inserts an output value step in your test.

To create a table output value while editing your test:

- 1** Open the SAP GUI for Windows application containing the table or grid from which you want to output values and display the table in the application.
-  **2** Confirm that the **Active Screen** button is selected.
- 3** In the Keyword View or Expert View, click the step whose Active Screen contains the table or grid for which you want to specify an output value.
- 4** In the Active Screen, right-click the table or grid for which you want to create an output value and select **Insert Output Value**. If the location you select is associated with more than one object, the Object Selection - Output Value Properties dialog box opens.
- 5** Select the table or grid for which you want to create an output value and click **OK**. The Define Row Range dialog box opens.

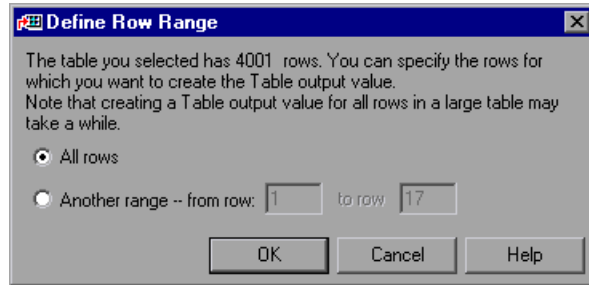
Tip: Instead of performing steps 2 to 4, you can right-click a table or grid object in the Keyword View or Expert View and select **Insert Output Value** to open the Table Output Value Properties dialog box for the selected table or grid.

► **Table controls.** The Define Row Range dialog box opens as follows:



Note: The total number of rows indicated in the first sentence of the dialog box for table controls is only an approximation. This is because only the data from visible rows is actually available for SAP Windows table controls.

- **Grid and APOGrid controls.** The Define Row Range dialog box for grid controls and APOGrid controls opens as follows:

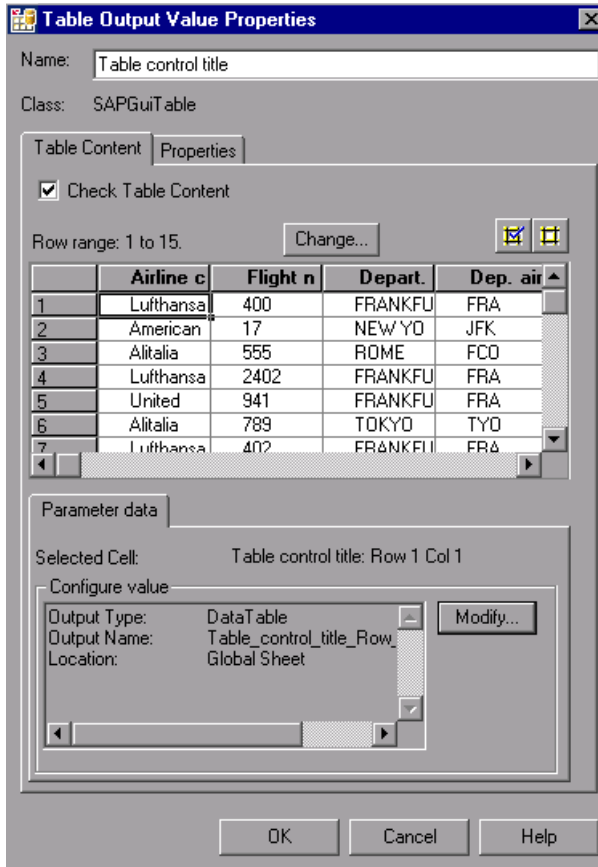


Note: The total number of rows indicated in the first sentence of the dialog box for grid controls is exact.

- 6 Select the range of rows from which you want to output values. You can include all the rows in the table or grid, only the visible rows (for SAP table controls only), or another range that you specify.

Note: The **Visible Rows** option is not available when outputting values for ActiveX grid controls.

- Click **OK**. The Table Output Value Properties dialog box opens.



Note: If you selected **All Rows** or you specified a large row range in the Define Row Range dialog box, it may take a few moments for the Table Output Value Properties dialog box to open.

- If required, you can modify the output value name in the **Name** box.

- 9 Specify your preferences for the cells you want to output. For more information on the Table Output Value Properties dialog box, see the *HP QuickTest Professional User Guide*.

Note: The Table Output Value Properties dialog box also has a **Change** button that enables you to modify the number of rows captured for the output value. For more information, see "Modifying a Table Output Value" on page 429.

- 10 Click the **Modify** button if you want to change the output type and/or settings for the selected value. The Output Options dialog box opens and displays the current output type and settings for the value. For more information on the Output Options dialog box, see the *HP QuickTest Professional User Guide*.
- 11 Click **OK** to close the dialog box. QuickTest inserts an output value step in your test.

Modifying a Table Output Value

You can modify existing table output value steps. You can change the cells to output or the options for selected cells. You can also change the rows that are available to output.

To modify the number of rows in an existing table output value step:

- 1 Open the SAP GUI for Windows application containing the table or grid for which you want to modify output values and display the table or grid in the application.
- 2 In the Keyword View or Expert View, right-click the output value step that you want to modify and select **Output Value Properties**. The Output Value Properties dialog box opens.
- 3 Click the **Change** button. The Modify Row Range dialog box opens.

- 4 Select the range of rows you want to make available to output. You can include all the rows in the table or grid, only the visible rows (for SAP table controls only), or another range that you specify.
- 5 Click **OK**. The Modify Row Range dialog box closes. The Table Output Value Properties dialog box displays the rows you specified in the Modify Row Range dialog box.
 - ▶ If your modified row range includes new rows, QuickTest captures the current values of the new rows from the open table in your SAP GUI for Windows application.
 - ▶ If your modified row range includes some or all of the rows that were already available for your output value, the Data Table settings of those cells are not changed.
 - ▶ If your modified row range excludes some or all of the rows that were previously available for your output value, those rows (and any output value settings you configured on cells in those rows) are deleted from the output value.

Spooling Data from a Table

If you want to spool all the data from an SAP GUI for Windows table into an external file, use the `GetCellData` method to loop through each cell in the table. You can then save the information to an external file.

The following example uses the `GetCellData` method to list the data of each cell in a table of 10 rows and 10 columns:

```
For i=1 to 10
  for j=1 to 10
    col="#" & j
    Dat=SAPGuiSession("Session").SAPGuiWindow("Create Standard").
      SAPGuiTable("SAPMV45ATCTRL_U_ERF_").GetCellData (i, col)
    'Enter lines of code that use the value of the returned Dat variable
  next
next
```

For more information on the `GetCellData` method, see the **SAP GUI for Windows** section of the *HP QuickTest Professional Object Model Reference*.

25

Adding SAP Windows Statements to Your Test or Component




In addition to recording the steps that you perform on your application, you can add statements to your test or component using the Keyword View. You can also add statements to your test using the Expert View or the Step Generator. This enables you to create a more comprehensive test or component for your SAP GUI for Windows application.

This chapter includes:

- ▶ Working with SAP Windows Test Objects on page 432
- ▶ Accessing Native Operations and Properties in Your SAP GUI for Windows Application on page 443

Working with SAP Windows Test Objects

The basic SAP Windows test object hierarchy consists of three levels: `SAPGuiSession.SAPGuiWindow.SAPGuiObject`, where `SAPGuiObject` represents an object in your SAP Windows application. For example, if you record a step on an edit box, QuickTest records a step with the following hierarchy:

 Session			
 SAP R/3	Resize	135,33	Resize the "SAP R/3" window to 135 by 33 characters.
 Fixed_cols	Set	"4"	Enter "4" in the "Fixed_cols" edit box.

```
SAPGuiSession("Session").SAPGuiWindow("SAP R/3").
  SAPGuiEdit("Fixed_cols").Set "4"
```

QuickTest has a set of SAP Windows test object classes that represent objects in your application on which you can record operations. There are also other objects that you can add to your test or component manually.

QuickTest also has an alternative recording mechanism that can be used to record objects in your application that are not represented by a specific test object class. For more information, see "SAP GUI for Windows Alternative Recording Mechanism" on page 442.

This section describes how QuickTest identifies the objects in an SAP GUI for Windows application and provides information on each of the following objects:

- ▶ SAPGuiSession Object (see page 433)
- ▶ Basic SAP GUI for Windows UI Controls
 - ▶ SAPGuiButton Object (see page 433)
 - ▶ SAPGuiCheckBox Object (see page 434)
 - ▶ SAPGuiComboBox Object (see page 435)
 - ▶ SAPGuiEdit Object (see page 434)
 - ▶ SAPGuiMenubar Object (see page 433)
 - ▶ SAPGuiOKCode Object (see page 434)
 - ▶ SAPGuiRadioButton Object (see page 434)

- SAPGuiTextArea Object (see page 435)
- SAPGuiWindow Object (see page 434)
- SAPGuiAPOGrid Object (see page 435)
- SAPGuiGrid Object (see page 436)
- SAPGuiCalendar Object (see page 436)
- SAPGuiElement Object (see page 437)
- SAPGuiLabel Object (see page 437)
- SAPGuiStatusBar Object (see page 437)
- SAPGuiTable Object (see page 438)
- SAPGuiTabStrip Object (see page 438)
- SAPGuiToolbar Object (see page 439)
- SAPGuiTree Object (see page 440)
- SAPGuiUtil Object (see page 442)

SAPGuiSession Object



The SAPGuiSession object represents the SAP GUI for Windows session on which an operation is performed. For tests and components, you can record on the SAPGuiSession object or add it manually using the Keyword View. For tests, you can also insert SAPGuiSession statements using the Step Generator or manually in the Expert View to create, reset, synchronize, or close a session.

Basic SAP GUI for Windows UI Controls

The example window on the next page contains the following objects:



➤ **SAPGuiMenubar.** Represents the menu bar at the top of the main SAP window.



➤ **SAPGuiButton.** Represents push-buttons in your application. The SAPGuiButton object is also recorded when you perform operations on buttons in the toolbar of the main SAP window.

Note that the SAPGuiToolbar object is used only for toolbar objects within your SAP GUI for Windows application. For more information, see "SAPGuiToolbar Object" on page 439.



► **SAPGuiOKCode.** Represents the edit box in which you enter commands to navigate to the desired transaction.



► **SAPGuiWindow.** Represents the main SAP window and the dialog boxes in your SAP GUI for Windows application.



► **SAPGuiEdit.** Represents fields in which you can enter a single line of text.



► **SAPGuiCheckbox.** Represents toggle check box objects in your application.



► **SAPGuiRadioButton.** Represents radio button objects in your application.

The screenshot shows the SAP User Interface Design: Boxes dialog box. The title bar includes menus: Screens, Elements, Dynamics, Menu design, System, Help. The main area contains a toolbar with icons for Cut, Copy, Paste, and Output on y. Below the toolbar are input fields for Customer (highlighted in yellow) and Company code (12). A 'Select items' section contains four checkboxes: Open items (checked), Cleared items (checked), Parked items (unchecked), and With special G/L transactions (checked). Below this are three buttons: Sel. condit ons..., With worklist, and Without worklist. At the bottom, there is a 'Settings' section with three dropdown menus: Line structure (Standard), Totals variant (Document type), and Sort variant (Standard). To the right of the settings is a 'List begins with' section with two radio buttons: Line items (selected) and Totals.

Annotations on the left side of the screenshot point to various UI elements:

- SAPGuiMenuBar: Points to the menu bar.
- SAPGuiButton: Points to the Cut icon in the toolbar.
- SAPGuiOKCode: Points to the Customer input field.
- SAPGuiWindow: Points to the main dialog area.
- SAPGuiEdit: Points to the Company code input field.
- SAPGuiCheckBox: Points to the 'Open items' checkbox.
- SAPGuiButton: Points to the 'Sel. condit ons...' button.
- SAPGuiRadioButton: Points to the 'Line items' radio button.

The example below contains the following objects:



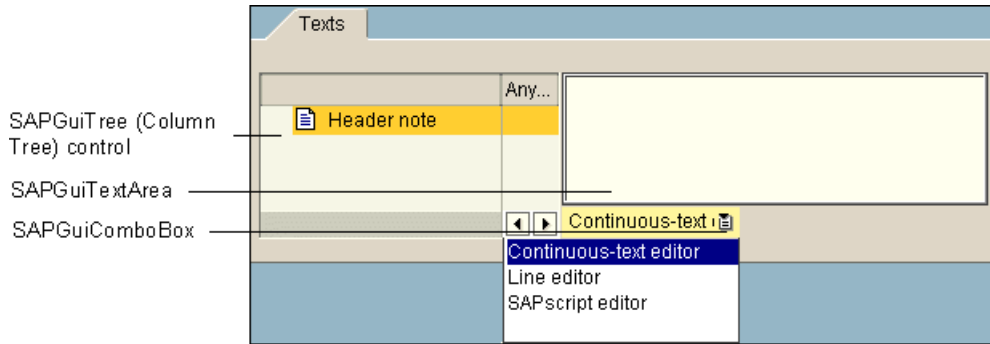
► **SAPGuiTree.** Represents simple or table tree controls. For more information, see "SAPGuiTree Object" on page 440.



► **SAPGuiTextArea.** Represents multi-line text areas.



► **SAPGuiComboBox.** Represents single- or multiple-selection combo boxes.



SAPGuiAPOGrid Object



The SAPGuiAPOGrid object represents APO Grid controls in your SAP GUI for Windows application. APO Grid controls are similar to Grid controls, with some additional functionality. The SAPGuiAPOGrid is available only in the SAP APO module.



SAPGuiGrid Object



The SAPGuiGrid object represents ActiveX grid controls in your SAP GUI for Windows application. Grid controls usually have toolbars that enable you to manipulate or perform operations on the values in the grid. The following image shows an SAPGuiGrid object after a row has been added to the grid as a result of highlighting the **Distance** column and clicking the **Total** button in the toolbar.

SAPGuiGrid for a grid control

ID	No.	Apt	Durati...	Departure	Arrival time	Σ	Distance	Dis.	Σ
LH	400	JFK	8:24	10:10:00	11:34:00		6.162 KM		0
LH	402	JFK	8:24	13:30:00	15:05:00		6.162 KM	X	0
LH	2402	SXF	1:05	10:30:00	11:35:00		555 KM		0
QF	6	SIN	12:25	23:45:00	18:10:00		10.000 KM		1
SQ	2	SFO	10:50	09:30:00	03:35:00		4.218 MI		0
SQ	158	JKT	2:25	11:30:00	13:55:00		560 MI		0
SQ	866	H...	4:30	07:10:00	11:40:00		1.625 MI		0
SQ	988	TYO	5:40	16:35:00	00:15:00		3.125 MI		1
UA	941	SFO	12:36	14:30:00	21:06:00		5.685 MI		0
UA	3504	FRA	13:35	15:00:00	10:30:00		5.685 MI		1
							33.434 KM		5
							56.331 MI		0

A row added to the grid as a result of toolbar operations.

SAPGuiCalendar Object



The SAPGuiCalendar object represents a calendar control in your SAP GUI for Windows application. The following image shows an SAPGuiCalendar object with a date selected.

SAPGuiCalendar for a calendar control

24.09.2003		WN	MO	TU	WE	TH	FR	SA	SU
2003/8	32	4	5	6	7	8	9	10	
	33	11	12	13	14	15	16	17	
	34	18	19	20	21	22	23	24	
	35	25	26	27	28	29	30	31	
	36	1	2	3	4	5	6	7	
2003/9	37	8	9	10	11	12	13	14	
	38	15	16	17	18	19	20	21	
	39	22	23	24	25	26	27	28	
	40	29	30	1	2	3	4	5	
2003/10	41	6	7	8	9	10	11	12	
	42	13	14	15	16	17	18	19	
	43	20	21	22	23	24	25	26	
	44	27	28	29	30	31	1	2	
2003/11	45	3	4	5	6	7	8	9	
	46	10	11	12	13	14	15	16	
	47	17	18	19	20	21	22	23	

Date selected in a calendar control

SAPGuiElement Object

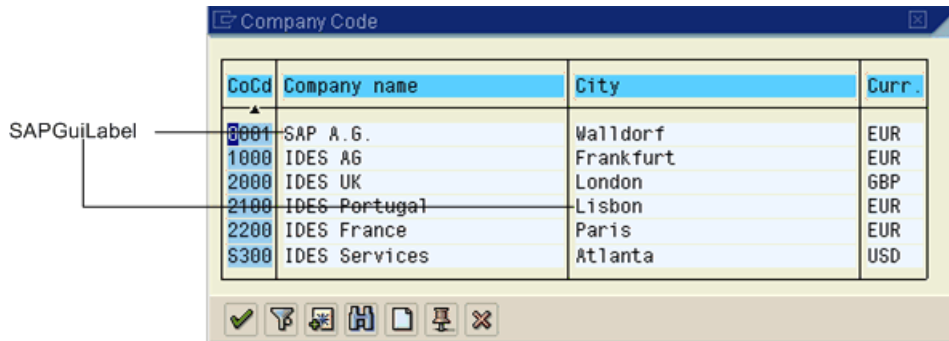


The SAPGuiElement object can represent any SAP Windows object. This object may be recorded if you insert a checkpoint or output value on an unrecognized SAP Windows object. You can also enter this object into your test manually to refer to any SAP Windows object that matches a specified programmatic description. For more information on programmatic descriptions, see the *HP QuickTest Professional User Guide*.

SAPGuiLabel Object



The following image shows a Possible Entries dialog box containing SAPGuiLabel (static text) objects. When you record on an SAPGuiLabel object, QuickTest always records the cursor position (**SetCaretPos** method) within the label object.



SAPGuiStatusBar Object



The following image shows an SAPGuiStatusBar object. You cannot record operations performed on a status bar, but you can check the entire text or any reserved parameter within the status bar's message text. When the **Record status bar messages** option is selected in the SAP pane of the Options dialog box, a step is automatically recorded each time a message is sent from the server. If this option is not selected, you can add **SAPGuiStatusBar** steps to the script only by inserting checkpoints or output values while recording.



SAPGuiTable Object



The SAPGuiTable object represents table controls in your SAP GUI for Windows application. SAPGuiTable objects are generally simple and may not have toolbar buttons (other than the **Table Settings** button).

SAPGuiTable

Table Settings button

Al...	Fli...	Depart.city	D...	Destination cit	D...	Flight ti...	Departur
AA	17	NEW YORK	JFK	SAN FRANCISCO	SFO		13:30:0
AA	64	SAN FRANCISCO	SFO	NEW YORK	JFK		09:00:0
AZ	555	ROME	FCO	FRANKFURT	FRA		19:00:0
AZ	788	ROME	FCO	TOKYO	TYO		12:00:0
AZ	789	TOKYO	TYO	ROME	FCO		11:45:0
AZ	790	ROME	FCO	OS	Flight time	KIX	10:35:0

SAPGuiTabStrip Object



The following image shows an SAPGuiTabStrip object. Note that using the tab navigation button to select a tab is recorded the same way as actually clicking the tab in a tab strip. Clicks on the tab rotation buttons are not recorded.

SAPGuiTabStrip

Assignments Control Period closing General data Investment ...

Company code 0003 Soesel & Partner GmbH &

Business area

Plant 0004 Helsinki

Object class

Profit center

Cost center respons.

WBS element

Request. cost center

Request. co. code

Sales order

Location/plant /

External order no.

Note: In rare cases, a running test or component may need to find an object in a tab strip that is not currently selected. In such an event, QuickTest finds the correct tab strip in which the object is located, selects that tab strip, and activates the object. Then the test or component continues to run. A comment that summarizes this automatic selection of the correct tab strip appears in the results.

SAPGuiToolbar Object



The SAPGuiToolbar object represents toolbar objects in your SAP GUI for Windows application.

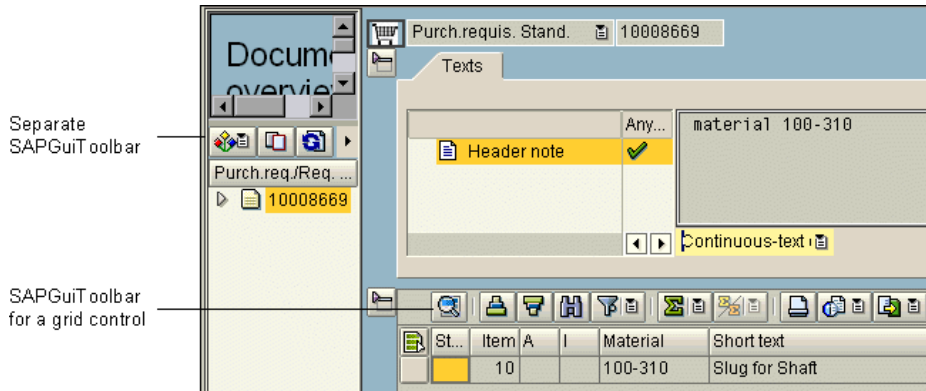
SAPGuiToolbars are recorded for:

- ▶ **GuiComponentType - 202.** Separate toolbar controls (toolbars that are not an integral part of another object).
- ▶ **GuiComponentType - 204.** Toolbars that are inside grid controls.

Note: The Object Spy does not locate SAPGuiToolbars that are part of grid controls. They are treated as part of the SAPGuiGrid object.

The toolbar buttons on the main SAP window toolbar are recorded as SAPGuiButton objects.

The example below shows a separate toolbar above a tree control that enables operations on the tree control, but is not associated with it. It also shows a toolbar inside a grid control.

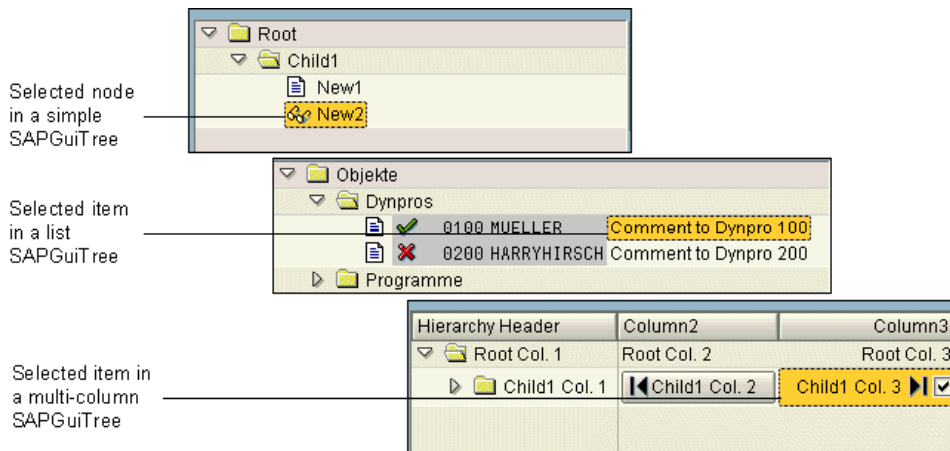


SAPGuiTree Object



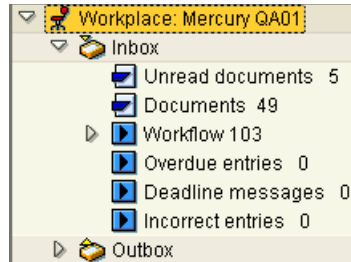
The SAPGuiTree object represents all tree controls in your application, including simple tree controls, list tree controls, and column tree controls. Based on information received from the SAP API, QuickTest recognizes the elements in simple trees as *nodes* and the elements in list and column trees as *items*. Therefore, you may record an **ActivateItem** step for one tree and an **ActivateNode** step for another tree, for example.

The examples below show the three different SAPGuiTree object types:



The names of some nodes or items may change dynamically, for example, the number of unread documents in an inbox may change as documents are added or read. Therefore, when you run a step from a test or component on an SAP tree control, the name of the node or item may have changed since the step was recorded.

The tree below contains examples of elements whose names may change over time.



For example, the **Workplace: Mercury QA01** node may have a different name at another time (when a different user logs in). Similarly, all the nodes and items in the **Inbox**, whose names include quantities (numbers of documents or table entries), are likely to change over time.

To enable QuickTest to identify these nodes and items when running a test or component, you can use regular expressions when you specify the path of the item or node.

The syntax when writing the value of the node or item name as a regular expression is:

RegExp:=<regular expression>

For example, a value for the **Unread documents** node of the above tree (that uses a regular expression) would look similar to this:

RegExp:=Workplace: .*;Inbox;RegExp:=Unread Documents \d*

The majority of the value is simple text. The regular expression parts of the value in this example are the two asterisks—one before **Inbox** and one at the end of the line—that represent the Workplace name and the number of unread documents respectively.

Each level of the tree can have its own RegExp:= value.

The following example illustrates a regular expression in the Keyword View and in the Expert View.

Item	Operation	Value
TableTreeControl	SelectNode	"RegExp:=Workplace: .*;Inbox;RegExp:=Unread Documents \d**"

```
SAPGuiSession("Session").SAPGuiWindow("Business Workplace of").
  SAPGuiTree("TableTreeControl").SelectNode "RegExp:=Workplace: .*;Inbox;
  RegExp:=Unread Documents \d**"
```

For more information on regular expressions, see the *HP QuickTest Professional User Guide*.

SAPGuiUtil Object



You can enter SAPGuiUtil statements that perform connection operations on your SAP GUI for Windows application during the run session. The SAPGuiUtil object is a reserved object and is not recorded.

SAP GUI for Windows Alternative Recording Mechanism

The QuickTest Professional Add-in for SAP Solutions provides an alternative recording mechanism for specific SAP GUI for Windows objects that do not have built-in test object support. This mechanism uses the **SAPGuiElement** test object and the **Object** method to record all the SAP GUI for Windows API events.

For example, using this recording mechanism, when you double-click an image object, the following statement is recorded in the Expert View:

```
SAPGuiSession("Session").SAPGuiWindow("SAP R/3").
SAPGuiElement("ImageCtrl").Object.doubleClickPictureArea "90","30"
```

The following SAP GUI for Windows objects are automatically recorded using this mechanism:

- ▶ Picture or Image controls
- ▶ BarChart controls

Accessing Native Operations and Properties in Your SAP GUI for Windows Application

You can use the **Object** property to access native (internal) operations and properties of objects in your SAP GUI for Windows application. The **Object** property is available for all SAP Windows objects

Tip: You can use the Object Spy to view the native operations and properties of an object in your application.

For example, you can use the grid's **setCurrentCell** method to set a row number to -1 (according to the SAP API).

```
SAPGuiSession("Session").SAPGuiWindow("Organization").
  SAPGuiGrid("GridViewCtrl").Object.setCurrentCell -1, "ADD_FIELD3"
```

The **Object** property is also useful for checking the value of properties that are not available using a standard checkpoint in your SAP GUI for Windows application.

For example, you can use the grid control's native **selectionMode** property to determine the types of selections that the grid supports. If the property returns the value **RowsAndColumns**, this indicates that the grid supports multiple row selection. In this case, you can use the **SelectRowsRange** test object method to select multiple rows in the grid.

```
GridMode = SAPGuiSession("Session").SAPGuiWindow("Vorg_nge").
  SAPGuiGrid("GridViewCtrl").Object.selectionMode
msgbox GridMode
If GridMode = "RowsAndColumns" then
  SAPGuiSession("Session").SAPGuiWindow("Vorg_nge").
  SAPGuiGrid("GridViewCtrl").SelectRowsRange 2,6
  SAPGuiSession("Session").SAPGuiWindow("Vorg_nge").
  SAPGuiButton("Enter").Click
End If
```

For more information on using the Object property, see the *HP QuickTest Professional User Guide*.

Part X

The Siebel Add-in

26

Using the Siebel Add-in

You can use the QuickTest Professional Siebel Add-in to test Siebel objects (controls).

For details on supported Siebel environments, see the **Siebel Add-in** section of the *HP QuickTest Professional Product Availability Matrix*, available from the Documentation Library Home page or the root folder of the QuickTest Professional DVD.

The Siebel Add-in provides test objects, methods, and properties that can be used when testing objects in Siebel applications. For more information, see the **Siebel** section of the *HP QuickTest Professional Object Model Reference*.

The following table summarizes basic information about the Siebel Add-in and how it relates to some commonly-used aspects of QuickTest.

General Information	
Add-in Type	This is a Web-based add-in. Much of its functionality is the same as other Web-based add-ins. See "Testing Web-Based Applications" on page 45.
Checkpoints and Output Values	<ul style="list-style-type: none"> ▶ See the sections describing checkpoints and output values in the <i>HP QuickTest Professional User Guide</i> (for tests), and the <i>HP QuickTest Professional for Business Process Testing User Guide</i> (for components). ▶ See "Considerations for Checking Siebel Objects" on page 473. ▶ See "Supported Checkpoints and Output Values Per Add-in" on page 769.
Other	<ul style="list-style-type: none"> ▶ You can use Siebel Test Express to automatically generate a new object repository, or update an existing object repository. <p>See "Generating an Object Repository Using Siebel Test Express" on page 479.</p>
Prerequisites	
Opening Your Application	You must open QuickTest and set Record and Run options before opening your Siebel application. Open the application only after you begin the recording session.
Add-in Dependencies	None
Other	<p>To test a Siebel 7.7.x or later application, you must:</p> <ul style="list-style-type: none"> ▶ Modify the Siebel Test Automation module configuration. ▶ Instruct your Siebel application to generate test automation information. <p>See "Setting Up Your Siebel 7.7.x or Later Environment" on page 452.</p>

Setting Preferences	
Options Dialog Box	Use the Web pane. (Tools > Options > Web node) See "Setting Web Testing Options" on page 52.
Record and Run Settings Dialog Box (tests only)	Use the Siebel tab. (Automation > Record and Run Settings) See "Setting Siebel Record and Run Options" on page 458.
Test Settings Dialog Box (tests only)	Use the Web pane. (File > Settings > Web node) See "Defining Web Settings for Your Application Area" on page 71.
Custom Active Screen Capture Settings Dialog Box (tests only)	Use the Web section. (Tools > Options > Active Screen node > Custom Level) See the section on the Custom Active Screen Capture Settings dialog box in the <i>HP QuickTest Professional User Guide</i> .
Application Area Settings Dialog Box (components only)	Use the Web pane. (File > Settings > Web node) <ul style="list-style-type: none"> ▶ See "Defining Web Settings for Your Application Area" on page 71. ▶ Use the Applications pane. (File > Settings > Applications node) See "Setting Siebel Application Options for Components" on page 464.

This chapter includes:

- ▶ Considerations for Working with the Siebel Add-in on page 450
- ▶ Setting Up Your Siebel 7.7.x or Later Environment on page 452

Considerations for Working with the Siebel Add-in

- ▶ QuickTest supports testing on both standard-interactivity and high-interactivity Siebel applications.
 - ▶ Standard-interactivity applications download data as it becomes necessary. This interface is designed for users accessing the application from outside the corporate network.
 - ▶ High-interactivity applications download the majority of the required data at one time, requiring fewer navigations. This interface is designed for heavy use, for example, by call centers.
- ▶ QuickTest learns objects in Siebel 7.7.x or later applications in a different way than in Siebel 7.0.x and 7.5.x applications. The Siebel Add-in has two different groups of test objects. The test object used to represent an object in your application depends on the Siebel version of your application and the implementation of the object. For more information, see "Understanding the Siebel Test Object Model" on page 456.
- ▶ When you load the Siebel Add-in, the Web event recording configurations designed for this add-in are loaded and are used whenever you record on a Siebel object. The Siebel Web event recording configurations do not affect the way QuickTest behaves when you record on other non-Siebel Web objects. For more information, see "Web Event Recording Configurations" on page 74.
- ▶ When you load the Siebel Add-in, the object identification settings are also automatically customized for Siebel. You do not need to make any changes to them. Therefore, the Siebel and Web options in the Object Identification dialog box are unavailable.
- ▶ Loading the ActiveX and Siebel add-ins together may cause problems when recording on some ActiveX methods.

Siebel 7.7.x or Later

As you record a test or component on your Siebel 7.7.x or later application, QuickTest records the operations you perform. QuickTest works directly with the Siebel Test Automation API (**SiebelAx_Test_Automation_18306.exe**) to record your operations. Therefore, although QuickTest records a step for each operation you perform, it adds the steps to your test or component only when API events are sent to QuickTest (when information is sent to the Siebel server).

When test automation is activated on a Siebel 7.7.x or later server and requested in the URL, the Siebel Web Engine (SWE) generates additional information about each object in the Siebel application when constructing the Web page. Each object has a specific set of properties, events, and methods that provide functionality for the Siebel application. The Siebel Test Automation API maps to these objects to enable you to manipulate your Siebel application from QuickTest when recording and running tests or components on the Siebel application.

Siebel 7.0.x/7.5.x

The Siebel Add-in can also identify Siebel objects by the **siebel attached text** property (the static text displayed with a Siebel object), rather than by the HTML name of the object. This enables you to maintain the test or component with dynamically created pages.

Setting Up Your Siebel 7.7.x or Later Environment

QuickTest Professional support for Siebel 7.7.x or later applications is based on the Siebel Test Automation API (**SiebelAx_Test_Automation_18306.exe**). Before you can create or run tests or components on your Siebel 7.7.x or later application, you must modify the Siebel Test Automation module configuration and instruct your Siebel application to generate test automation information.

Note: You do not need to make any configuration changes in Siebel 7.0.x and 7.5.x applications to create and run tests or components on these Siebel application versions.

Configuring the Siebel Test Automation Module

To test your Siebel 7.7.x or later application using the Siebel Add-in, you must confirm that your Siebel server has the Siebel Test Automation module installed and correctly configured to perform test automation. For detailed information, see the section that describes how to set up your functional testing environment in *Testing Siebel eBusiness Applications Version 7.7*, provided with your Siebel installation.

Generating Test Automation Information for Your Siebel Application

To create and run tests or components on your Siebel 7.7.x or later application, you must instruct the Siebel Web Engine (SWE) to generate test automation information for the Siebel application, using a SWE command. To do so, append the **SWECmd=AutoOn** token to the URL of your Siebel server. For example: `http://hostname/callcenter/start.swe?SWECmd=AutoOn`. If you do not append this token, the SWE does not generate test automation information.

If you select the **Open the following application when a record or run session begins** option in the Siebel tab of the Record and Run Settings dialog box, QuickTest automatically appends the Siebel Test Automation information to the URL (you do not need to specify it manually in the URL). For more information on the Record and Run Settings dialog box options, see "Setting Siebel Record and Run Options" on page 458.

Note: If a session timeout error occurs in your Siebel 7.7.x or later application, the Siebel Test Automation URL parameter values are not saved. After you log out and log in again, you must navigate to the correct URL that contains the required Siebel Test Automation parameter values (including password parameter values, if any—see below).

Generating Test Automation Information for Your Secured Siebel Application

If a password for generating test automation information is defined on your Siebel Server, you must also indicate that password in the URL (in addition to the **SWECmd=AutoOn** token described above). The URL token is in the format **AutoToken=password**. For example: `http://hostname/callcenter/start.swe?SWECmd=AutoOn&AutoToken=mYPass`. This enables QuickTest to run the Siebel Test Automation API **SiebelAx_Test_Automation_18306.exe** even in secure mode.

If a password is defined for the Siebel Server and you do not append this token to the URL, the SWE does not generate test automation information.

For information on whether your Siebel Server is secured for test automation, contact your Siebel system administrator.

If you select the **Open the following application when a record or run session begins** option in the Siebel tab of the Record and Run Settings dialog box, click the **Advanced** button, and specify the password in the **Siebel automation access code** box in the Advanced Siebel Record and Run Settings dialog box, QuickTest automatically appends the password information to the URL (you do not need to specify it manually in the URL). For more information on the Record and Run Settings dialog box options, see "Setting Siebel Record and Run Options" on page 458.

27

Creating and Running Tests and Components on Siebel Objects

The Siebel eBusiness platform is widely used in many organizations for their business process applications. QuickTest can create and run tests and components on these applications using special test objects and operations (methods and properties) that are customized for Siebel.

The customized Siebel test objects, methods, and properties make scripts simpler to read, maintain, enhance, and parameterize, enabling both advanced and novice users to create sophisticated tests and components on Siebel applications.

For a list of the specific Siebel versions supported by the QuickTest Professional Siebel Add-in, see the *HP QuickTest Professional Product Availability Matrix*, available from the Documentation Library Home page or the root folder of the QuickTest Professional DVD.

This chapter includes:

- Understanding the Siebel Test Object Model on page 456
- Setting Siebel Record and Run Options on page 458
- Setting Siebel Application Options for Components on page 464
- Using Environment Variables to Specify Record and Run or Applications Settings on page 464
- Recording Steps on Siebel Objects on page 466
- Information for Users of Earlier Versions of the QuickTest Professional Siebel Add-in on page 467
- Troubleshooting & Limitations - Siebel Add-in on page 468

Understanding the Siebel Test Object Model

The Siebel test object model is comprised of two different groups of test objects: test objects with the prefix `Sbl` and test objects with the prefix `Sieb`. If you are recording on a Siebel 7.0.x or 7.5.x application, QuickTest learns only `Sbl` test objects. If you are learning objects on a Siebel 7.7.x or later application, QuickTest may learn only `Sieb` test objects or a combination of `Sbl` and `Sieb` test objects, depending on the way in which your Siebel application was implemented.

For more information on each of the Siebel test objects, see the **Siebel** section of the *HP QuickTest Professional Object Model Reference*.

When you perform an operation on your Siebel application while recording a test or component, QuickTest:

- ▶ identifies the object on which you performed the operation and creates the appropriate test object in the test or component.
- ▶ reads the current value of the object's properties in your application and stores them in the object repository as the test object's property values.
- ▶ chooses a unique name for the test object, generally using the value of one of its prominent properties.
- ▶ records the operation (method) that you performed on the object and displays the operation as a step in the Keyword View and as a statement in the Expert View.

For example, suppose you select a check box for a specific account on a page of your Siebel application. This check box has the label **Competitor**.

QuickTest identifies the check box as a `SiebCheckbox` object. It creates a `SiebCheckbox` test object with the name **Competitor** and records the following properties and values as the description for the **Competitor** `SiebCheckbox`.

Type	Property	Value
ABC	repositoryname	Competitor
ABC	classname	SiebCheckbox

It also records that you performed a SetOn method to select the SiebCheckbox object.

QuickTest displays your step in the Keyword View like this:

Item	Operation	Documentation
▼ Action1		
▼ Siebel Call Center		
▼ Accounts		
▼ Account Details		
▼ Account		
<input checked="" type="checkbox"/> Competitor	SetOn	Select the "Competitor" check box.

QuickTest displays your step in the Expert View like this:

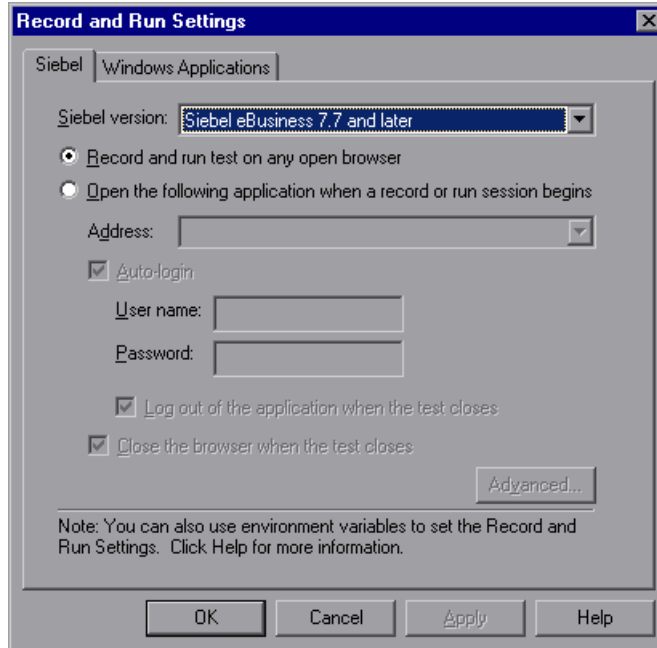
```
SiebApplication("Siebel Call Center").SiebScreen("Accounts").
    SiebView("Account Details").SiebApplet("Account").
        SiebCheckbox("Competitor").SetOn
```

When you run a test or component, QuickTest identifies each object in your application by its test object class and its *description*: the set of identification properties and values used to uniquely identify the object. In the above example, during the run session, QuickTest searches the object repository for the SiebCheckbox object named Competitor to look up its description. Based on the description it finds (**repositoryname** = Competitor and **classname** = SiebCheckbox), QuickTest searches the application for a SiebCheckbox object named Competitor. When it finds the object, QuickTest performs the SetOn method on the object to select the check box.

Setting Siebel Record and Run Options

You can control how QuickTest starts recording and running tests on Siebel objects by setting the record and run options.

The Record and Run Settings dialog box opens automatically each time you begin recording a new test (unless you open the dialog box and set your preferences manually before you begin recording).



Notes:

The Record and Run Settings dialog box applies only to tests. Record settings for components are specified in the Applications tab of the Application Area Settings dialog box. For more information, see "Setting Siebel Application Options for Components" on page 464.

If you have tests that were last modified using the Siebel Add-in, version 6.5, you need to convert your Record and Run Settings to use the Siebel tab instead of the Web tab. For more information, see "Information for Users of Earlier Versions of the QuickTest Professional Siebel Add-in" on page 467.

For more information on testing a Siebel 7.7.x or later application, see "Additional Information for Siebel 7.7.x or Later Applications" on page 463.

If you specify an application, you can supply a user name and password for QuickTest to use to log in to your Siebel application automatically, and you can select whether to log out of the application and/or close the browser when the test closes.

You can also use the **Advanced** option to change the default connection timeout setting and, if necessary, specify the password required to access Siebel Test Automation.

The Siebel tab includes the following options:

Option	Description
<p>Siebel version</p>	<p>Specifies the Siebel version for the applications on which you want to record your test. The version that you choose remains selected for all subsequent tests.</p> <p>You can use an environment variable to specify the Siebel version. For more information, see "Using Environment Variables to Specify Record and Run or Applications Settings" on page 464.</p>
<p>Record and run tests on any open browser</p>	<p>Instructs QuickTest to use any Internet Explorer browser to record and run the test.</p> <p>QuickTest can record and run only browsers that are opened after QuickTest is opened. If you are using Siebel 7.7.x or later, make sure you specify the required test automation parameters, as described in "Generating Test Automation Information for Your Siebel Application" on page 452.</p>
<p>Open the following application when a record or run session begins</p>	<p>Instructs QuickTest to open the specified application when record or run sessions begin.</p>
<p>Address (Enabled only when Open the following application when a record or run session begins is selected)</p>	<p>Instructs QuickTest to open Internet Explorer to the specified URL. It is recommended that you use the following format: <host>/<application name>/start.swe</p> <p>For example: siebapp/callcenter_enu/start.swe</p> <p>You can use an environment variable to specify the URL. For more information, see "Using Environment Variables to Specify Record and Run or Applications Settings" on page 464.</p>

Option	Description
<p>Auto-login (Enabled only when Open the following application when a record or run session begins is selected)</p>	<p>Instructs QuickTest to open the specified Siebel application using the specified login details.</p> <p>You can use an environment variable to specify the Auto-login setting. For more information, see "Using Environment Variables to Specify Record and Run or Applications Settings" on page 464.</p>
<p>User (Enabled only when Auto-login is selected)</p>	<p>The user name used to log in to the specified application.</p> <p>You can use an environment variable to specify the user name. For more information, see "Using Environment Variables to Specify Record and Run or Applications Settings" on page 464.</p>
<p>Password (Enabled only when Auto-login is selected)</p>	<p>The password for the specified user name.</p> <p>You can use an environment variable to specify the password. For more information, see "Using Environment Variables to Specify Record and Run or Applications Settings" on page 464.</p>
<p>Log out of the application when the test closes (Enabled only when Auto-login is selected)</p>	<p>Instructs QuickTest to log out of the specified application automatically when the test closes. Any other Siebel sessions that were opened before, during, or after the test run are not affected.</p> <p>You can use an environment variable to specify the Log out setting. For more information, see "Using Environment Variables to Specify Record and Run or Applications Settings" on page 464.</p>

Option	Description
<p>Close the browser when the test closes (Enabled only when Open the following application when a record or run session begins is selected)</p>	<p>Instructs QuickTest to close the opened browser when the test closes. Any other browsers that were opened before, during, or after the test run are not affected.</p>
<p>Advanced (Enabled only when Siebel version 7.7 and later and Open the following application when a record or run session begins are selected)</p>	<p>Opens the Advanced Siebel Record and Run Settings dialog box, where you can specify the following options:</p> <ul style="list-style-type: none"> ▶ Siebel automation request timeout. The timeout period (in seconds) for each attempt to connect to Siebel Test Automation when running the test. The default timeout is 120 seconds. ▶ Siebel automation access code. The predefined security code required to enable access to Siebel Test Automation, if specified by your organization's access security policy.

For more information on the Record and Run Settings dialog box, see "Using the Record and Run Settings Dialog Box" on page 38.

Additional Information for Siebel 7.7.x or Later Applications

To test a Siebel 7.7.x or later application, you must open the Siebel application with Siebel Test Automation loaded, by specifying additional URL parameter values. For more information, see "Setting Up Your Siebel 7.7.x or Later Environment" on page 452.

If you select the **Open the following application when a record or run session begins** option in the Siebel tab of the Record and Run Settings dialog box, QuickTest automatically appends the Siebel Test Automation information to the URL (you do not need to specify it manually in the URL). If you select to record and run on any open browser, you must specify the required parameter values as part of the application URL when you open the application.

If you select the **Open the following application when a record or run session begins** option in the Siebel tab of the Record and Run Settings dialog box, and specify the password in the **Siebel automation access code** box in the Advanced Siebel Record and Run Settings dialog box, QuickTest automatically appends the password information to the URL (you do not need to specify it manually in the URL). If you select to record and run on any open browser, or do not specify the password in the Advanced Siebel Record and Run Settings dialog box, you must specify the required password values as part of the application URL when you open the application.

If a session timeout error occurs in your Siebel 7.7.x or later application, the Siebel Test Automation URL parameter values are not saved. After you log out and log in again, you must navigate to the correct URL that contains the required Siebel Test Automation parameter values.

Setting Siebel Application Options for Components

The Applications tab of the Application Area Settings dialog box enables you to specify the Siebel version on which you are recording the component. You can record steps only on applications that use the specified Siebel version. You can also view these settings in read-only format in the Applications tab of the Business Component Settings dialog box.

In the **Siebel version** box, specify the Siebel version for the applications on which you want to record your component. The version that you choose remains selected for all subsequent components.

Note: You can use an environment variable to specify the Siebel version. For more information, see "Using Environment Variables to Specify Record and Run or Applications Settings", below.

Using Environment Variables to Specify Record and Run or Applications Settings

You can use environment variables to specify the options you want to use for recording and running your test or recording your component.

If you define any of these environment variables, these variables override the corresponding values in the Siebel tab of the Record and Run Settings dialog box for tests or the Applications tab of the Application Area dialog box for application areas and components.

- ▶ For more information, see "Setting Siebel Record and Run Options" on page 458 or "Setting Siebel Application Options for Components" on page 464.
- ▶ For more information on defining and working with environment variables, see "Using Environment Variables to Specify the Record and Run Details for Your Test" on page 40.

You can use the variable names listed in the table below to define Siebel application details:

Option	Variable Name	Description
Siebel version	APPLICATION_ENV	The Siebel version for the applications on which you want to record your test or component. Possible values: 77 7075 This option is available for tests and components.
Address	URL_ENV	The URL of the application you want to open. This option is available only for tests.
Auto-login	AUTO_LOGIN_ENV	Indicates whether to automatically log in to the application to open. This option is available only for tests. Possible values: True False
User	USER_NAME_ENV	The user name used to log in to the application to open. This option is available only for tests.
Password	PASSWORD_ENV	The encrypted password for the application to open. This option is available only for tests.
Log out of the application when the test closes	LOGOUT_ENV	Indicates whether to automatically log out of the application when the test closes. This option is available only for tests. Possible values: True False

Recording Steps on Siebel Objects

When you record an operation on a Siebel object, QuickTest inserts a step with the relevant Siebel object in the Keyword View and adds the corresponding statement in the Expert View.

For example, if you select an item from a list, the Keyword View may be displayed as follows:

Item	Operation	Value	Documentation
<ul style="list-style-type: none"> ▼ Action1 <ul style="list-style-type: none"> ▼ Siebel Call Center <ul style="list-style-type: none"> ▼ Accounts <ul style="list-style-type: none"> ▼ Account Details <ul style="list-style-type: none"> ▼ Account <ul style="list-style-type: none"> Account Type 	Select	"Consultant"	Select the "Consultant" item in the "Account Type" pick list.

QuickTest records this step in the Expert View as:

```
SiebApplication("Siebel Call Center").SiebScreen("Accounts").
  SiebView("Account Details").SiebApplet("Account").
  SiebPicklist("Account Type").Select "Consultant"
```

Tip: It is recommended to log out of your Siebel application at the end of the recording session before closing the browser.

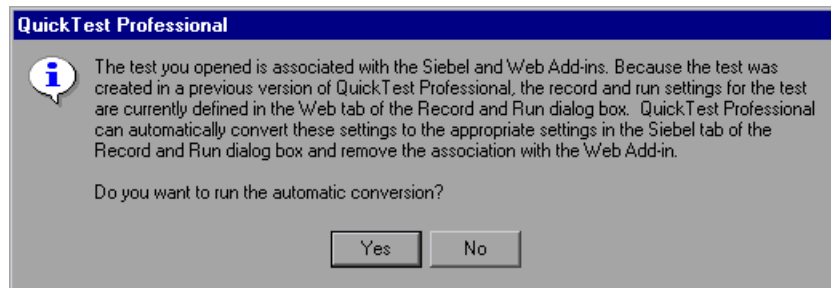
If you have the Siebel Add-in installed on QuickTest Professional, you can use QuickTest Professional to generate an object repository for your application. For more details, see Chapter 29, "Generating an Object Repository Using Siebel Test Express."

Information for Users of Earlier Versions of the QuickTest Professional Siebel Add-in

Tests created in earlier versions of the Siebel Add-in can be opened in QuickTest Professional Siebel Add-in. The Siebel Add-in provides an option to convert the test's old record and run settings to the new version settings automatically. In addition, the naming convention for Siebel test objects has been modified for QuickTest Professional Siebel Add-in.

Opening Siebel Add-in 6.5 Tests in Siebel Add-in

The first time you open a test that was created in an earlier version of the Siebel Add-in, a message opens asking whether you want to convert the 6.5 record and run settings to the appropriate settings automatically. This is because in Siebel Add-in 6.5, the record and run settings were defined in the Web tab of the Record and Run Settings dialog box, and now they need to be defined in the Siebel tab of the Record and Run Settings dialog box.



Click **Yes** to convert the record and run settings for the test automatically, or click **No** to leave the settings as they are. If you click **Yes**, the following settings are modified in the Siebel tab of the Record and Run Settings dialog box:

- ▶ The Siebel version is set as **Siebel eBusiness 7.0/7.5**.
- ▶ The browser settings and/or URL that were defined in the Web tab are transferred to the Siebel tab.
- ▶ The Web Add-in is removed from the list of add-ins associated with the test.

Note: If you choose not to convert the settings automatically, or if you choose to convert the settings but do not save the test before closing it, the message asking if you want to convert the settings is not displayed the next time you open the test. You can manually change the settings for a test at any time in the Siebel tab of the Record and Run Settings dialog box.

Troubleshooting & Limitations - Siebel Add-in

This section contains general troubleshooting and limitation information about the Siebel Add-in.

General

Recording on multiple Siebel application versions in the same computer may cause steps not to be recorded.

Checkpoints and the Object Spy

- ▶ To create a table content checkpoint or output value for the appropriate object type (for example, SiebList, SiebPicklist, or SiebPageTabs) when editing your test or component, you must open the application to the exact screen in which the object appears. Otherwise, only the Properties tab is displayed in the Table Checkpoint dialog box or Table Output Value dialog box.
- ▶ Checkpoints created for SiebList objects that contain a **Total** row may fail during a run session if the action that led to the update of the **Total** row was not recorded.
- ▶ The Object Spy and checkpoints identify expanded calculator and calendar popup objects as **Window("Siebel control popup")**.

This section also includes troubleshooting and limitation information about the following Siebel versions:

- ▶ "Siebel 7.7.x or Later" on page 469
- ▶ "Siebel 7.0.x and 7.5.x" on page 470

Siebel 7.7.x or Later

- ▶ Certain objects, methods, or properties may be available from within QuickTest even though they are not described in the documentation. This is because QuickTest retrieves the latest **SiebelObject.xml** file when loading the Siebel add-in and opening a Siebel application, and because the documentation is updated according to version of the XML file that is available at the time of the QuickTest product release.
- ▶ Certain objects, for example, in the SmartScript module, do not have a value for the repository name property and are therefore not recorded and are not recognized by the Object Spy.

Workaround: Use low-level recording.

- ▶ Gantt chart operations and RichText editor toolbar operations are not recorded.

Workaround: Use low-level recording.

- ▶ The appointment calendar object can be recorded only if the ActiveX Add-in is enabled.
- ▶ If you record the creation of a new appointment in an appointment calendar, the test or component may fail when you run it.

Workaround: Manually add an **onkeypress** FireEvent to the WebElement before the **Set** step.

- ▶ The Active Screen is empty for steps recorded on pop-up tables.
- ▶ Inner objects that are placed in cells of a SiebList object cannot be accessed in the standard way, even if they are recorded. This may cause the following limitations:
 - ▶ The entire SiebList object is highlighted if the test or component script line contains an operation on a SiebList inner object.
 - ▶ The **ChildObjects** method for SiebList objects returns 0.
 - ▶ The Add Objects option in the Object Repository dialog box cannot be used to add SiebList inner objects to the object repository.

- ▶ If a warning message opens while recording your test or component, for example, if you insert invalid data, QuickTest may record these operations in the incorrect order.

Workaround: Manually change the order of the steps in your test after recording.

- ▶ Context-sensitive help (F1 Help) may not be available for Siebel 7.7.x or later objects and/or methods that were added by Siebel after the QuickTest 11.00 release. In addition, auto-documentation (in the Keyword View Documentation column) and step documentation (in the Step Generator) may not be available for these objects and/or methods.

Siebel 7.0.x and 7.5.x

Creating and Running Testing Documents

- ▶ QuickTest does not support recording on Siebel applications using keyboard shortcuts.

Workaround: Use the mouse to record on Siebel applications.

- ▶ QuickTest does not record the scrolling of a set of records in an SblTable.

Workaround: While recording, scroll the table row by row.

Tip: You can use the Expert View to manually edit the statement to scroll multiple rows.

- ▶ By default, QuickTest does not record Editor control operations (used mainly in long **Description** fields).

Workaround: Use low-level recording, making sure you record the scrolling to the control if needed.

Working with Siebel Controls

- ▶ When you click the **Search** icon for the first time during a browser session, a frame opens that is different from all other search frames. When running test iterations, the correct frame may not be identified.

Workaround: Close the browser at the end of every iteration.

- ▶ Each Siebel version includes changes/modifications to the user interface. As a result, steps created on previous Siebel versions on elements that no longer exist in the interface will probably fail and should be replaced.

For example, the button arrow used to view the next set of records on the top line of the Siebel table that appears in earlier versions of Siebel was replaced in Siebel version 7.5.2 with a scroll bar at the side of the table. In this case, replace `Image("Next Record")`. Click with an operation on the scroll bar.

- ▶ The name of the first column in an `SblTable` object cannot be retrieved.

Workaround: Use the column index to perform the operation on the cells in the first column.

Standard-Interactivity (SI) Applications

- ▶ In some SI application dialog boxes, in cases where selecting a check box causes a navigation to occur (for example, in a check box table column, such as the **New** column), QuickTest may not record the subsequent steps or may record them inaccurately.

Workaround: To continue recording accurately, click anywhere in the page before the next operation.

- ▶ When recording on a Currency Calculator pop-up control, clicking **OK** immediately after entering a currency value may result in a recording error.

Workaround: Before clicking **OK** in a Currency Calculator pop-up control within a `SblAdvancedEdit` object, select another control within the pop-up and click **OK**.

High-Interactivity (HI) Applications

- ▶ Depending on your browser's security settings and the Siebel patches that are installed, several dialog boxes may open when logging in to your Siebel application. It is recommended to run tests or components when all required Siebel patches are downloaded and installed. If for some reason, you cannot do this, manually delete the **Sync** steps added between the steps recorded on the security alerts.

- ▶ QuickTest cannot record a **SblTable.Sort** operation if it is the first operation inside an MVG (Multi-Value Group) applet.

Workaround: Click anywhere in the MVG applet and then sort it.

- ▶ When recording on a **SblAdvancedEdit** object that opens a pop-up object, QuickTest records only the **Set** method and does not record the operations within the pop-up object. However, if you open a table from the pop-up object, QuickTest does record the operations performed within this secondary table. These statements are not required in the test or component, since the operation of inserting the Pickup table selected item into the main table is also recorded. In some cases, these redundant statements interfere with the run session.

Workaround: If the test or component does not run as expected, delete the statements recorded on secondary tables opened from a pop-up object.

- ▶ When adding an attachment to a Siebel table, QuickTest records additional statements that may interfere with the run session.

Workaround: After recording, delete the **OpenCellElement** and **Add** statements that were recorded when you added an attachment.

- ▶ When inserting a value into a Siebel table cell using the Currency Calculator control, QuickTest may record a new **SelectCell** step before the **SetCellData** if you move the cursor to another cell before clicking in the cell in which you entered a value.

Workaround: While recording, always close the Currency Calculator by pressing the ENTER key. If, for some reason, the Currency Calculator was not closed using the ENTER key, you can manually change the order between the **SetCellData** and **SelectCell** steps.

28

Enhancing Your Siebel Test or Component

After you create your test or component, you can enhance it by adding checkpoints, retrieving output values, parameterizing values, and inserting Siebel objects, methods and properties.

This chapter includes:

- Considerations for Checking Siebel Objects on page 473
- Accessing Native Operations and Properties in Siebel 7.0.x and 7.5.x Applications on page 475
- Spooling Data from a Siebel Table on page 476

Considerations for Checking Siebel Objects

- You check most Siebel objects or output their property values in the same way as you do for other objects supported by QuickTest, with exceptions for SblTable objects and **Sieb** tabular test objects.

Note: Table checkpoints are not supported for business components.

- ▶ You check **SblTable** objects and output their values in the same way as you do for other table objects supported by QuickTest—using the Table Checkpoint Properties dialog box or Table Output Value Properties dialog box—with the following differences:
 - ▶ In Siebel 7.0.x or 7.5.x high-interactivity applications, you must have your Siebel application open to the page that contains the table while creating a table checkpoint or output value.

When creating table checkpoints or output values, do not include the header line of the SblTable object when selecting cells to check or output. To clear the selection in this first row of cells, double-click row heading **1** to the left of the table.

Double-click to clear all cells in the row

	1	2	3	4	5	6	7
1		New	Last Nam	First Nam	Job Title	Email	Work Pho

Tip: When working with SblTable objects, you can spool all of the visible data from a table into an external file. For more information, see "Spooling Data from a Siebel Table" on page 476.

- ▶ Specific test objects in Siebel 7.7.x applications (with **Sieb** prefixes) have tabular characteristics. QuickTest treats **Sieb** tabular test objects as table-type objects and enables you to check both their content and/or their identification properties. You can also output content and/or identification property values for use in your test or component. The following **Sieb** test objects have tabular characteristics: **SiebCommunicationsToolbar**, **SiebList**, **SiebMenu**, **SiebPageTabs**, **SiebPDQ**, **SiebPicklist**, **SiebScreenViews**, **SiebThreadbar**, **SiebToolbar**, and **SiebViewApplets**.

Tip: When working with **Sieb** tabular objects, you can spool all of the visible data from the object into an external file. For more information, see "Spooling Data from a Siebel Table" on page 476.

- ▶ When testing high-interactivity applications:
 - ▶ If the **Sieb** tabular object is not open in your Siebel application when you create a checkpoint, the Table Checkpoint Properties dialog box contains only the Properties tab and the option to select which type of information to check (content or properties) is disabled.
 - ▶ If the **Sieb** tabular object is not open in your Siebel application when you create the output value, the Table Output Value Properties dialog box contains only the Properties tab, and the option to select which type of information to output (content or properties) is disabled.
- ▶ If you want to access an inner object contained in a **SiebList** object, hold the CTRL key while you click the **SiebList** object with the pointing hand mechanism.

Accessing Native Operations and Properties in Siebel 7.0.x and 7.5.x Applications

In addition to the Siebel-specific test objects and operations, you can also use the **Object** property to access native (internal) operations and properties of the HTML or ActiveX elements that wrap Siebel objects. The **Object** property is available for all Siebel 7.0.x and 7.5.x objects.

Tip: You can use the Object Spy to view the native operations and properties of an object in your application.

The **Object** property is also useful for checking the value of properties that are not available using a standard Siebel checkpoint.

The following example uses the **Object** property to access the raw HTML element that represents the SblTabStrip object, retrieve its HTML tag name and size, and display this information in message boxes.

```
set obj = Browser("Siebel Call").Page("Siebel Call").Frame("Siebel Call").
SblTabStrip("ScreenTabStrip").Object
msgbox obj.tagName
msgbox obj.height
msgbox obj.width
```

Note: Relying on native properties may be problematic if you are upgrading your Siebel application to a newer version, in which objects may have a different structure. For example, the conversion of HTML objects to ActiveX objects in the Internet Explorer Option Pack.

For more information on using the Object property, see the *HP QuickTest Professional User Guide*.

Spooling Data from a Siebel Table

If you want to spool all the visible data from a **SblTable** or a **Sieb** tabular object (such as a **SiebList** object) into an external file, you can loop through each cell in the table and then save the information to an external file.

The following example uses the **GetCellData** method to list the data of each cell in a SblTable object with 10 rows and 10 columns:

```
For i=0 to 10
  For j=0 to 10
    Dat=Browser("Siebel eChannel").Page("Siebel eChannel_8").
    Frame("Campaign Explorer").SblTable("Campaign").
    GetCellData (i, j)
    SaveToExternalFile (Dat)
  Next
Next
```

The following example uses the **RowCount** and **ColumnsCount** methods to list the data of each cell in a SiebList object:

```

RowCount = SiebApplication("Siebel Call Center").
    SiebScreen("Accounts").SiebView("My Accounts").
    SiebApplet("Accounts").SiebList("List").RowCount
ColsCount = SiebApplication("Siebel Call Center").
    SiebScreen("Accounts").SiebView("My Accounts").
    SiebApplet("Accounts").SiebList("List").ColumnsCount
For i=0 to RowCount-1
    For j=0 to ColsCount-1
        ColumnName = SiebApplication("Siebel Call Center").
            SiebScreen("Accounts").SiebView("My Accounts").
            SiebApplet("Accounts").SiebList("List").
            GetColumnRepositoryNameByIndex(j)
        Dat=SiebApplication("Siebel Call Center").SiebScreen("Accounts").
            SiebView("My Accounts").SiebApplet("Accounts").
            SiebList("List").GetCellText(ColumnName,i)
        SaveToExternalFile (Dat)
    Next
Next

```

For more information on the **GetCellData**, **RowCount**, and **ColumnsCount** methods, see the **Siebel** section of the *HP QuickTest Professional Object Model Reference*.

29

Generating an Object Repository Using Siebel Test Express

You can use Siebel Test Express to automatically generate or update an object repository.

This chapter includes:

- ▶ About Generating an Object Repository Using Siebel Test Express on page 480
- ▶ Siebel Test Express System Requirements and Supported Environments on page 481
- ▶ Using Siebel Test Express to Create an Object Repository on page 481
- ▶ Using Siebel Test Express to Update an Existing Object Repository on page 490

About Generating an Object Repository Using Siebel Test Express

If you have the Siebel Add-in installed on QuickTest Professional, you can use Siebel Test Express to automatically generate a new shared object repository, or update an existing object repository.

You can create new shared object repositories using the Create Object Repository Wizard. Using the wizard you can select the applications or top-level application objects for which to create an object repository. Siebel Test Express scans the Siebel application and creates test objects for every child object contained in the applications or top-level objects that you specify. After you have created the shared object repository, you can save it to the file system or to a Quality Center project using the Object Repository Manager.

You can also use Siebel Test Express to update an existing object repository. The Update Object Repository Wizard enables you to select the applications or top-level objects to include in the update, as well as the date from which to search for and include new or modified objects. The date refers to when the objects were last added or modified in the object repository.

After you have updated an object repository, the Object Repository Merge Tool merges the new and modified objects with objects from your existing object repository.

This chapter explains how to create or update an object repository using Siebel Test Express. For more information on working with object repositories in general, see the *HP QuickTest Professional User Guide*.

Siebel Test Express System Requirements and Supported Environments

To successfully run Siebel Test Express, the Siebel Add-in must be installed and loaded.

Siebel Test Express supports Siebel 7.7 or later high-interactivity applications that are based on the Siebel Test Automation API.

Note: To work with Siebel Test Express in QuickTest, ensure that the Siebel Test Automation API version installed on your server is one that supports Siebel Test Express.

Using Siebel Test Express to Create an Object Repository

You can use Siebel Test Express to generate a new shared object repository for a Siebel application. After you have created the shared object repository, you can save it to the file system or to a Quality Center project. For information on working with or saving shared object repositories, see the section on the Object Repository Manager in the *HP QuickTest Professional User Guide*.

To create a shared object repository using Siebel Test Express:

- 1** Select **Resources > Object Repository Manager**. The Object Repository Manager opens.
- 2** In the Object Repository Manager, select **Tools > Siebel Test Express > Create Object Repository** or click the **Create Object Repository** button on the Object Repository Manager toolbar. The Create Object Repository Wizard opens to the Connection Information screen.



- 3 Follow the steps of the wizard to create the new shared object repository.

Note: You can run only one instance of the Siebel Test Express Object Repository Wizard on a computer at any given time.

For more information on using the Create Object Repository Wizard, see:

- "Understanding the Connection Information Screen" on page 483
- "Understanding the Screen Selection Screen For Creating Object Repositories" on page 485
- "Understanding the Importing Test Objects Screen" on page 487
- "Understanding the Object Repository Created Screen" on page 489

Understanding the Connection Information Screen

In the Connection Information screen you enter the connection information for logging in to the Siebel server. This information is saved as meta data in the generated object repository file, and is automatically entered for you when you use the Create Object Repository wizard to update the same object repository.

Note:

- ▶ If you are creating a new repository, the information that was entered in this screen the last time you used the wizard is automatically entered.
 - ▶ If you are updating the repository, the information that was saved as meta data with the repository file is entered in this screen.
-

The data required in this screen is not necessarily the same as the data you use to log into the Siebel application as a user. You should request the information for this screen from your Siebel server administrator.

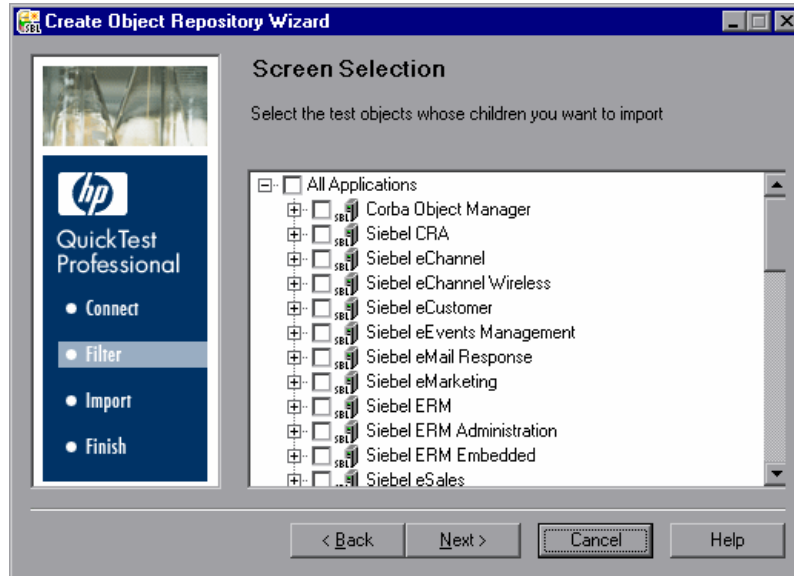
The Connection Information screen contains the following options:

Connect Information	Description
Server URL	The URL of the Siebel server (including http://).
User name	Your user name.
Password	Your password.
Database name	The name of the Siebel database.
Table owner	The table owner you want to use for the specified Siebel database.
Siebel repository	Optional. The name of the Siebel repository. If you do not enter a name, Siebel will use a default name.

Note: While the Connection Information screen is displayed, you cannot make the Object Repository Manager or QuickTest windows active.

Understanding the Screen Selection Screen For Creating Object Repositories

The Screen Selection screen in the Create Object Repository Wizard displays a list of all the available applications, according to the connection information entered in the Connection Information screen.



You can select the applications for which to create the object repository or you can expand the application node and then select one or more top-level objects. It is recommended to select only the top-level objects that you need. Importing an entire application may take a very long time.

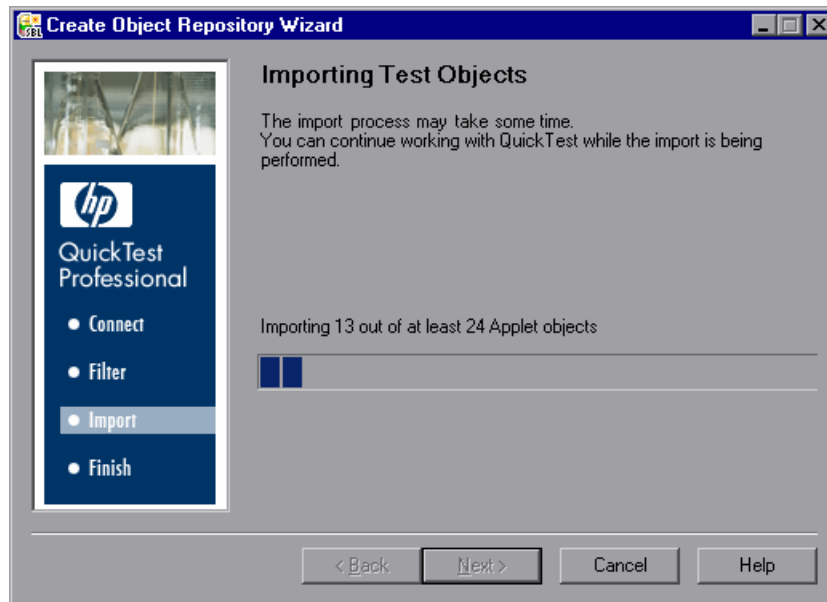
When Siebel Test Express creates the object repository, it imports the information and creates test objects for all children (descendants) of the applications or top-level objects you select in this screen.

If the last used profile from a previous import operation was saved, the profile is loaded, and you can edit the selected options, as required. The new selections are saved in the profile to be used for future import operations on the same object repository.

Note: While the Screen Selection screen is displayed, you cannot make the Object Repository Manager or QuickTest windows active. Once you click **Next** in this screen, you will be able to switch the focus to the Object Repository Manager or QuickTest, and you can work in either of these windows while the wizard is generating the object repository. However, do not close the Object Repository Manager or QuickTest. If you do try closing either window, a message is displayed, warning that the object repository generation process will be stopped and all data will be lost. Click **No** to continue creating the object repository.

Understanding the Importing Test Objects Screen

The Importing Test Objects screen shows the progress of the import process. The number imported indicates the number of applet objects that have already been imported, including all child objects of that applet. At the same time that the wizard imports the objects, it is also retrieving information about the total number of applets it needs to import. While the wizard is retrieving this information, the total number changes and the words *at least* show that the wizard is still retrieving information. When the total number is known, the words *at least* are no longer displayed.



During the import process, you can cancel the operation if required. If you cancel the operation, a message is displayed notifying you that stopping the import process will result in an incomplete object repository.

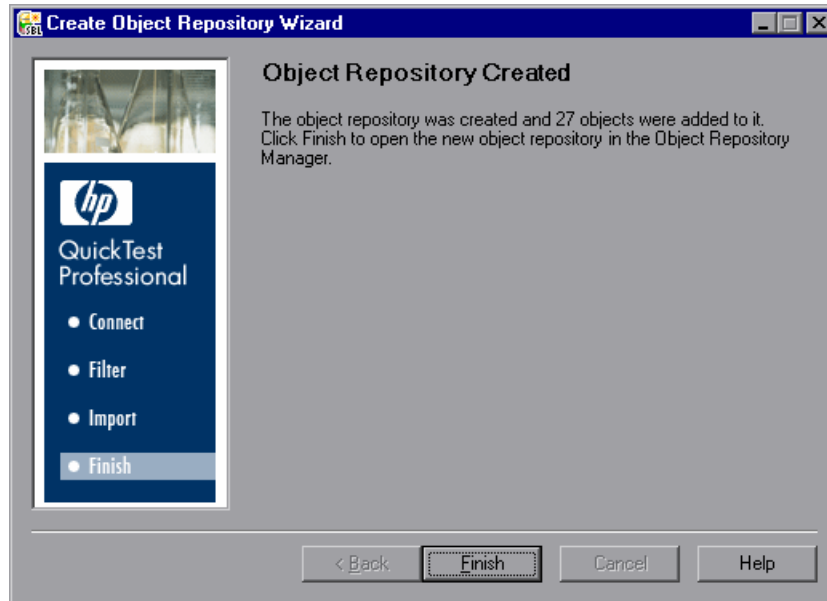
- ▶ Click **Yes** to save the partially imported repository
- ▶ Click **No** to discard the partially imported repository
- ▶ Click **Cancel** in the message box to continue importing objects to the repository

Notes:

- ▶ Importing the object repository can take up to several hours, depending on the size of the repository.
 - ▶ You can work in either the Object Repository Manager or QuickTest while the wizard is generating the object repository. However, do not close either window. If you do try closing either window, a message is displayed, warning that the object repository generation process will be stopped and all data will be lost. Click **No** to continue generating the object repository.
 - ▶ While the wizard is updating the object repository, that object repository file is locked and you cannot modify it in the Object Repository Manager.
-

Understanding the Object Repository Created Screen

The Object Repository Created screen opens after all the objects have been imported. It displays the total number of objects added to the repository.



Note: If any errors occur during the import process, a warning and an **Error Log** button are displayed in the Object Repository Created screen. The log contains error and exception data from the Siebel server listing the failed calls and the object that caused the error. Click the **Error Log** button to save the error log. By default, the error log is called **TestExpressErrorLog.xml**, and it is saved in the **<QuickTest Professional>\Tests** folder.

Click **Finish** to close the wizard and display the new object repository in the Object Repository Manager. This can take a few minutes.

Using Siebel Test Express to Update an Existing Object Repository

You can use Siebel Test Express to update an existing shared object repository. After you have updated your shared object repository, the Object Repository Merge Tool merges the objects in your new object repository with your existing object repository.

To use update an existing object repository using Siebel Test Express:

1 Select **Resources > Object Repository Manager**. The Object Repository Manager opens.



2 Select **File > Open** or click the **Open** button. The Open Shared Object Repository dialog box opens.

3 Select the object repository file you want to update, and click **Open**. The object repository file opens.

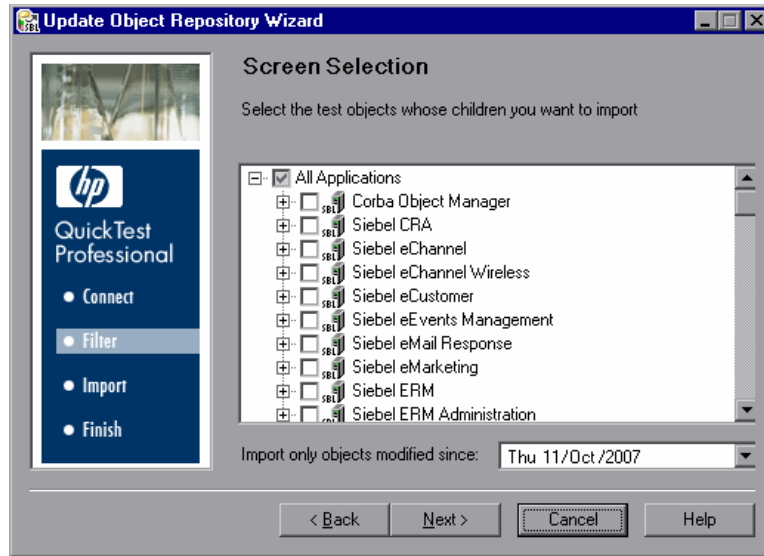
By default, the object repository file opens in read-only mode. To update a repository, it must be open in editable format. You can open it in editable format by clearing the **Open in read-only mode** check box in the Open Shared Object Repository dialog box. You can also enable editing by choosing **File > Enable Editing** after you open the repository.



4 Select **Tools > Siebel Test Express > Update Object Repository** or click the **Update Object Repository** button on the Object Repository Manager toolbar. The Update Object Repository Wizard opens to the Connection Information screen. For more information on the Connection Information screen, see "Understanding the Connection Information Screen" on page 483.

Note: While the Connection Information screen is displayed, you cannot make the Object Repository Manager or QuickTest windows active.

- 5 Click **Next**. The Screen Selection screen opens. The Screen Selection screen displays the filter of selected objects that was stored when the object repository was created (or last updated), as well as the date on which the object repository was created or last updated.



Keep the displayed date or select a new date by clicking the arrow next to the **Import only objects modified since** box, and select the date from the displayed calendar. All objects modified before the selected date are ignored during the import process. Using this option can speed up the importing process.

Note: If you select objects other than the ones that were imported in the previous wizard session, then only objects modified since the selected date are imported.

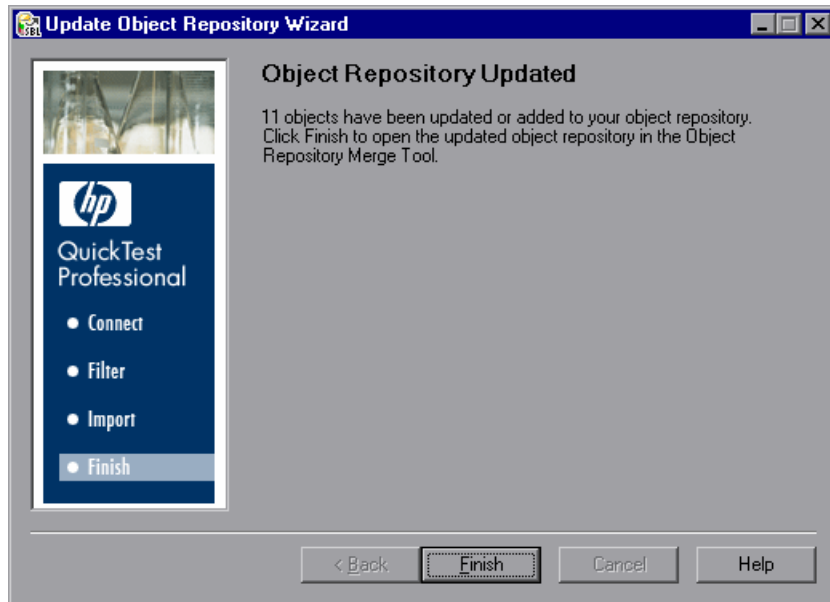
If you want to add objects to your object repository that were not previously imported, use the New Object Repository Wizard to import only those objects and then merge the generated object repository with your current object repository.

- 6 Click **Next**. The Importing Test Objects screen opens. For more information on the Importing Test Objects screen, see "Understanding the Importing Test Objects Screen" on page 487.
-

Note: You can work in either the Object Repository Manager or QuickTest while the wizard is generating the object repository. However, do not close either window. If you do try closing either window, a message is displayed, warning that the object repository generation process will be stopped and all data will be lost. Click **No** to continue generating the object repository.

While the wizard is updating the object repository, that object repository file is locked and you cannot modify it in the Object Repository Manager.

- 7 After all the objects have been imported, the Object Repository Updated screen opens and indicates the number of objects that have been added or modified in the object repository.



- 8 Click **Finish** to disconnect from Siebel Test Express. If one or more objects were added or updated during the process, the Object Repository Merge Tool opens. This can take a few minutes.

Using the Object Repository Merge Tool to Merge an Updated Siebel Object Repository

The Object Repository Merge Tool merges the updated local object repository (secondary repository) with your existing shared object repository (primary repository) to create a new shared object repository file (target repository).

Conflicts between objects in the primary and secondary repository files are resolved automatically by the Merge Tool according to the default resolution settings and the Merge Tool displays the Statistics dialog box, which lists the files that were merged, and the number and type of any conflicts that were resolved during the merge. You can accept or modify these resolutions to match your needs.

When you have ensured that the object conflicts are resolved satisfactorily, you can save the target repository to the file system, or to a Quality Center project (if QuickTest is currently connected to a Quality Center project).

For more information on working with the Object Repository Merge Tool, see the *HP QuickTest Professional User Guide*.

Part II

Standard Windows Testing Support

30

Using Standard Windows Testing Support

You can use the standard Windows testing support provided by QuickTest to test objects (controls) developed using the Win32 API or MFC platforms. QuickTest standard Windows testing support is built-in and does not require you to load any QuickTest add-in.

QuickTest also uses built-in standard Windows testing support and standard Windows test objects to identify objects from other environments if the relevant add-in is not installed and loaded.

Standard Windows testing support provides test objects, methods, and properties that can be used when testing objects in standard Windows applications. For more information, see the **Standard Windows** section of the *HP QuickTest Professional Object Model Reference*.

The following table summarizes basic information about standard Windows testing support and how it relates to some commonly-used aspects of QuickTest.

General Information	
Add-in Type	The standard Windows testing support functions like a Windows-based add-in. Much of its functionality is the same as other Windows-based add-ins. See "Testing Windows-Based Applications" on page 87
Checkpoints and Output Values	<ul style="list-style-type: none"> ▶ See the sections describing checkpoints and output values in the <i>HP QuickTest Professional User Guide</i> (for tests), and the <i>HP QuickTest Professional for Business Process Testing User Guide</i> (for components). ▶ See "Supported Checkpoints and Output Values Per Add-in" on page 769.
Prerequisites	
Opening Your Application	You can open your standard Windows application before or after opening QuickTest. Standard Windows testing support is always loaded in QuickTest. It is therefore not an available option in the Add-in Manager.
Add-in Dependencies	None

Setting Preferences	
Options Dialog Box	Use the Windows Applications pane. (Tools > Options > Windows Applications node) See "The Options Dialog Box: Windows Applications Pane" on page 103.
Record and Run Settings Dialog Box (tests only)	Use the Windows Applications tab. (Automation > Record and Run Settings) See "The Record and Run Settings Dialog Box: Windows Applications Tab" on page 90. Note: QuickTest recognizes standard Windows objects only in applications that are opened after changing the settings in the Windows Applications tab of the Record and Run Settings dialog box.
Custom Active Screen Capture Settings Dialog Box (tests only)	Use the Windows applications section. (Tools > Options > Active Screen node > Custom Level) See the section on the Custom Active Screen Capture Settings dialog box in the <i>HP QuickTest Professional User Guide</i> .
Application Area Settings Dialog Box (components only)	Use the Applications pane. (File > Settings > Applications node) See the section on defining Application Settings for your application area in the <i>HP QuickTest Professional for Business Process Testing User Guide</i> .

This chapter includes:

Troubleshooting and Limitations - Standard Windows on page 500

Troubleshooting and Limitations - Standard Windows

This section describes troubleshooting and limitations for working with Standard Windows test objects.

- ▶ When recording on WinMenu objects, the Active Screen is not captured.
- ▶ You cannot insert a checkpoint on a WinMenu object.

Workaround: Use the **CheckProperty** and **CheckItemProperty** methods to check specific property and item property values.

- ▶ If you record using Windows logo key shortcuts, the recording may be inaccurate.

Workaround: Use the **Start** menu instead of the Windows logo key when recording.

- ▶ Changing the style of a WinCalendar (from single selection to multi-selection, for example) will cause the run session to fail.
- ▶ When using the pointing hand mechanism from the Object Spy to point to MFC static text or tab controls, QuickTest may fail to return the correct object.

Workaround: Add the object to the object repository. To do this, point to the object's parent window, select the parent window object in the Object Selection dialog box, click **OK**, and perform one of the following in the Define Object Filter dialog box:

- ▶ Select the **All object types** option to add all of the objects in the parent window to the object repository.
- ▶ Select the **Selected object types** option, click the **Select** button, and then select the specific object type(s) you want to add to the object repository.

After you add the object to the object repository, you can use the **GetROProperty** method to retrieve the run-time values of its properties.

For example:

```
width = Dialog("Login").Static("Agent Name:").GetROProperty("width")  
MsgBox width
```

- ▶ Checkpoints are not supported for WinComboBox objects of style Simple ComboBox.

Part III

The Stingray Add-in

31

Using the Stingray Add-in

You can use the Stingray Add-in to test Stingray objects (controls).

The QuickTest Professional Stingray Add-in recognizes and records on supported Stingray Objective Grid and Stingray Objective Toolkit controls. For details on supported Stingray environments, see the **Stingray Add-in** section of the *HP QuickTest Professional Product Availability Matrix*, available from the Documentation Library Home page or the root folder of the QuickTest Professional DVD.

The Stingray Add-in uses a sub-set of the standard Windows test objects, methods, and properties, which can be used when testing objects (controls) in Stingray applications. For more information, see the **Stingray** section of the *HP QuickTest Professional Object Model Reference*.

The following table summarizes basic information about the Stingray Add-in and how it relates to some commonly-used aspects of QuickTest.

General Information	
Add-in Type	This is a Windows-based add-in. Much of its functionality is the same as other Windows-based add-ins. See "Testing Windows-Based Applications" on page 87.
Checkpoints and Output Values	<ul style="list-style-type: none"> ▶ See the sections describing checkpoints and output values in the <i>HP QuickTest Professional User Guide</i> (for tests), and the <i>HP QuickTest Professional for Business Process Testing User Guide</i> (for components). ▶ See "Supported Checkpoints and Output Values Per Add-in" on page 769.
Prerequisites	
Opening Your Application	You can open your Stingray application before or after opening QuickTest.
Add-in Dependencies	None
Other	<p>You must configure the Stingray Add-in to work with your application. See:</p> <ul style="list-style-type: none"> ▶ "Setting Up Stingray Object Support" on page 507. ▶ "Running the Stingray Support Configuration Wizard" on page 512.
Setting Preferences	
Options Dialog Box	<ul style="list-style-type: none"> ▶ Use the Stingray pane. (Tools > Options > Stingray node) See "Configuring Stingray Options" on page 521. ▶ Use the Windows Applications pane. (Tools > Options > Windows Applications node) See "The Options Dialog Box: Windows Applications Pane" on page 103.

<p>Record and Run Settings Dialog Box (tests only)</p>	<p>Use the Windows Applications tab. (Automation > Record and Run Settings)</p> <p>See "The Record and Run Settings Dialog Box: Windows Applications Tab" on page 90.</p> <p>Notes:</p> <ul style="list-style-type: none"> ▶ In addition to the settings in the Record and Run Settings dialog box, you must also configure QuickTest to recognize your Stingray applications in the Stingray pane of the Options dialog box. For more information, see "Configuring Stingray Options" on page 521. ▶ If you select the Record and Run only on radio button in the Record and Run Settings dialog box, the settings also apply to (limit) the applications that are recognized for Object Spy and other pointing hand operations.
<p>Custom Active Screen Capture Settings Dialog Box (tests only)</p>	<p>Use the Windows applications section. (Tools > Options > Active Screen node > Custom Level)</p> <p>See the section on the Custom Active Screen Capture Settings dialog box in the <i>HP QuickTest Professional User Guide</i>.</p>
<p>Application Area Settings Dialog Box (components only)</p>	<p>Use the Applications pane. (File > Settings > Applications node)</p> <p>See the section on defining Application Settings for your application area in the <i>HP QuickTest Professional for Business Process Testing User Guide</i>.</p>

This chapter includes:

- ▶ Considerations for Working with the Stingray Add-in on page 506
- ▶ Setting Up Stingray Object Support on page 507
- ▶ Configuring Stingray Options on page 521
- ▶ Troubleshooting and Limitations - Stingray Add-in on page 527

Considerations for Working with the Stingray Add-in

QuickTest stores Stingray support configuration for each configured Stingray application separately. By default, QuickTest uses the latest configured Stingray agent version for all Stingray applications except those applications that are already configured.

For example, suppose you have two Stingray applications; application `grid1.exe` that uses Stingray Grid control version 9.03, and application `tree1.exe` that uses Stingray TreeView control version 11.00.

You can configure QuickTest to support both applications as follows:

- 1** Run the Stingray Support Configuration Wizard and configure support for the `grid1.exe` application. QuickTest saves the configuration for this application.
- 2** Run the Stingray Support Configuration Wizard again and configure support for the `tree1.exe` application. QuickTest saves the configuration for this application.

After performing these steps, QuickTest will support the `grid1.exe` application and support all Stingray applications that have Stingray TreeView controls version 11.00, including the `tree1.exe` application.

Setting Up Stingray Object Support

Before you begin working, you need to configure the Stingray Add-in to work with your application. QuickTest Professional support for Stingray objects is based on an agent entity that exists in the Stingray application. This agent interacts with QuickTest to enable record and run operations. There are two different modes for establishing the agent entity:

- ▶ **Run-time Agent Mode.** QuickTest injects an agent DLL into the application's process during run-time. This is the recommended mode. For more information, see "Understanding the Run-time Agent (Agent DLL)" on page 508.
- ▶ **Precompiled Agent Mode.** You make slight modifications to your Visual C++ project in addition to configuring the Stingray Add-in. Use this mode only if the run-time agent mode is unsuitable or cannot be used. For more information, see "Using the Precompiled Agent Mode" on page 509.

You choose your preferred mode and configure support for the Stingray Add-in using the Stingray Support Configuration Wizard. For more information, see "Running the Stingray Support Configuration Wizard" on page 512.

After you configure support for the Stingray Add-in, you can fine-tune the configuration options, if needed. For more information, see "Configuring Stingray Options" on page 521.

This section describes:

- ▶ "Understanding the Run-time Agent (Agent DLL)" on page 508
- ▶ "Using the Precompiled Agent Mode" on page 509
- ▶ "Running the Stingray Support Configuration Wizard" on page 512

Understanding the Run-time Agent (Agent DLL)

When you choose the run-time agent mode, QuickTest injects an agent DLL into the application's process during run-time. This recommended mode is non-intrusive and does not require any modifications to the source code of the application being tested.

You can use the run-time agent mode only with Stingray applications that are created with dynamically-linked MFC libraries. You can verify if your MFC libraries are linked dynamically or statically by launching the Stingray Support Configuration Wizard. If the wizard identifies that your Stingray application uses statically-linked MFC libraries, it issues a warning.

The run-time agent mode supports the most commonly used major Stingray versions, as well as some—but not all—minor versions. For a list of supported version combinations, see the *HP QuickTest Professional Product Availability Matrix*, available from the Documentation Library Home page or the root folder of the QuickTest Professional DVD. You can also verify if your Stingray application version is supported by launching the Stingray Support Configuration Wizard. If the wizard identifies that your Stingray application version is not supported, it issues a warning.

Note: The Stingray Add-in is designed to support only applications that are compiled in Release mode.

If you cannot use the run-time agent mode for any reason, you can still work with your Stingray application using the precompiled agent mode, instead. For more information, see "Using the Precompiled Agent Mode" on page 509, or contact HP Software Support.

Using the Precompiled Agent Mode

If your application is statically linked with the MFC libraries, you can use the precompiled agent mode to enable Stingray object support. The precompiled agent mode requires you to make slight modifications to your Visual C++ project to enable QuickTest to support your Stingray application. If you select the precompiled agent mode in the Stingray Support Configuration Wizard, you can compile your project using the Stingray Add-in agent files.

Note: If your Stingray application project was compiled with an earlier version of the Stingray Add-in agent, your project already contains the required support code. To take advantage of the latest functionality provided with this add-in, it is recommended to remove the existing Stingray Add-in agent files from your project and recompile using the latest agent files.

When working with the precompiled agent mode, both Stingray Objective Grid and Stingray Objective Toolkit must be installed on your computer, even if your application contains only one type of Stingray control, such as a grid control or a tab control. The installed versions must match the version combinations supported for this add-in. For a list of supported version combinations, see the *HP QuickTest Professional Product Availability Matrix*, available from the Documentation Library Home page or the root folder of the QuickTest Professional DVD.

Note: If you do not have the required Stingray Objective Grid and Stingray Objective Toolkit version combination, contact HP Software Support for assistance.

Setting up Stingray support using the precompiled agent mode requires adding one support header file to your application's Visual C++ project and copying one library file to your Visual C++ project directory. After you complete these steps, you can compile your application, as usual.

Note: Use the precompiled agent mode only if the run-time agent mode is unsuitable or cannot be used.

To set up your project using the precompiled agent mode:

- 1** If your Stingray application was previously compiled with agent files from an earlier version of the Stingray Add-in, it is recommended to remove the existing agent files from your project. Otherwise, start from step 2.

Note: If you choose not to replace your existing Stingray Add-in agent files with the latest agent files, do not continue with this procedure. Although you will be able to work with the QuickTest Professional Stingray Add-in, you will not be able to take advantage of the latest functionality.

- 2** Copy the **StgAgentLib.h** header file from **<QuickTest Installation Folder>\bin\StingrayAgent\AgentLib\src\StgAgentLib.h** to your Visual C++ project directory. (You can optionally add the header file to the list of header files in your workspace.)
- 3** Check the version of the Stingray Objective Grid or Stingray Objective Toolkit used by your application and search for the corresponding support library file, **StgAgentLib.lib**.

For example, if your application is not compiled in Unicode and uses Objective Grid version 9.03 and Objective Toolkit version 8.03 linked with MFC version 7.1, search for the library file in: **<QuickTest Installation Folder>\bin\StingrayAgent\AgentLib\bin\MFC71\OG903_OT803**

If the application is linked with MFC80, is compiled in Unicode and uses Objective Grid version 10.0 and Objective Toolkit version 9.0, search for the library file in: <QuickTest Installation Folder>\bin\StingrayAgent\AgentLib\bin\MFC80\OG1000U_OT900U

Note: Each support library file specifies a combination of Objective Grid and Objective Toolkit versions. You must choose a combination of Objective Grid or Objective Toolkit versions, even if your application uses only one of these Stingray tools. For a list of supported Stingray version combinations, see the *HP QuickTest Professional Product Availability Matrix*, available from the Documentation Library Home page or the root folder of the QuickTest Professional DVD.

- 4 Copy the **StgAgentLib.lib** support library file to your Visual C++ project directory.
 - 5 Add the **#include "StgAgentLib.h"** statement to one of your **cpp** files, such as, **MainFrm.cpp**.
 - 6 Insert the **ReleaseWRVC();** function call in one of the functions called when your application terminates, for example, **CMainFrame::OnDestroy()**.
-

Note: Inserting this function call instructs the agent to perform required clean up operations related to the support library code.

When you build your application executable, the added header file automatically links the **StgAgentLib.lib** support library to your application statically, enabling the library code to be activated automatically during the run session.

- 7 Make sure that the **Precompiled Agent** option is selected in the Stingray Support Configuration Wizard. For more information, see "Running the Stingray Support Configuration Wizard" on page 512.

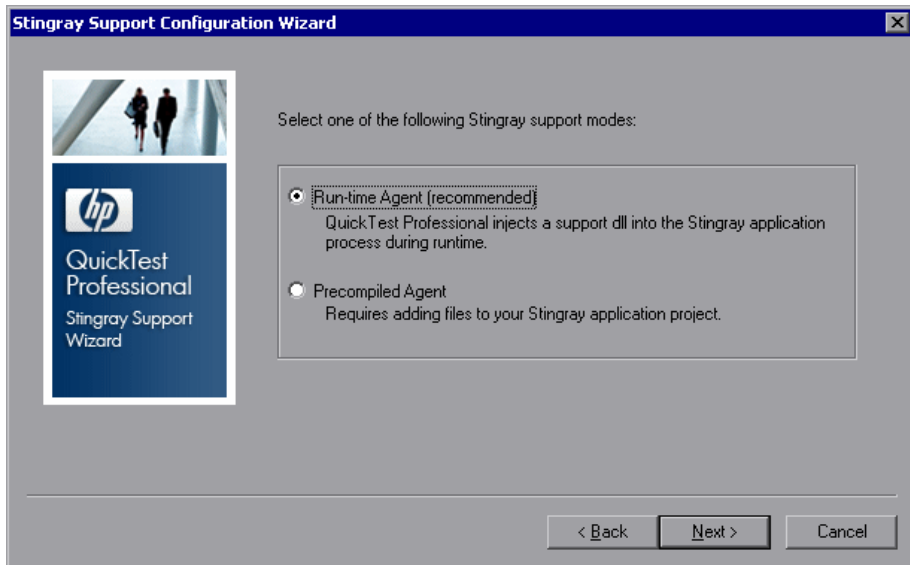
Running the Stingray Support Configuration Wizard

After the installation of QuickTest, you can open the Stingray Support Configuration Wizard from the Additional Installation Requirements dialog box. You can also open the wizard later by choosing it from the QuickTest program group or by activating it from the Stingray pane of the Options dialog box.

The wizard walks you through the steps that are necessary to configure QuickTest to work according to the agent mode that you select.

To run the Stingray Support Configuration Wizard:

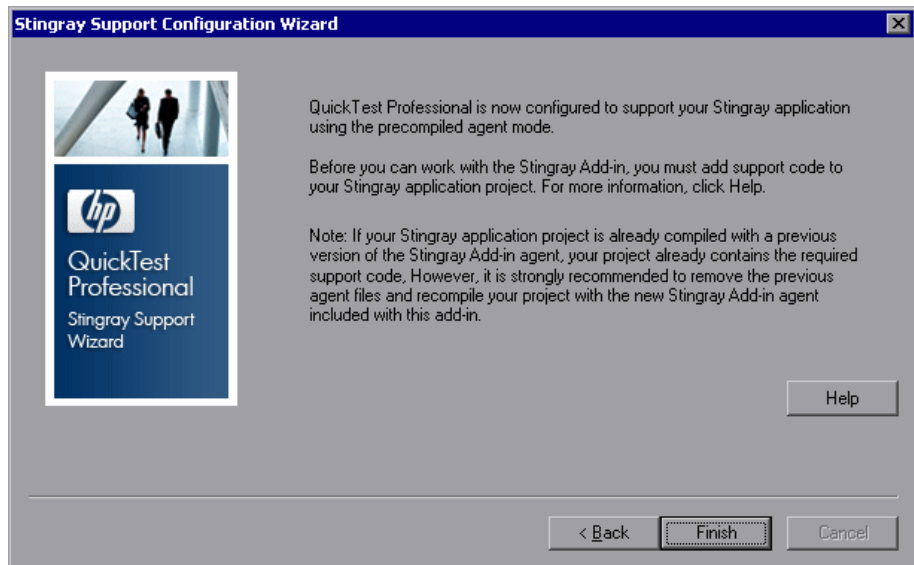
- 1 From the **Start** menu, select **Programs >HP QuickTest Professional > Stingray Support Configuration Wizard**.
- 2 In the Stingray Support Configuration Wizard welcome screen, click **Next**. The mode selection screen is displayed.



3 Select one of the following support modes:

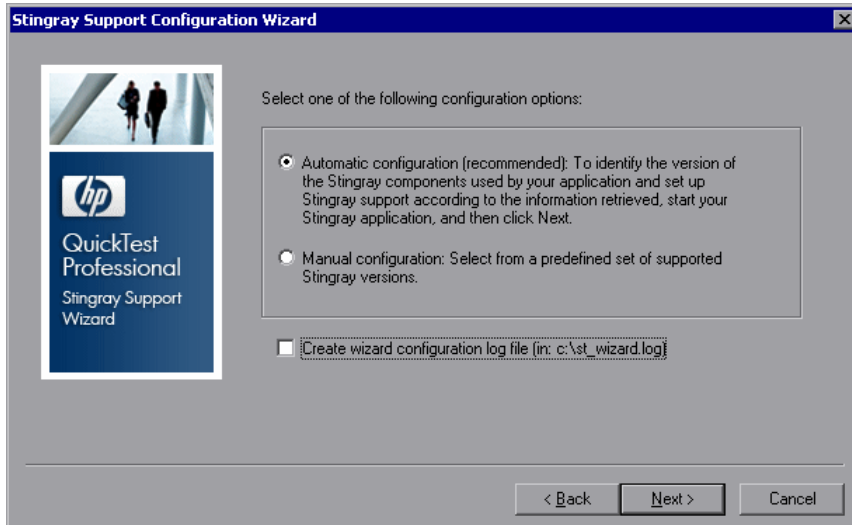
- **Run-time Agent.** A simple, non-intrusive mode that adds a support DLL to the Stingray application process during run-time. This is the recommended mode. If you select this mode, click **Next** and proceed to step 5 on page 514.
- **Precompiled Agent.** A mode that requires you to make slight modifications to your Stingray application project so that QuickTest can support your Stingray application. If you select this mode, click **Next** and proceed to step 4 below.

4 If you select **Precompiled Agent**, the following screen is displayed.



- Click **Help** to display information describing how to add support code to your Stingray application project. For more information, see "Using the Precompiled Agent Mode" on page 509.
- Click **Finish** to close the wizard. If you have not already compiled your application with the Stingray Add-in agent files, you must do so before you begin to work with the QuickTest Professional Stingray Add-in. For more information, see "Using the Precompiled Agent Mode" on page 509.

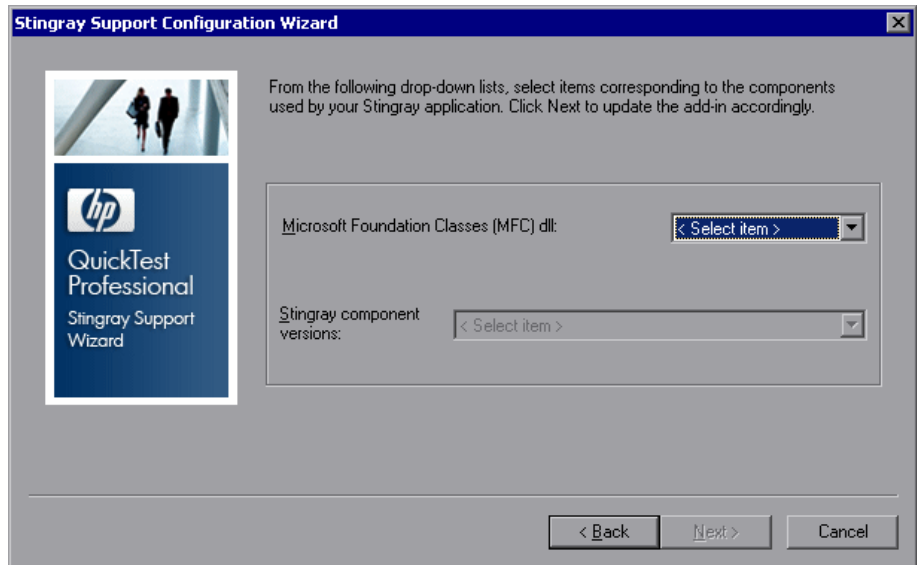
- 5 If you select **Run-time Agent** in the mode selection screen, the following configuration options screen is displayed.



- 6 Select one of the following configuration options:
- ▶ **Automatic configuration.** Instructs the wizard to configure Stingray support automatically according to the detected MFC DLL and the version of Stingray components used in your application.
If you select this option, click **Next** and proceed to step 8 on page 516.
 - ▶ **Manual configuration.** Enables you to configure Stingray support manually by specifying the MFC DLL and Stingray component version used in your application. This is useful, for example, if your application is statically linked to the Stingray libraries.
If you select this option, click **Next** and proceed to step 7 on page 515.

Note: If, at any time, you encounter problems with the Stingray Add-in, you can create a diagnostic log file by selecting the **Create wizard configuration log file** check box. If you contact HP Software Support for assistance, you may be asked to provide this log file for diagnostic purposes.

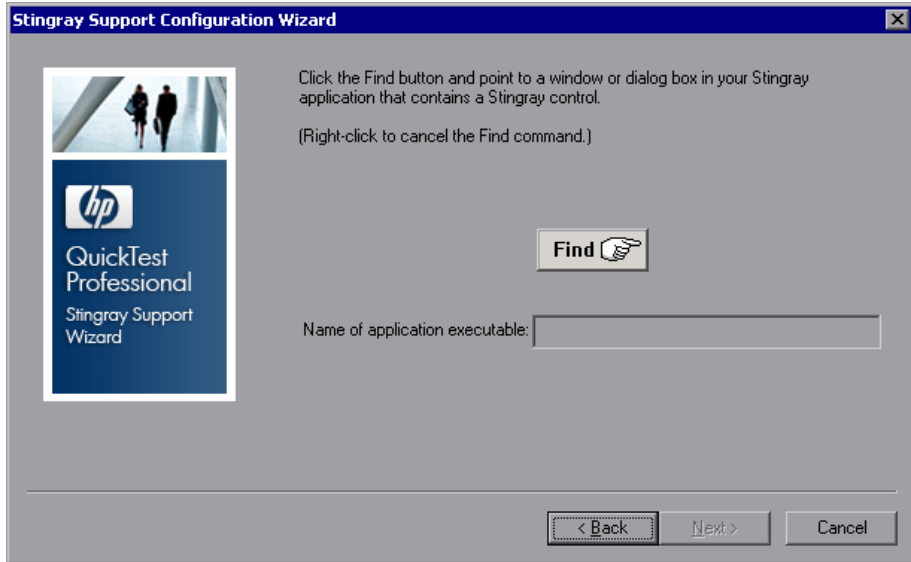
- 7 If you select **Manual configuration**, the following components screen is displayed.



Select a **Microsoft Foundation Classes (MFC) dll** and a **Stingray component version** Objective Grid and Objective Toolkit combination from the drop-down lists.

Proceed to step 13 on page 520.

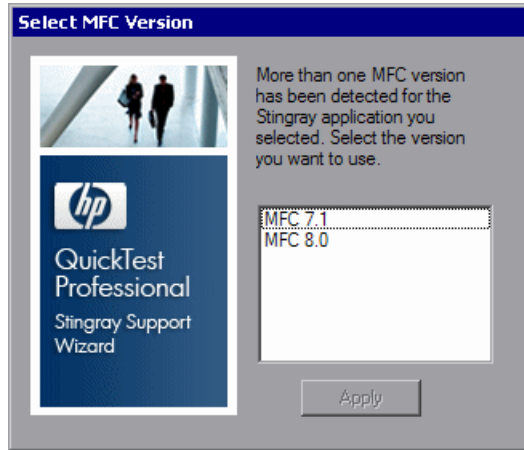
- 8 If you select **Automatic configuration**, the application executable detection screen opens. Click the **Find** button and point to a window or dialog box in your application that contains a Stingray control. QuickTest automatically detects the name of the application executable.



You can right-click at any time to cancel the **Find** command.

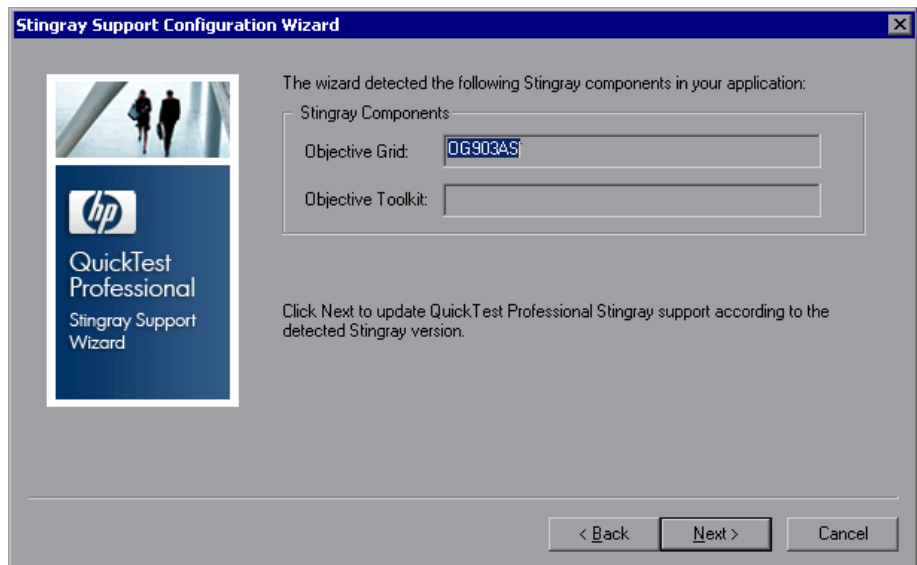
Tip: If you want to bring another window into focus or perform operations such as a right-click or mouseover to display a context menu, you can press and hold the **Ctrl** key. This temporarily disables the **Find** mechanism and enables you to perform regular mouse operations. When the window or dialog box containing the Stingray control is displayed, release the **Ctrl** key. Note that pressing the **Ctrl** key does not enable you to select an application from the Windows task bar, therefore you must make sure that the window you want to access is not minimized.

- 9 If QuickTest detects more than one MFC version for the Stingray application, the following dialog box opens:



Select the relevant version and click **Apply**.

- 10 In the application executable detection screen, click **Next**. The wizard displays the Stingray components it detected in your application process.



If, in the previous screen, you pointed to a non-Stingray application, or to a Stingray application whose components QuickTest could not detect, a warning message displays stating that QuickTest failed to detect the Stingray components in your application.

QuickTest may fail to detect components of a Stingray application for several reasons. For example, the application may be statically linked to Stingray libraries, preventing the wizard from identifying the version of the Stingray libraries. In this case, click **Back** twice and select **Manual configuration** to configure Stingray support manually. For more information, see step 6 on page 514.

Another reason may be that the application is statically linked to the Microsoft Foundation Class (MFC) libraries. In this case, click **Back** three times and select **Precompiled Agent**. For more information, see step 3 on page 513.

In addition, if QuickTest detects a Stingray version that is not supported by the Stingray Add-in, or is slightly different from the officially supported versions, a warning message displays.

If you are working with a Stingray version that is not supported, contact HP Software Support who may be able to provide you with a support agent for your specific version. For more information, see the *HP QuickTest Professional Product Availability Matrix*, available from the Documentation Library Home page or the root folder of the QuickTest Professional DVD.

- 11 Click **Next**. QuickTest configures Stingray support according to the detected Stingray version and displays the following screen.



Note: To test applications created with other Stingray versions, you must run the Stingray Support Configuration Wizard again. To open the wizard, select **Start > Programs > HP QuickTest Professional > Stingray Support Configuration Wizard**. Alternatively, in QuickTest, select **Tools > Options > Stingray** pane and click the **Select Version** button. For more information, see "Considerations for Working with the Stingray Add-in" on page 506.

- 12** If you want the same settings you configured in this procedure to be effective for all users of the computer, select **Apply to all users on this computer**.

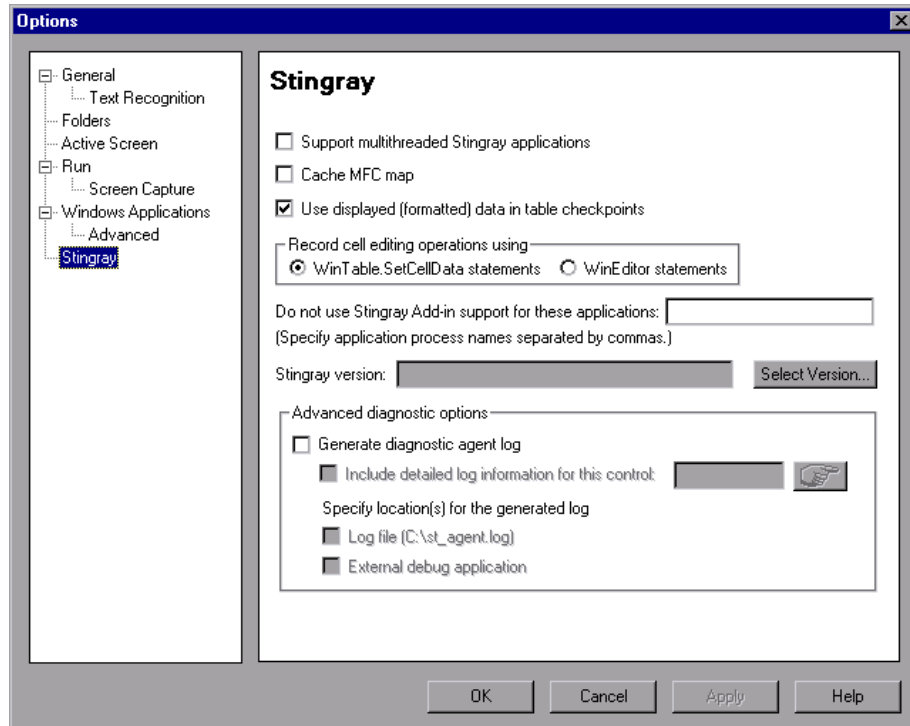
Note: You must have administrator permissions on the computer to configure support for all users. If you do not have administrator permissions, the **Apply to all users on this computer** option is disabled.

- 13** Click **Finish** to close the wizard.

For more information on using QuickTest, see the *HP QuickTest Professional User Guide*.

Configuring Stingray Options

The Stingray pane of the Options dialog box (**Tools > Options > Stingray** node) enables you to configure how QuickTest records and runs tests and components on Stingray Objective Grid and Objective Toolkit objects.



If the Stingray Add-in is configured correctly, you may not need to make any modifications using this pane. However, if you did not run the Stingray Support Configuration Wizard after installing the Stingray Add-in, or if you encounter difficulties when recording and running tests and components on Stingray applications, you can use the options in this pane to fine tune the configuration. For example, you can enable support for multithreaded applications by selecting the relevant option in this pane.

After making changes to options in this pane, you must restart QuickTest before you continue working with this add-in.

User interface elements are described below:

Support multithreaded Stingray applications

Instructs QuickTest to support multithreaded Stingray applications. If you are not sure whether you are working with a multithreaded Stingray application, first try to record and run on your Stingray application without selecting this check box. If you experience difficulties, you can select this check box and try again.

By default, this check box is cleared.

Note: Select this check box only if you are working with a multithreaded application.

Cache MFC map

Instruct QuickTest to use auxiliary caching as a backup for MFC's internal mapping of window handles to Visual C++ objects. If QuickTest cannot identify one or more Stingray controls while recording or running a test or component, you can select this check box to instruct QuickTest to use a cached map instead of using the Stingray application for identification.

By default, this check box is cleared.

Use displayed (formatted) data in table checkpoints

Instructs QuickTest to use the formatted data value in the Stingray grid control. You can use this option when working with table checkpoints (not supported for components). For example, if the actual value of a cell in a Stingray application is formatted to display two digits to the right of the decimal point, QuickTest will use that rounded number instead of the actual number when checking the value during the run session.

By default, this check box is selected.

Record cell editing options using

Instructs QuickTest to record typing operations in a Stingray grid (edit) cell using one of the following options:



- **WinTable.SetCellData statements.** (Default) Uses the SetCellData method to record the final value that you enter in a grid cell. This option results in a single step in your test or component. In most cases, this option makes the step more readable and easier to modify manually
- **WinEditor statements.** QuickTest records each operation that you perform in a Stingray grid edit cell as a separate WinEditor step. For example, operations such as placing the cursor in a specific place in the edit box, typing a single character, or deleting a character may be recorded as individual steps. This can make your test or component less readable and more difficult to modify manually, but this may be useful if you want to test the behavior of specific editing operations.

For example, suppose that during a recording session, you place the cursor in an edit-type cell that already contains the value `abc`. You place the cursor before the `b`, delete the `b` and `c` characters, and then you type `bcde`.

If you are using the **WinTable.SetCellData statements** option, QuickTest records the following in Expert View:

```
Window("GRIDAPP").Window("GridAp1").WinTable("StingrayGrid").SelectCell "#2", "#3"
Window("GRIDAPP").Window("GridAp1").WinTable("StingrayGrid").SetCellData "#2", "#3", "abcde"
```






QuickTest inserts these steps as follows in the Keyword View:

 StingrayGrid	SelectCell	"#2", "#3"	Select the cell in row "#2", column "#3" in the "StingrayGrid" table.
 StingrayGrid	SetCellData	"#2", "#3", "abcde"	Enter "abcde" in the "StingrayGrid" table in row "#2", column "#3".

If you are using the **WinEditor statements** option, QuickTest records the following in Expert View:

```
Window("GRIDAPP").Window("GridAp1").WinTable("StingrayGrid").SelectCell "#2", "#3"
Window("GRIDAPP").Window("GridAp1").WinEditor("Edit").SetCaretPos 0,1
Window("GRIDAPP").Window("GridAp1").WinEditor("Edit").Type micDel
Window("GRIDAPP").Window("GridAp1").WinEditor("Edit").Type micDel
Window("GRIDAPP").Window("GridAp1").WinEditor("Edit_2").Type "bcde"
```

QuickTest inserts these steps as follows in the Keyword View:

 StingrayGrid	SelectCell	"#2", "#3"	Select the cell in row "#2", column "#3" in the "StingrayGrid" table.
 Edit	SetCaretPos	0,1	Move the cursor to position 0, 1 in the "Edit" text area.
 Edit	Type	micDel	Type micDel in the "Edit" text area.
 Edit	Type	micDel	Type micDel in the "Edit" text area.
 Edit_2	Type	"bcde"	Type "bcde" in the "Edit_2" text area.

**Do not use Stingray Add-in support for these applications:
(Specify application process names separated by commas.)**

Instructs QuickTest to treat the applications you specify as non-Stingray applications.

Some open, non-Stingray processes (such as **explorer.exe**) can cause unexpected behavior when recording and running tests and components on Stingray applications. By adding the process names to this edit box, you can help prevent this unexpected behavior.

Notes:

- ▶ In some cases, the executable file you use to open an application is only a launching process, which then opens the actual application process. In these cases, make sure that you specify the name of the actual application process and not the launching process.
 - ▶ When working with tests, this option is relevant only if you selected the **Record and run test on any open Windows-based application** in the Record and Run Settings dialog box (**Automation > Record and Run Settings**). For more information on the options available in the Record and Run Settings dialog box, see the *HP QuickTest Professional User Guide*.
-

Stingray version

Indicates the versions of the Stingray Objective Grid and Stingray Objective Toolkit libraries used for identifying Stingray objects in your application (read-only).

Select Version

Opens the Stingray Support Configuration Wizard, which enables you to select the combination of Objective Grid and Objective Toolkit versions with which you want to work.

For more information, see "Running the Stingray Support Configuration Wizard" on page 512.

Generate diagnostic agent log

Instructs QuickTest to generate a diagnostic agent log file. You can use this option if you encounter problems with the Stingray Add-in, for example, if QuickTest does not recognize a Stingray grid control while recording. HP Software Support may ask you to generate this log and send it together with your service request.

When you select this check box, the following options are enabled:

- **Include detailed log information for this control**
- **Log file (C:\st_agent.log)**
- **External debug application**

Note: If you select this check box, you must specify the location for the generated log. For more information, see "Specify location(s) for the generated log" on page 526.

Include detailed log information for this control

Instructs QuickTest to include detailed information in the generated log for a specific Stingray control, in addition to the general QuickTest/agent communication log information. For example, you may want to generate additional log details for a specific Stingray grid.

To select the object for which you want to generate detailed log information:

Click the pointing hand and then click the relevant Stingray control. The selected object's window handle is displayed in the edit box.

Note: This option is available only when the **Generate diagnostic agent log** check box is selected.

Specify location(s) for the generated log

Instructs QuickTest to generate the log to the selected locations. You can select one or both of the following options:

- ▶ **Log file (C:\st_agent.log).** Saves the diagnostic log to the **st_agent.log** text file on your **C:** drive.
- ▶ **External debug application.** Exports the diagnostic log data to an external debug application, such as the freeware application, DebugView, or Microsoft VisualStudio.

Note: These options are available only when the **Generate diagnostic agent log** check box is selected.

Troubleshooting and Limitations - Stingray Add-in

This section describes troubleshooting and limitations for the Stingray Add-in.

General

- ▶ Applying Stingray Support Configuration settings to all users on the computer has no effect on users that have opened QuickTest at least once.

Workaround: Apply Stingray Support Configuration settings separately for each user that has opened QuickTest at least once.

- ▶ QuickTest does not support both Unicode and non-Unicode in the same application when the Stingray Add-in is loaded.

Creating and Running Tests and Components

- ▶ If your Stingray application was built using the precompiled agent mode and you have used the Stingray Support Configuration Wizard at least once to set a Stingray run-time agent, then recording, learning, or running steps on the application may fail.

- ▶ By default, only singlethreaded Stingray applications are supported.

To provide support for multithreaded applications, in QuickTest, select **Tools > Options > Stingray** node. Select the **Support multithreaded Stingray applications** check box and click **OK**. Close and restart QuickTest.

For more information, see the *HP QuickTest Professional Add-ins Guide*.

- ▶ The Stingray Add-in does not support Objective Edit or Objective Chart controls.
- ▶ The **ExpandAll** method is not supported for Stingray tree controls.

- ▶ Sometimes, the MFC internal map that correlates a window handle of a control with a Visual C++ object may not contain an entry for all Stingray controls. In such cases, the Stingray Add-in may fail to recognize certain Stingray controls because it relies on this map when retrieving information from the application.

Workaround: The Stingray Add-in contains an auxiliary mechanism that serves as a fallback for the lack of MFC map entries in the situation described above. To activate this mechanism, in QuickTest, select **Tools > Options > Stingray** node. Select the **Cache MFC map** check box and click **OK**. Close and restart QuickTest.

Note: This mechanism is not activated by default because it imposes some performance overhead.

- ▶ When working with nested tab controls, you may need to manually modify the corresponding entries in the object repository to enable unique identification. For example, you may need to add an ordinal identifier to the existing description.
- ▶ By default, edit boxes, check boxes, and drop-down (combo) lists are supported when recording on a Stingray grid. Other types of controls embedded in Stingray grids may be supported partially or may not be supported at all.

Note: The CGXTabbedComboBox control and the CGXCheckBoxEx control type are not supported during recording.

Workaround: To work with controls other than the supported ones, manually add **SetCellData** statements to your test or component (instead of recording user actions inside cells).

- ▶ **GetCellData** and **SetCellData** methods are limited to 3000 characters.

- ▶ By default, only the following grid classes are supported:
 - ▶ CGXBrowserView
 - ▶ CGXBrowserWnd
 - ▶ CGXGridWnd
 - ▶ CGXGridView
 - ▶ CGXGridHandleView

- ▶ When Stingray tree control items have tooltips, recording the selection of an item by clicking its label may fail.

Workaround: Select the requested item by performing a click on the item's icon.

Part IV

The Terminal Emulator Add-in

32

Using the Terminal Emulator Add-in

You can use QuickTest Professional with the Terminal Emulator Add-in to test terminal emulator applications that support HLLAPI (High Level Language Application Programming Interface) as well as those that do not, for example, emulator sessions configured to work with the VT100 protocol (using the **Text-only** option). HLLAPI allows a PC application to communicate with a mainframe application with extended capabilities.

If your emulator supports HLLAPI, QuickTest recognizes the screen and field objects in your emulator screen. If your emulator does not support HLLAPI, or you have configured QuickTest in **Text-only** mode, QuickTest records operations in terms of the text as it appears in the rows and columns of your emulator screen.

The QuickTest Professional Terminal Emulator Add-in includes preconfigured settings for several terminal emulators. The Terminal Emulator Add-in also enables you to configure the settings for most other terminal emulators using the Terminal Emulator Configuration Wizard.

For details on supported emulators, see the **Terminal Emulator Add-in** section of the *HP QuickTest Professional Product Availability Matrix*, available from the Documentation Library Home page or the root folder of the QuickTest Professional DVD.

To configure your HLLAPI emulator to work with QuickTest, see "Setting Your HLLAPI Terminal Emulator to Work with QuickTest" on page 552.

The following table summarizes basic information about the Terminal Emulator Add-in and how it relates to some commonly-used aspects of QuickTest.

General Information	
Add-in Type	This is a Windows-based add-in. Much of its functionality is the same as other Windows-based add-ins. See "Testing Windows-Based Applications" on page 87.
Checkpoints and Output Values	<ul style="list-style-type: none"> ▶ See the sections describing checkpoints and output values in the <i>HP QuickTest Professional User Guide</i> (for tests), and the <i>HP QuickTest Professional for Business Process Testing User Guide</i> (for components). ▶ See "Enhancing Your Terminal Emulator Tests and Components" on page 585. ▶ See "Supported Checkpoints and Output Values Per Add-in" on page 769.
Prerequisites	
Opening Your Application	You can open your Terminal Emulator Add-in application before or after opening QuickTest and creating a test.
Add-in Dependencies	None
Other	<ul style="list-style-type: none"> ▶ Before using the Terminal Emulator Add-in for the first time, you must enable QuickTest to identify your terminal emulator. See "Using the Terminal Emulator Configuration Wizard" on page 536. ▶ You must configure your terminal emulator settings to work with QuickTest. See "Setting Your HLLAPI Terminal Emulator to Work with QuickTest" on page 552.

Setting Preferences	
Options Dialog Box	<ul style="list-style-type: none"> ➤ Use the Terminal Emulator pane. (Tools > Options > Terminal Emulator node) See "Modifying Your Terminal Emulator Settings" on page 562. ➤ Use the Windows Applications pane. (Tools > Options > Windows Applications node) See "The Options Dialog Box: Windows Applications Pane" on page 103.
Custom Active Screen Capture Settings Dialog Box (tests only)	<p>Use the Terminal Emulator section in the dialog box. (Tools > Options > Active Screen node > Custom Level)</p> <p>See the section on the Custom Active Screen Capture Settings dialog box in the <i>HP QuickTest Professional User Guide</i>.</p>
Application Area Settings Dialog Box (components only)	<p>Use the Applications pane. (File > Settings > Application node)</p> <p>See the section on defining Application Settings for your application area in the <i>HP QuickTest Professional for Business Process Testing User Guide</i>.</p>

This chapter includes:

- Using the Terminal Emulator Configuration Wizard on page 536
- Copying Existing Configurations on page 550
- Setting Your HLLAPI Terminal Emulator to Work with QuickTest on page 552

Using the Terminal Emulator Configuration Wizard

The Terminal Emulator Configuration Wizard guides you through the process of configuring the settings QuickTest needs to identify your terminal emulator. If your emulator is not in the list of preconfigured settings to select, you can define the way QuickTest identifies your emulator.

Note: If your terminal emulator settings for QuickTest have been configured on another computer, you can copy an existing configuration file onto your computer, instead of running the wizard. For more information, see "Copying Existing Configurations" on page 550.

The Terminal Emulator Configuration Wizard opens after you install QuickTest, if you selected **Run the Terminal Emulator wizard** in the Additional Installation Requirements dialog box. You can also run the wizard at any time by selecting **Tools > Options > Terminal Emulator** node from the QuickTest menu, and then clicking **Open Wizard** in the Terminal Emulator pane.

Note: The Terminal Emulator pane is available in the Options dialog box only when the Terminal Emulator add-in is installed and loaded.

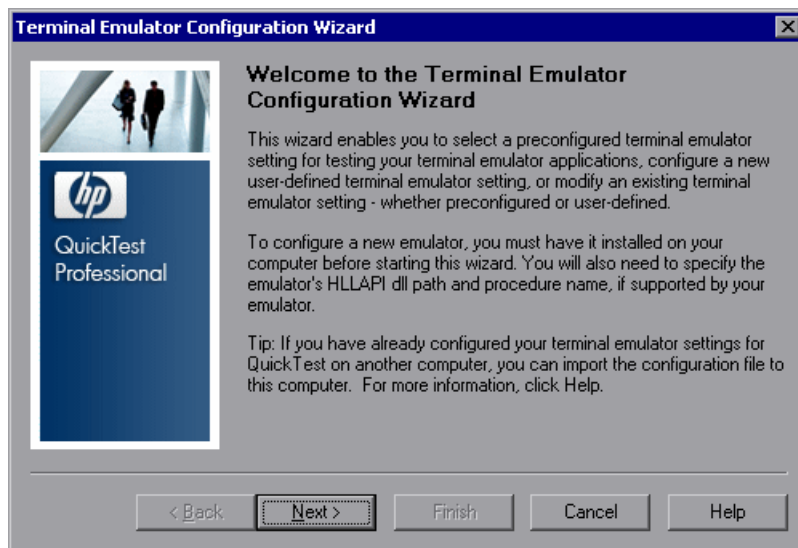
Once you are finished running the wizard, the terminal emulator you selected is set as the default emulator when you open QuickTest with the Terminal Emulator Add-in loaded. You can check your configurations by clicking the **Validate** button in Terminal Emulator pane of the Options dialog box. A description of any detected problem is displayed in the pane, as well as a link to a specific troubleshooting Help page. For more information, see "Validating your Terminal Emulator Configuration" on page 564.

You can also use the wizard to select a different emulator for use with your tests or components. For more information, see "Modifying Your Terminal Emulator Settings" on page 562.

Note: If you want to use a terminal emulator that supports HLLAPI, make sure that you close any application that is currently using the HLLAPI .dll file before you begin using the wizard. Otherwise, the wizard will not be able to connect to your terminal emulator.

Terminal Emulator Configuration Wizard Welcome Screen

The Welcome screen provides general information about the different options in the Terminal Emulator Configuration Wizard.

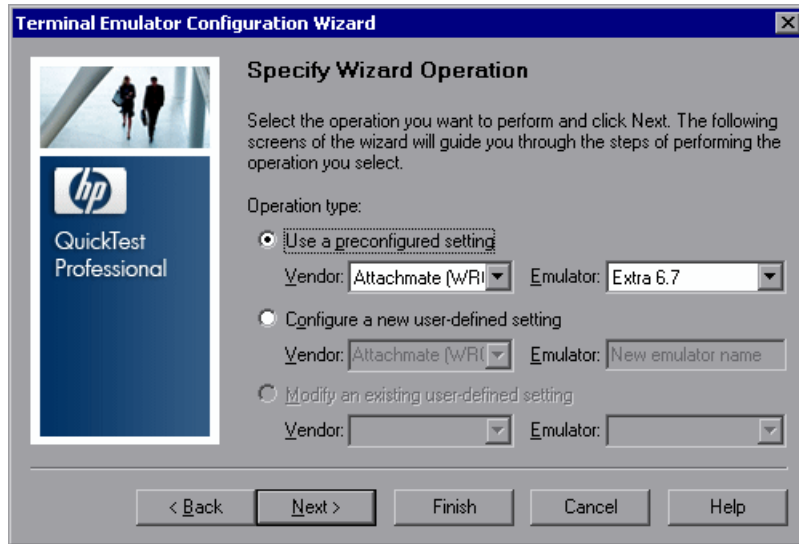


For information on copying an existing configuration, see "Copying Existing Configurations" on page 550.

When you click **Next**, the Specify Wizard Operation screen opens.

Specify Wizard Operation Screen

In the Specify Wizard Operation screen, you determine which operation you want the wizard to perform.



Select from one of the following options:

- ▶ **Use a preconfigured setting** by selecting one of the vendor/emulator settings that are supplied with your Terminal Emulator Add-in.
- ▶ **Configure a new user-defined setting** by supplying the details of your vendor and emulator.

After completing the wizard, the vendor and emulator names that you define here appear in the list of vendor/emulator combinations in the QuickTest **Tools > Options > Terminal Emulator** pane.

- ▶ **Modify an existing user-defined setting** that has been previously configured with the Terminal Emulator Configuration Wizard.

You can also modify a setting that has been copied to your computer registry, as described in "Copying Existing Configurations" on page 550.

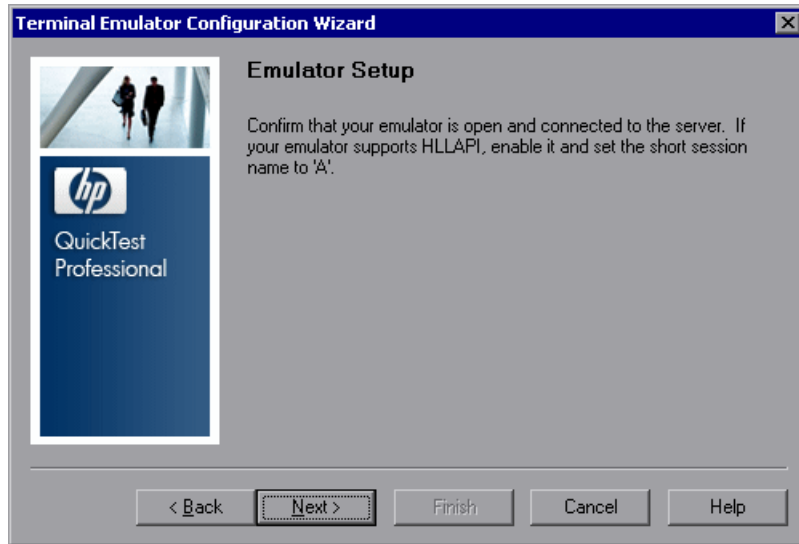
Click **Next**. If you chose to use a preconfigured setting, the Configure Emulator Screen Settings screen opens. For more information, see "Configure Emulator Screen Settings Screen" on page 547. If you chose to configure a new user-defined setting, or modify an existing user-defined setting, the Emulator Setup screen opens.

Note: If you chose a preconfigured setting, you can click **Finish** instead of **Next** to begin working with QuickTest to test the emulator you selected. However, if you are testing a Web-based emulator, or if QuickTest is not recording or recognizing objects as expected, it is recommended to click **Next** and define the emulator screen settings. Note that the emulator screen settings do not affect run sessions; they affect only recording and other object operations (for example, inserting checkpoints, using the Object Spy, and so on.)

For more information on using your emulator with QuickTest, see "Testing Terminal Emulator Applications" on page 559.

Emulator Setup Screen

The Emulator Setup screen instructs you to open your terminal emulator and connect it to the server.

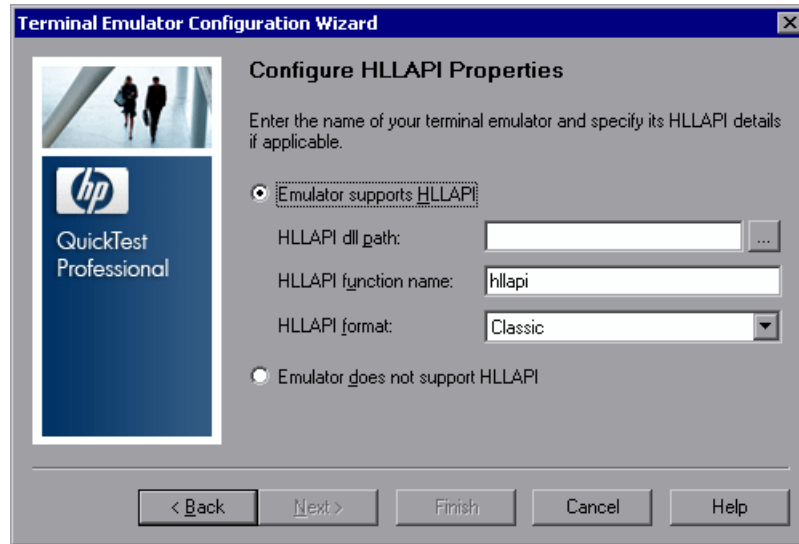


If your emulator supports HLLAPI, enable it and set the emulator short session name to the uppercase letter **A**. For more information, see "Setting Your HLLAPI Terminal Emulator to Work with QuickTest" on page 552.

When you click **Next**, the Configure HLLAPI Properties screen opens.

Configure HLLAPI Properties Screen

The Configure HLLAPI Properties screen allows you to specify whether your terminal emulator supports HLLAPI.



If your emulator does not support HLLAPI, select **Emulator does not support HLLAPI**. When you click **Next**, the Configure Emulator Classes screen opens. For more information, see "Configure Emulator Classes Screen" on page 545.

If your emulator supports HLLAPI, select **Emulator supports HLLAPI** and supply the information described below. If you are not sure what values to enter, consult your terminal emulator documentation or contact the vendor for your terminal emulator.

The table below lists the DLL and function names used by the supported terminal emulators.

Emulator Name	DLL Name	HLLAPI Function Name
Attachmate EXTRA! and Attachmate myEXTRA! Terminal Viewer	ehlapi32.dll	hllapi
Attachmate INFOConnect	ihlapi32.dll	WinHLLAPI

Emulator Name	DLL Name	HLLAPI Function Name
Hummingbird HostExplorer	ehllap32.dll	HLLAPI32
IBM Personal Communications (PCOM) and IBM WebSphere Host On-Demand	pcshll32.dll	hllapi
NetManage RUMBA and NetManage RUMBA Web-To-Host	ehllapi32.dll	hllapi
PuTTY	Not applicable	Not applicable
Seagull BlueZone	WHLAPI32.dll	hllapi
WRQ Reflection	hllapi32.dll	hllapi
Zephyr (PC/Web to Host)	PassHll.dll	hllapi

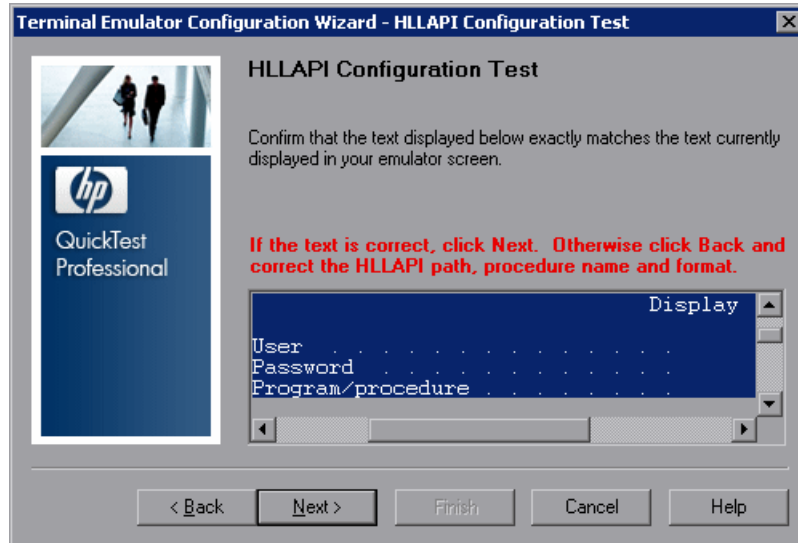
- ▶ **HLLAPI dll path.** QuickTest uses the HLLAPI dynamic-link library file specified for the selected emulator to connect to the emulator and to retrieve data concerning its current status. This file usually resides in the terminal emulator installation folder. You can click the browse button to search for the path.
- ▶ **HLLAPI function name.** The HLLAPI DLL for the selected emulator uses this function as the entry point for all HLLAPI calls.
- ▶ **HLLAPI format.** This is the format by which QuickTest attempts to identify your emulator screen. If you are working with VT protocols, select the **Text-only** option. Otherwise, it is recommended to select **Auto-detect**.

If, in the next screen, QuickTest is unable to capture the text from your terminal emulator, you may need to return to this screen and change this selection to **Classic**, **Extended**, or **Text-only**. You should also confirm the accuracy of the properties you entered in the screen.

When you click **Next**, the HLLAPI Configuration Test screen opens.

HLLAPI Configuration Test Screen

If you selected **Emulator supports HLLAPI** in the Configure HLLAPI Properties screen, the HLLAPI Configuration Test screen displays a screen capture test. This test enables you to determine whether QuickTest has been able to accurately identify your terminal emulator screen.



Check that the screen capture test is correct for your currently selected terminal emulator, and that all the text has been correctly identified and displayed.

If the wizard displays the emulator screen and the text correctly, click **Next**. The Configure Emulator Classes screen opens. For more information, see "Configure Emulator Classes Screen" on page 545.

Troubleshooting the HLLAPI Properties Configuration

If the wizard does not display the text correctly or if the HLLAPI configuration test fails, do the following:

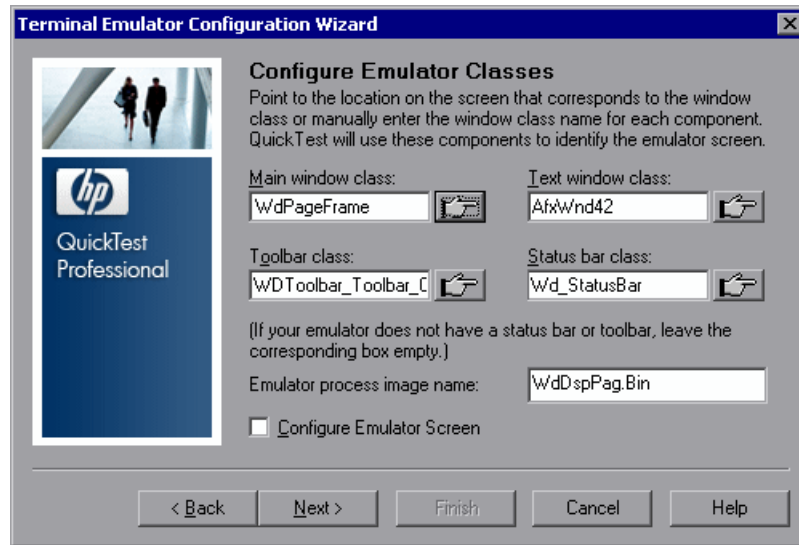
- 1** Click **Back**. Before repeating the test:
 - ▶ Make sure your emulator is connected to the host and the short name for the session is set to the uppercase letter **A**. For more information, see "Setting Your HLLAPI Terminal Emulator to Work with QuickTest" on page 552.
 - ▶ Check that the settings you entered in the Configure HLLAPI Properties screen are accurate (DLL path, procedure, format). For more information, see "Configure HLLAPI Properties Screen" on page 541.
 - ▶ Make sure that the HLLAPI .dll file you specified in the DLL path is not in use by QuickTest or another application. If the .dll file is currently in use by another application, click **Cancel** to close the wizard, close the application using the DLL, and restart the wizard. If the .dll file is currently in use by QuickTest, choose a different emulator and create a new test, then reopen the wizard and modify the original configuration as required.
- 2** If the display is still not correct, click **Back** and in the Configure HLLAPI Properties screen, change the **HLLAPI format** to **Text-only**. You should also use the **Text-only** option if you are working with a VT protocol, or if you have begun working in QuickTest and encountered problems with recording and running tests or components. For more information, see "Configure HLLAPI Properties Screen" on page 541.
- 3** If all the above tips have failed to solve the problem, click **Back** and in the Configure HLLAPI Properties screen, select **Emulator does not support HLLAPI**. For more information, see "Configure HLLAPI Properties Screen" on page 541.

Tip: If you have a good understanding of your emulator, you may be able to solve any problems you experience by adjusting the configuration settings. For more information, see Chapter 35, "Adjusting Your Terminal Emulator Configuration Settings."

Configure Emulator Classes Screen

QuickTest distinguishes between the window of the terminal emulator and the screens in the host application.

Entering the class information in the Configure Emulator Classes screen enables QuickTest to locate the information on the emulator screen by identifying the components of the terminal emulator window.



Identifying Emulator Components

To identify the components of the emulator, click the pointing hand and then click the corresponding object on your terminal emulator window:

- ▶ **Main window class.** Click the uppermost title bar of the main emulator window.
- ▶ **Text window class.** Click on the text within the emulator screen.
- ▶ **Toolbar class.** Click the terminal emulator's toolbar (if applicable).
- ▶ **Status bar class.** Click the lowest status bar of the main emulator window (if applicable).

Emulator Process Image Name

After the emulator main window class has been identified, the wizard detects the process name for the emulator and displays it in the **Emulator process image name** box. QuickTest uses this process name to identify the correct process for this terminal emulator when recording and running tests or components.

Check that the displayed process name is correct for this emulator.

Tip: You can view the image names of the currently loaded processes in the **Image Name** column of the Windows Task Manager Processes tab.

Configure Emulator Screen

If your emulator supports HLLAPI, the Terminal Emulator Add-in automatically retrieves the configuration settings for the emulator screen. These are normally correct. If you want to review or change these settings, select the **Configure Emulator Screen** check box.

If your emulator does not support HLLAPI, you need to configure your emulator screen correctly for use with QuickTest. Make sure that the **Configure Emulator Screen** check box is selected.

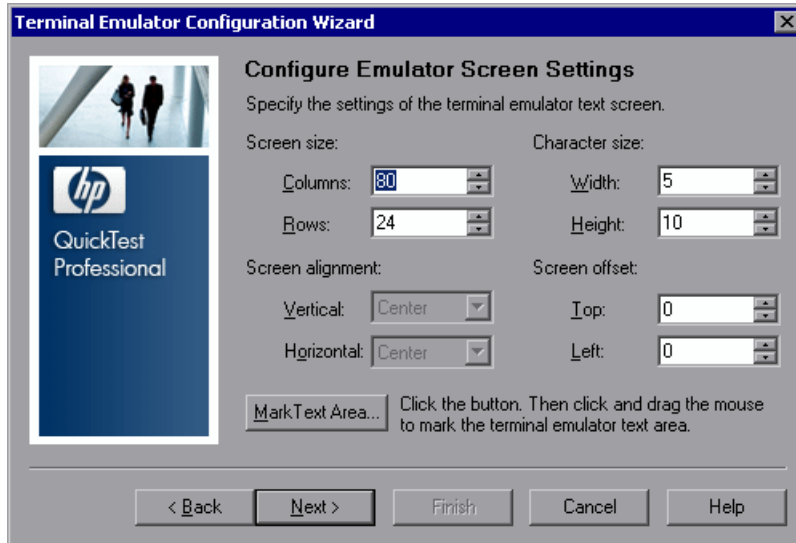
If you select the **Configure Emulator Screen** check box and click **Next**, the Configure Emulator Screen Settings screen opens. For more information, see "Configure Emulator Screen Settings Screen" on page 547.

If you choose not to configure the emulator screen settings, the Terminal Emulator Add-in automatically adjusts the screen size and alignment, using a proprietary algorithm with the settings retrieved for the emulator.

If you do not select the **Configure Emulator Screen** check box, when you click **Next**, the Completing the Terminal Emulator Configuration Wizard screen opens. For more information, see "Completing the Terminal Emulator Configuration Wizard Screen" on page 549.

Configure Emulator Screen Settings Screen

If you selected a preconfigured setting, or if you selected the **Configure Emulator Screen** check box in the Configure Emulator Classes screen, your emulator screen is displayed with a red grid overlay, and the Configure Emulator Screen Settings screen opens.



Change the settings of the emulator screen to correspond to the required settings for your emulator. The character size, column, and row details for your terminal emulator are generally available from your emulator's connection configuration menu.

As you change the settings for the emulator screen, the grid automatically adjusts to show the new settings.

Marking the Text Area

You can click **Mark Text Area** to define the dimensions of the terminal emulator text area on your emulator screen. When you click **Mark Text Area**, the wizard is minimized and the cursor becomes a crosshairs pointer. Drag the pointer on your emulator screen to define the text area.

After you mark the text area on your emulator screen you can fine-tune your settings by adjusting the text screen settings.

Adjusting the Text Screen Settings

You can specify your emulator's screen size in terms of:

- ▶ **Number of columns and rows.** Specify the number of columns and rows in your emulator screen.
- ▶ **Character size.** Select the width and height of your emulator's characters to fit correctly in the defined emulator screen.

You can specify how the text on your emulator's screen should be aligned in relation to the emulator window when the window size changes. The effect of these settings depends on the behavior of your emulator:

- ▶ **Screen alignment.** Select the vertical alignment (**Top** or **Center**) and the horizontal alignment (**Left** or **Center**) of the emulator screen inside the window. These options are already optimized for preconfigured emulator settings, and cannot be modified.

Tip: The screen alignment settings determine how QuickTest identifies the information on your emulator screen. If you are having trouble recording and running tests or components (for example, the `ClickPosition` method is not accurately determining the coordinates), try changing the **Screen alignment** settings.

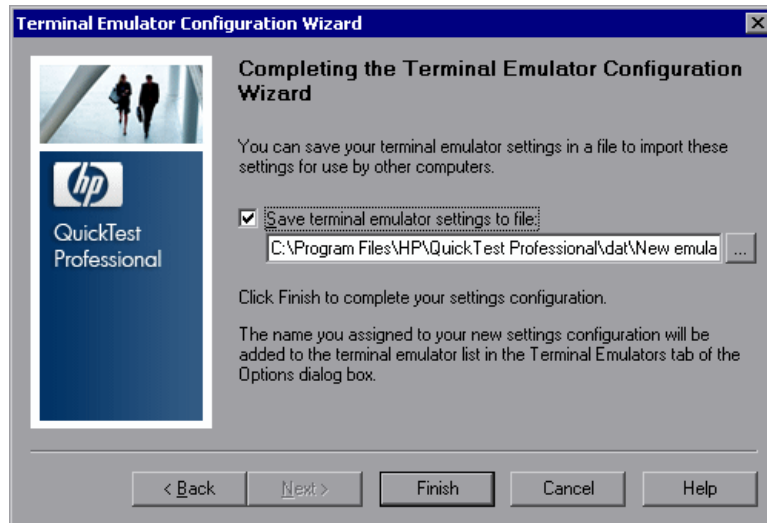
- ▶ **Screen offset.** Select the top and left offset for the text on your emulator screen in relation to the emulator window. For example, if you know that your emulator always reserves one blank row at the top of the screen, set the offset to **1**.

Note: You can open the wizard again at any time to adjust these settings, by clicking **Open Wizard** in the QuickTest **Tools > Options > Terminal Emulator** pane. For more information, see "Modifying Your Terminal Emulator Settings" on page 562.

When you click **Next**, the Completing the Terminal Emulator Configuration Wizard screen opens.

Completing the Terminal Emulator Configuration Wizard Screen

When you have finished configuring the settings for your terminal emulator, the Completing the Terminal Emulator Configuration Wizard screen opens.



To save your settings to a separate registry file, select **Save terminal emulator settings to file** and specify a location.

Note: It is recommended that you save the settings you have just configured to a separate registry file. This allows you to restore this exact configuration if you later change your configuration settings. For more information, see "Modifying Your Terminal Emulator Settings" on page 562.

If you save your settings to a registry file, other users will be able to copy and use your terminal emulator configuration. For more information, see "Copying Existing Configurations" on page 550.

When you click **Finish**, the name you assigned to your new configuration settings is added to the list of available terminal emulators in the Terminal Emulator pane of the Options dialog box.

Note: If you are using the wizard from the Terminal Emulator pane of the Options dialog box, any changes you make are not applied to the currently open test or component. To apply your changes, close your test or component and reopen it.

Copying Existing Configurations

After one user has configured the QuickTest settings for a specific emulator using the wizard, other users can copy the configuration to their computer.

Note: Before the configuration can be copied, it must be saved to a registry file, using the **Save terminal emulator settings to file** option in the wizard's final screen. For more information, see "Completing the Terminal Emulator Configuration Wizard Screen" on page 549.

If the settings for your terminal emulator have been configured and saved to a file on another computer (or on a network drive), you can copy this file to your computer, instead of running the wizard and defining the settings yourself. Before you copy the saved configuration, make sure you know the vendor name and the emulator name assigned to the configuration, and the exact name and location of the file. The file has a **.reg** extension.

After you have copied a configuration file from another location, the emulator name assigned to this configuration is added to the list of available terminal emulators for your QuickTest installation.

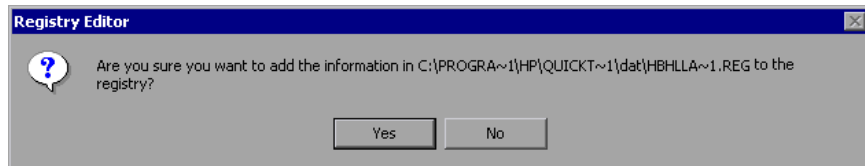
To copy an existing configuration file onto your computer:

- 1 Locate the registry file containing the configuration settings for your emulator. The file has a **.reg** extension.
- 2 Copy the file to the <QuickTest installation folder>\dat folder on your computer.

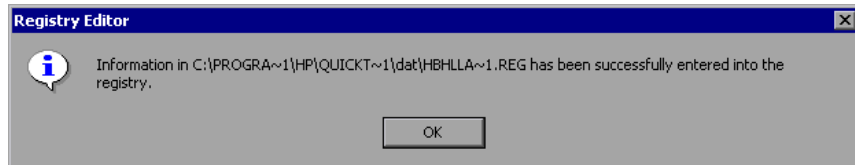
The path for the **dat** folder in a typical installation is:

C:\Program Files\HP\QuickTest Professional\dat.

- 3 Double-click the registry file to open the Registry Editor window.



- 4 Click **Yes** to add the information into the registry. A message opens confirming that the information has been copied into the registry.



- 5 Click **OK**. The emulator name assigned to this configuration is added to the list of available terminal emulators for your QuickTest installation.

After you have opened QuickTest with the Terminal Emulator Add-in loaded, you can select the new emulator name from the list in the **Tools > Options > Terminal Emulator** pane, and set it as your default emulator. You can also open the wizard to modify the emulator settings. For more information, see "Modifying Your Terminal Emulator Settings" on page 562.

Note: If you copy a configuration file after starting QuickTest, you need to close and reopen QuickTest to see the updated list of available emulators.

Setting Your HLLAPI Terminal Emulator to Work with QuickTest

If you are working with an emulator that supports HLLAPI, you must do the following to enable testing on your terminal emulator application:

- Connect your emulator to the host before running the Terminal Emulator Configuration Wizard and before recording each test or component.
- Assign the uppercase letter **A** as the short name for the current emulator session.

Note: You may need to restart the emulator after changing these settings.

The sections below describe how to configure the following emulators to work with the Terminal Emulator Add-in:

- "Attachmate EXTRA!" on page 553
- "Attachmate myEXTRA! Terminal Viewer" on page 553
- "Attachmate INFOConnect" on page 554
- "Hummingbird HostExplorer" on page 554
- "IBM Personal Communications (PCOM)" on page 555
- "IBM WebSphere Host On-Demand" on page 555
- "NetManange RUMBA" on page 556
- "NetManage RUMBA Web-to-Host" on page 556
- "Seagull BlueZone" on page 557
- "WRQ Reflection" on page 557
- "Zephyr Passport" on page 557

Note: For more information on supported emulator versions and protocols, see the *HP QuickTest Professional Product Availability Matrix*, available from the Documentation Library Home page or the root folder of the QuickTest Professional DVD.

Attachmate EXTRA!

To connect your EXTRA! terminal emulator to QuickTest:

- 1 Load EXTRA!.
- 2 In EXTRA!, select **Options > Global Preferences**. The Global Preferences dialog box opens.
- 3 Click the **Advanced** tab.
- 4 In the HLLAPI shortname list, select the uppercase letter **A** as the **Short Name**.
- 5 Click the browse button, browse to and select your session profile, and click **OK**.
- 6 Save the profile.

Tip: It is recommended to save the profile before you start testing with QuickTest Professional. This enables you to configure the terminal emulator once and then reuse the saved settings.

Attachmate myEXTRA! Terminal Viewer

To connect your myEXTRA! terminal viewer to QuickTest:

- 1 Open the myEXTRA! Management and Control Services window.
- 2 In the Management and Control Services window, select **Products > Terminal Viewers**. The Terminal Viewers tree is displayed in the left pane.
- 3 In the Terminal Viewers tree, select the required terminal.

- 4 In the right pane, select the required session and click **Properties**.
- 5 In the Properties pane, click **Configure** to configure the connection.
- 6 In the **General** tab of the Configure pane, select the **Support HLLAPI** check box and set the session name to **A**.
- 7 Save the session.

Note: If this is the first time that you are connecting to a myEXTRA! terminal viewer, you must install the HLLAPI DLL. Click **Preferences** and then click the **Install HLLAPI Client Components** link.

Attachmate INFOConnect

To connect your INFOConnect terminal emulator to QuickTest:

- 1 Select **Options > Global Preferences** from the main menu.
- 1 Select the **Advanced** tab.
- 2 Select **A** as the session short name.
- 3 To associate the session short name (A), with your session, click **Browse** and locate your session profile in the file system.
- 4 Click **OK**.

Hummingbird HostExplorer

To connect your HostExplorer terminal emulator to QuickTest:

- 1 Load HostExplorer.
- 2 From the HostExplorer main menu, select **File > Save Session Profile**.
- 3 The Save Profile dialog box opens. Set the **HLLAPI Short Name** to the uppercase letter **A**.
- 4 From the main menu, select **Options > API Settings**.
- 5 The API Global Settings dialog box opens. Check the **Update screen after PS update** and **Auto sync** options.

6 Click **OK**.

Alternatively:

- 1** Load HostExplorer.
- 2** Open a saved session.
- 3** Select **Options > Edit Session Profile**.
- 4** Select **Terminal > API** in the categories tree.
- 5** Select **A** as the session short name and click **OK**.
- 6** Save the session profile.

IBM Personal Communications (PCOM)

The preconfigured settings enable QuickTest to work with IBM PCOM terminal emulators.

IBM WebSphere Host On-Demand

To connect your WebSphere Host On-Demand terminal emulator to QuickTest:

- 1** Download the WebSphere Host On-Demand EHLLAPI Enablement Tool from the IBM Web site.
- 2** Follow the installation instructions in the *WebSphere Host On-Demand EHLLAPI Enablement Tool Readme* file.
- 3** To be able to record on the WebSphere Host On-Demand terminal emulator, define the session options as follows:
 - Click **Configure** and select **Properties** from the list. Then select **Preferences > Start Options** and set Auto-Start HLLAPI Enabler to **Yes**.
 - Set the **Start In Separate Window** option to **Yes**.
 - Set the **Alternate Terminal** option to **Disable**.

The server and client should not be installed on a computer on which another terminal emulator is installed.

NetManage RUMBA

To connect your RUMBA terminal emulator to QuickTest:

- 1 Load RUMBA.
- 2 In RUMBA, select **Options > API**. The API Options dialog box opens.
- 3 Click the **Identification** tab.
- 4 In the **Session Short Name** field, type the uppercase letter **A**.
- 5 Click **OK**.
- 6 Save the profile.

Tip: It is recommended to save the profile before you start testing with QuickTest Professional. This enables you to configure the terminal emulator once and then reuse the saved settings.

NetManage RUMBA Web-to-Host

To connect your RUMBA Web-to-Host terminal emulator to QuickTest:

- 1 Open the RUMBA Web-to-Host Session Configuration Manager and open a session.
- 2 In addition to your standard configuration steps in the Configuration Manager:
 - a Select **Pro client** from the **Implementation** drop-down list.
 - b Click **HLLAPI Configuration** and select **A** from the **Session Short Name** drop-down list.
- 3 Save the profile.

Notes:

- ▶ Only Mainframe Display is supported for the Java Client.
 - ▶ Only Replay is supported for both Java client and Pro client.
-

Seagull BlueZone

To connect your BlueZone terminal emulator to QuickTest:

- 1** Load BlueZone.
- 2** In BlueZone, select **Options > API**. The API Properties dialog box opens.
- 3** Click the **Options** tab.
- 4** In the **Short Name Session Identifier** field, type the uppercase letter **A**.
- 5** Click **OK**.
- 6** Save the session.

WRQ Reflection

To connect your Reflection terminal emulator to QuickTest:

- 1** Open a new or existing session.
- 2** Select **Setup > Terminal**.
- 3** In the **Short Name** field, type the uppercase letter **A**.
- 4** Click **OK**.

Zephyr Passport

To connect your Zephyr Passport terminal emulator to QuickTest:

- 1** Open a new or existing session.
- 2** Check that the session shortname **(A) Passport.zws** appears in the window title bar.

33

Testing Terminal Emulator Applications

You can use the QuickTest Professional Terminal Emulator Add-in to test applications running on most terminal emulators. The QuickTest Professional Terminal Emulator Add-in recognizes your terminal emulator and records and runs the operations you perform on the screens and fields of the running application.

Before you record or run a test or component on a terminal emulator application, you must connect your emulator to the host and ensure that the emulator is configured properly. For more information, see "Setting Your HLLAPI Terminal Emulator to Work with QuickTest" on page 552.

Note: You can record on only one terminal emulator session at a time. Multiple open sessions may cause problems with recording and running tests or components.

This chapter explains how to use QuickTest to record and run tests or components on terminal emulator applications. For more information on working with QuickTest, see the *HP QuickTest Professional User Guide*, and the *HP QuickTest Professional for Business Process Testing User Guide*.

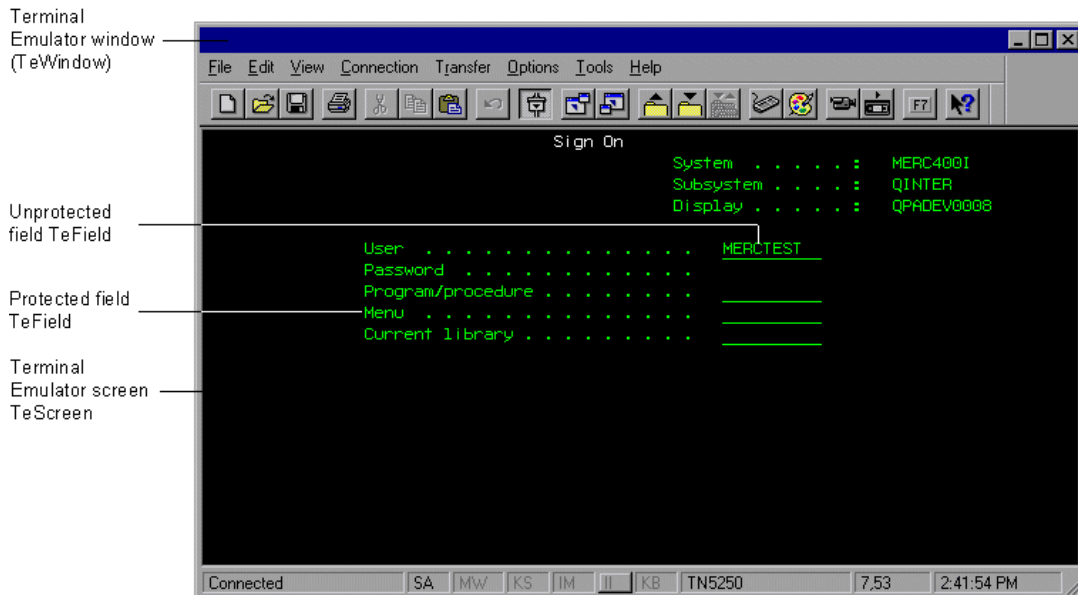
This chapter includes:

- About Testing Terminal Emulator Applications on page 561
- Modifying Your Terminal Emulator Settings on page 562
- Validating your Terminal Emulator Configuration on page 564
- Understanding the Test Object Model on page 568
- Identifying Test Object Classes for Terminal Emulators on page 569
- Understanding Terminal Emulator Recovery Scenarios on page 573
- Recording Tests and Components on Terminal Emulator Applications on page 574
- Troubleshooting and Limitations - Terminal Emulator on page 576

About Testing Terminal Emulator Applications

QuickTest distinguishes between the window of the terminal emulator and the screens in the host application. The terminal emulator window consists of the frame, menus, toolbar, and status bar of the terminal emulator itself. This window remains constant throughout each terminal emulator session.

The terminal emulator screen refers to the area of the window in which the application is displayed. Each time the host responds to user input to the application, the screen changes.



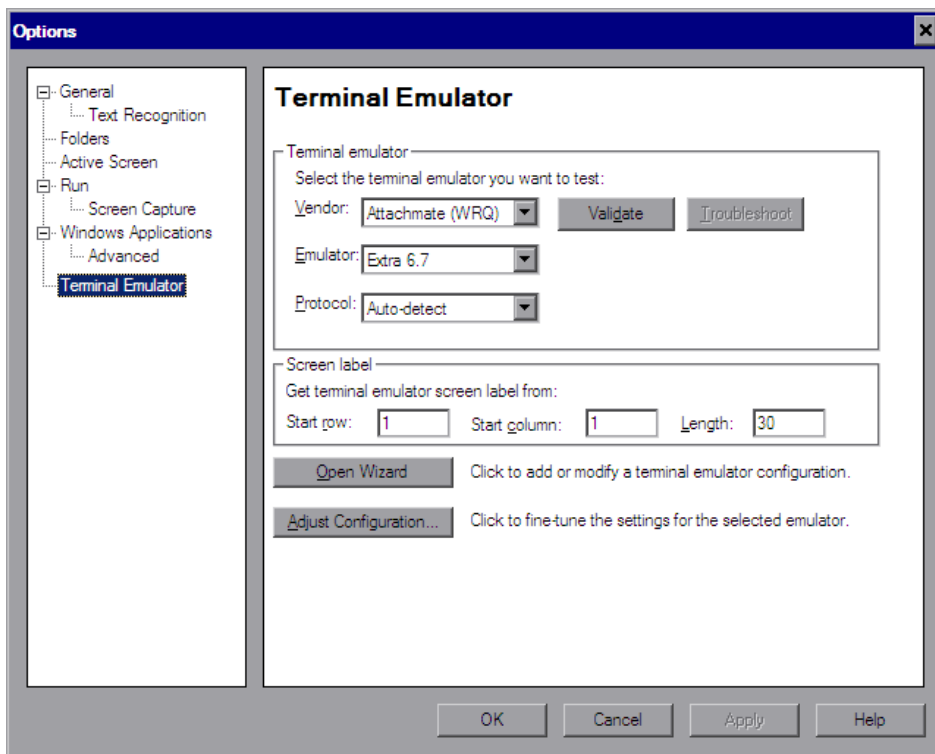
Note: When running a test or component using the QuickTest Professional Terminal Emulator Add-in, your test or component can include calls to WinRunner tests as long as these tests do not use the WinRunner Terminal Emulator Add-in. Similarly, when running a WinRunner test with the WinRunner Terminal Emulator Add-in, your test can include calls to QuickTest Professional tests, as long as these tests do not use the QuickTest Professional Terminal Emulator Add-in.

Modifying Your Terminal Emulator Settings

You configure terminal emulator settings using the Terminal Emulator Configuration Wizard. The wizard opens after you install the QuickTest Professional Terminal Emulator Add-in. For more information, see "Using the Terminal Emulator Configuration Wizard" on page 536.

If you need to change your selection or modify any of the settings, you can do so in the Terminal Emulator pane of the Options dialog box (**Tools > Options > Terminal Emulator** node). You can also validate your current terminal emulator configurations in this pane.

This pane is available only when the QuickTest Professional Terminal Emulator Add-in is installed and loaded. Any changes you make to the settings in this pane are immediately applied to the currently open test or component when you click **OK**.



You can select an emulator to test using the **Vendor** and **Emulator** list boxes. The displayed lists include all of the vendor/emulator combination settings that are:

- ▶ Preconfigured (supplied with your Terminal Emulator Add-in)
- ▶ Copied to your computer
- ▶ Previously configured using the Terminal Emulator Configuration Wizard

You can also open the wizard to configure a new terminal emulator setting or modify an existing setting.

The Terminal Emulator pane includes the following options:

- ▶ **Vendor.** The list of available terminal emulator vendors. Select the vendor for your emulator.
- ▶ **Emulator.** The list of terminal emulators available for the selected vendor. Select the emulator application you want to test.
- ▶ **Validate.** Validates the current configurations of the selected emulator, and provides a description of any detected problem. For more information, see "Validating your Terminal Emulator Configuration" on page 564.
- ▶ **Troubleshoot.** Opens a specific Help page that provides a troubleshooting solution where available. For more information, see "Validating your Terminal Emulator Configuration" on page 564.
- ▶ **Protocol.** The protocol that your emulator uses. It is recommended to select **Auto-detect**.
- ▶ **Screen label.** The area from which QuickTest reads the **label** property of the emulator screen while recording a test or component. If the location and length of the label are correctly defined, QuickTest uses this value as the name of the TeScreen object. The **Screen label** area is enabled only for emulators that support HLLAPI.

Enter the **Start row** and **Start column** coordinates that mark the beginning of the emulator label. Define the size of the label by entering the **Length** (in characters).

It is possible to change the way that QuickTest reads the **label** property of the emulator screen, by adjusting the configuration settings. For more information, see "Adjusting Your Terminal Emulator Configuration Settings" on page 593.

- ▶ **Open Wizard.** Opens the Terminal Emulator Configuration Wizard. The wizard enables you to define new settings for a terminal emulator or to modify existing user-defined settings. For more information, see "Using the Terminal Emulator Configuration Wizard" on page 536.
- ▶ **Adjust Configuration.** Opens the Terminal Emulator Configuration Adjustments dialog box to allow changes to existing configuration settings in exceptional circumstances.

In general, it is recommended to use the Terminal Emulator Configuration Wizard to configure your emulator screen settings. For more information, see "Configure Emulator Screen Settings Screen" on page 547. You should use the **Adjust Configuration** option only if you have a good understanding of your terminal emulator settings and of the impact that such changes may have on your tests or components. For more information, see Chapter 35, "Adjusting Your Terminal Emulator Configuration Settings."

Validating your Terminal Emulator Configuration

QuickTest provides a **Validate** button in the Terminal Emulator pane of the Options dialog box that checks the current configurations of the selected emulator. If a problem is detected, a brief description is displayed in the pane. You can also click the **Troubleshoot** button to view a specific troubleshooting Help page that provides additional information where available. The Validate feature reports problems resulting from:

- ▶ invalid terminal emulator configurations in the Terminal Emulator pane (**Tools > Options > Terminal Emulator** node).
- ▶ invalid configurations made when configuring the terminal emulator using the Terminal Emulator Configuration Wizard (**Tools > Options > Terminal Emulator** node > **Open Wizard**).
- ▶ errors in the terminal emulator itself.

The following possible responses and troubleshooting procedures are provided by validating and troubleshooting the emulator configuration:

- "Invalid HLLAPI DLL" on page 565
- "Cannot detect an open session" on page 565
- "Cannot locate the main window class" on page 566
- "Cannot detect the emulator screen" on page 566
- "Cannot connect to the open session" on page 566
- "Cannot retrieve session text" on page 567
- "Cannot detect open session, or Cannot locate the main window class" on page 567
- "HLLAPI DLL not found" on page 567
- "More than one session open" on page 568
- "Unknown error" on page 568

Invalid HLLAPI DLL

The required HLLAPI or EHLLAPI function cannot be found, because the configured DLL is invalid.

- Ensure that you have configured the correct DLL path and name in the Terminal Emulator Configuration Wizard (**Tools > Options > Terminal Emulator** node > **Open Wizard**).
- For more information, see the table listing the DLL names used by supported terminal emulators in "Configure HLLAPI Properties Screen" on page 541, or the documentation provided by your emulator provider.

Cannot detect an open session

QuickTest cannot detect an open terminal emulator session.

- Ensure that you have opened a current session in your terminal emulator.
- For HLLAPI emulators, ensure that the emulator short session name is set to the uppercase letter **A**. You may need to restart the emulator after changing this setting.

Cannot locate the main window class

QuickTest cannot find the terminal emulator main window class name.

- ▶ Ensure that you have correctly configured the terminal emulator main window class name in the Terminal Emulator Configuration Wizard (**Tools > Options > Terminal Emulator** node > **Open Wizard**).
- ▶ If the main window class name has a postfix that changes each time you launch the emulator, enter only the non-changing portion of the name in the Terminal Emulator Configuration Wizard.

Cannot detect the emulator screen

QuickTest cannot find the terminal emulator main window class name.

- ▶ Ensure that you have correctly configured the terminal emulator main window class name in the Terminal Emulator Configuration Wizard (**Tools > Options > Terminal Emulator** node > **Open Wizard**).
- ▶ If the main window class name has a postfix that changes each time you launch the emulator, enter only the non-changing portion of the name in the Terminal Emulator Configuration Wizard.

Cannot connect to the open session

Although a current session is open, invoking an HLLAPI function resulted in an error.

- ▶ Restart QuickTest Professional and then restart the emulator. If this does not resolve the problem, contact your emulator provider.

Cannot retrieve session text

QuickTest cannot display text captured in the current session.

- HLLAPI Emulators—Restart QuickTest Professional and then restart the emulator. If this does not resolve the problem, contact your emulator provider.
- Non-HLLAPI Emulators—Click **Validate** again. If the error message is repeated, check that the emulator screen is brought to the front during the validate process (even when using remote access). If this is the case, contact HP Customer Support.

Cannot detect open session, or Cannot locate the main window class

QuickTest cannot detect an open terminal emulator session, or find the terminal emulator main window class name.

- Ensure that you have opened a current session in your terminal emulator.
- Ensure that you have correctly configured the terminal emulator main window class name in the Terminal Emulator Configuration Wizard (**Tools > Options > Terminal Emulator** node > **Open Wizard**).
- If the main window class name has a postfix that changes each time you launch the emulator, enter only the non-changing portion of the name in the Terminal Emulator Configuration Wizard.

HLLAPI DLL not found

QuickTest cannot find the HLLAPI DLL specified for the selected emulator.

- Ensure that you have configured the correct DLL path and name in the Terminal Emulator Configuration Wizard (**Tools > Options > Terminal Emulator** node > **Open Wizard**).
- For more information, see the table listing the DLL names used by supported terminal emulators in "Configure HLLAPI Properties Screen" on page 541, or the documentation provided by your emulator provider.

More than one session open

More than one terminal emulator session is currently open.

- Close additional sessions.

Unknown error

The validation process failed due to an unknown error.

- Restart QuickTest Professional and then restart the emulator.

Understanding the Test Object Model

The *test object model* is the set of object types or *classes* that QuickTest uses to represent the objects in your application. Each test object class has a list of properties that can uniquely identify objects of that class, and a set of relevant operations that QuickTest can perform during a run session.

For example, suppose you type **Guest** into an appropriate field on a **Sign On** screen of your terminal emulator application. This field has the text **User** attached to it.

QuickTest identifies the field as a TeField object. It creates a TeField test object with the name **User** and records the following properties and values as the description for the **User** TeField:

Type	Property	Value
ABC	attached text	User
ABC	protected	False

It also records that you performed a Set method on the TeField object.

QuickTest displays your step in the Keyword View like this:

Item	Operation	Value	Documentation
▼ Action1			
▼ TeWindow			
▼ Sign On			
User	Set	"GUEST"	Enter "GUEST" in the "User" field.
Password	SetSecure	"40beca759186e..."	Enter the encrypted string "40beca759186e7f20d496981"

QuickTest displays your step in the Expert View like this:

```
TeWindow("TeWindow").TeScreen("Sign On").TeField("User").Set "Guest"
```

Identifying Test Object Classes for Terminal Emulators

QuickTest, with add-in support for terminal emulators, identifies the following test objects:

- TeWindow Object
- TeScreen and TeField Objects
- TeTextScreen Object

TeWindow Object

TeWindow is the *window* test object for all supported terminal emulators, consisting of the frame, menus, toolbar, and status bar (where applicable) of the terminal emulator itself. It represents the terminal emulator window for the session.

The TeWindow object has a number of properties associated with it, but no window-specific methods.

You can use the values of the **Emulator status** property and the other properties of the TeWindow object to define recovery scenarios for your terminal emulator application tests or components. Recovery scenarios define possible unexpected events and errors, and the operations necessary to recover the run session. For more information on recovery scenarios for terminal emulator tests or components, see "Understanding Terminal Emulator Recovery Scenarios" on page 573.

The property values for the TeWindow object can be retrieved using the GetTOProperty and GetROProperty methods. For more information on these methods and the TeWindow object properties, see the *HP QuickTest Professional Object Model Reference*.

You can record the selection of menu options in the terminal emulator window, as well as the operations that are performed in the dialog boxes that open from the menu options. For example, you can test a business process in which the user selects a menu option that displays a dialog containing a list of macros, and then selects a macro to run.

QuickTest records these operations using the standard Windows test objects and methods for menus and dialog boxes. For more information on standard Windows objects, methods, and properties, see the *HP QuickTest Professional Object Model Reference*.

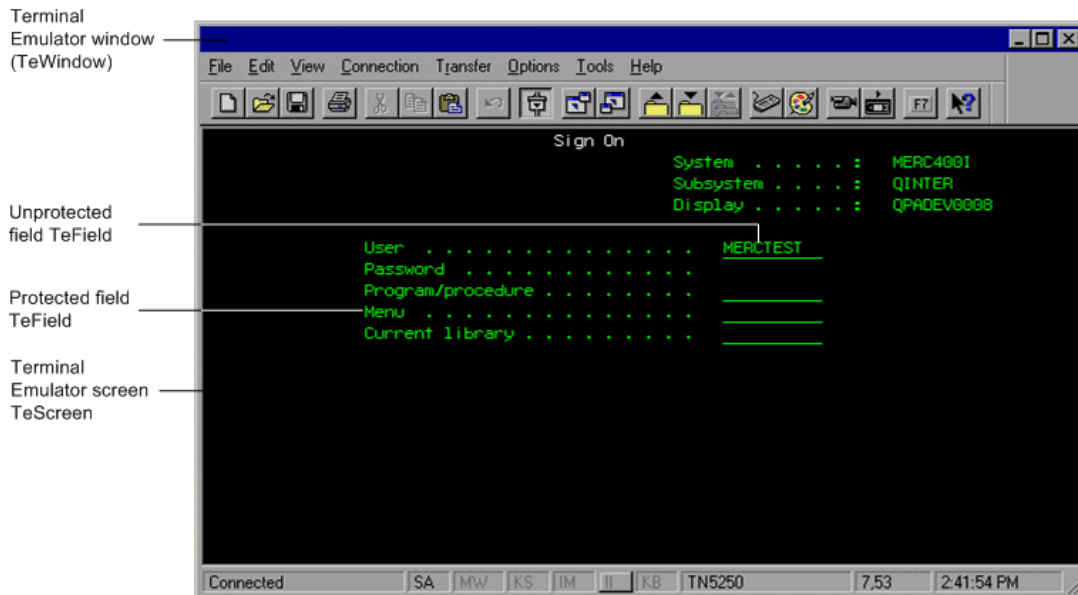
It is possible to prevent QuickTest from recording menu selections and the resulting dialog boxes by adjusting the terminal emulator configuration settings. For example, you may not want emulator-specific menu and dialog box steps in your test or component if cross-emulator compatibility is important, or if these steps are not relevant to your test or component. For more information, see Chapter 35, "Adjusting Your Terminal Emulator Configuration Settings."

Note: QuickTest does not record operations on the toolbar and status bar in the terminal emulator window. However, you can insert checkpoints or output values for the status bar of the terminal emulator window while recording. For more information, see Chapter 34, "Enhancing Your Terminal Emulator Tests and Components."

TeScreen and TeField Objects

The TeScreen and TeField objects are used for preconfigured terminal emulators, or those that have been user-defined as fully supporting HLLAPI. TeScreen is the *screen* test object, and TeField is the *field* test object.

The TeScreen object is the application area. It changes each time input is received from the host. The TeField object includes unprotected fields, which can receive input, and protected fields, which contain fixed text.

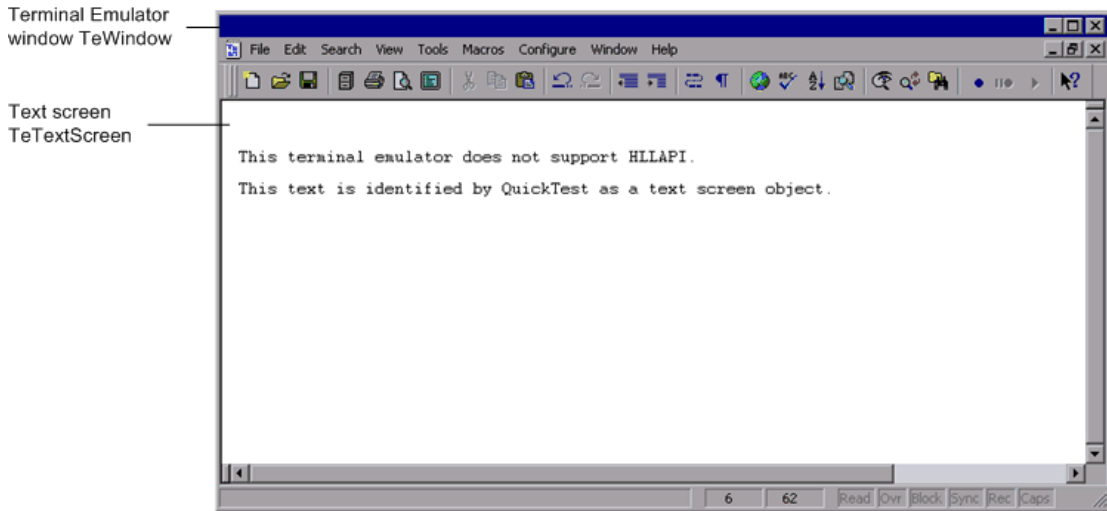


Note: By default, QuickTest identifies a screen object using its **label** property. The value of the **label** property is determined by the location of the label area, which is defined by the values of the **screen label column**, **screen label length** and **screen label row** properties. These three properties are part of the TeScreen test object description and are therefore not available in the Object Identification dialog box. For more information on the Object Identification dialog box, see the *HP QuickTest Professional User Guide*.

TeTextScreen Object

The TeTextScreen object is the *text screen* test object for terminal emulators that do not support HLLAPI or that have been configured as supporting text-only HLLAPI operations.

With the TeTextScreen object, QuickTest identifies elements in the terminal emulator screen as text on a specific location in the screen. It does not recognize fields.



Note: The recorded and smart identification properties of the TeTextScreen object cannot be configured. This object is therefore not included in the **Test Object classes** list for the **Terminal Emulators** environment in the Object Identification dialog box. For more information on the Object Identification dialog box, see the *HP QuickTest Professional User Guide*.

Understanding Terminal Emulator Recovery Scenarios

QuickTest allows you to define recovery scenarios for your tests or components, to cater for various unexpected events, such as crashes and error situations, which can disrupt your tests or components and distort your results.

You can use the values of the **Emulator status** property and the other properties of the TeWindow object to define specific recovery scenarios for your terminal emulator application tests or components.

The possible values for the **Emulator status** property are:

- **Busy.** Emulator is communicating with the server.
- **Disconnected.** Emulator is not connected to the server.
- **Locked.** Emulator cannot currently accept input.
- **Ready.** Emulator is waiting for input.
- **Unavailable.** Emulator status cannot be identified.

For each emulator status, you can create a recovery scenario that performs an appropriate recovery operation. For example:

- **Disconnected.** Reconnect to the server, using a function call recovery operation that includes recorded steps for connecting, API commands in a VB Script, or a keyboard shortcut key, according to the capabilities of your terminal emulator.
- **Ready.** Perform specific operations according to the content of a displayed error message, including pressing the relevant key.
- **Locked.** Activate the emulator's RESET key, or use a handler function to disconnect from the server and reconnect.

For detailed information on defining recovery scenarios, see the *HP QuickTest Professional User Guide*.

Recording Tests and Components on Terminal Emulator Applications

As you record, the test or component reflects the objects in your application and the type of operation you perform (such as pressing function keys or typing in fields). Each object has a defined set of properties that determines its behavior and appearance. QuickTest learns these properties and uses them to identify and locate objects during a run session.

Tip: You can launch your terminal emulator using the **SystemUtil.Run** method as the first step of your test or component. For more information, see the section on running and closing applications programmatically in the *HP QuickTest Professional User Guide*, and the **Standard Windows** section of the *HP QuickTest Professional Object Model Reference*.

By default, when you record a test or component, QuickTest automatically inserts synchronization points so that during a run session, execution will be delayed until the application is ready to receive input. You can also add synchronization points manually. For more information, see "Synchronizing the Run Session" on page 587.

The following is a sample of a QuickTest test recorded on a terminal emulator application that fully supports HLLAPI.

While recording, the user pressed the ENTER key in the first screen of an application, waited for the screen to change, and then typed the name MERCTEST and a password in the appropriate fields.

```
TeWindow("TeWindow").TeScreen("Welcome").SendKey TE_ENTER
TeWindow("TeWindow").TeScreen("Welcome").Sync
TeWindow("TeWindow").TeScreen("Sign On").TeField("User").Set "MERCTEST"
TeWindow("TeWindow").TeScreen("Sign On").TeField("Password").
    SetSecure "3c4feb5bc6233d6e6898bc"
```

QuickTest displays this test in the Keyword View like this:

Item	Operation	Value	Documentation
▼ Action1			
▼ TeWindow			
Welcome	SendKey	TE_ENTER	Press the TE_ENTER keyboard key.
Welcome	Sync		Wait for the "Welcome" screen to synchronize.
▼ Sign On			
User	Set	"MERCTEST"	Enter "MERCTEST" in the "User" field.
Password	SetSecure	"3c4feb5bc62..."	Enter the encrypted string "3c4feb5bc6233d6e6898bc" in th

The following is a sample test on a terminal emulator that does not support HLLAPI or that has been configured to support text-only HLLAPI operations.

Note that QuickTest records the TeTextScreen object instead of the TeScreen object and that it does not record TeField objects. The operations are recorded in terms of keyboard and mouse operations on the text screen, rather than operations within fields.

```
TeWindow("TeWindow").TeTextScreen("TeTextScreen").ClickPosition 24,2
TeWindow("TeWindow").TeTextScreen("TeTextScreen").Type ""
TeWindow("TeWindow").TeTextScreen("TeTextScreen").Type micReturn
TeWindow("TeWindow").TeTextScreen("TeTextScreen").WaitString
    "FRSMAIN", 1,2,1,8,2000
TeWindow("TeWindow").TeTextScreen("TeTextScreen").Type "qa1"
TeWindow("TeWindow").TeTextScreen("TeTextScreen").Type micReturn
TeWindow("TeWindow").TeTextScreen("TeTextScreen").Sync
```

QuickTest displays this test in the Keyword View like this:

Item	Operation	Value	Documentation
▼ Action1			
▼ TeWindow			
TeTextScreen	ClickPosition	24,2	Click row 24, column 2 of the "TeTextScreen" screen.
TeTextScreen	Type	"I"	Type "I" in the "TeTextScreen" screen.
TeTextScreen	Type	micReturn	Type micReturn in the "TeTextScreen" screen.
TeTextScreen	WaitString	"FRSMAIN", ...	Wait 2000 milliseconds for the "FRSMAIN" string to appe
TeTextScreen	Type	"qa1"	Type "qa1" in the "TeTextScreen" screen.
TeTextScreen	Type	micReturn	Type micReturn in the "TeTextScreen" screen.
TeTextScreen	Sync		Wait for the "TeTextScreen" screen to synchronize.

Note: If you are using an emulator that is configured as fully supporting HLLAPI and you need to record specific steps in terms of keyboard and mouse operations on the text screen (instead of operations within fields), it is possible to change the recording mode for your emulator by adjusting the configuration. For more information, see Chapter 35, "Adjusting Your Terminal Emulator Configuration Settings."

Troubleshooting and Limitations - Terminal Emulator

This section contains general troubleshooting and limitation information about the Web add-in, and includes the following sections:

- "Installing and Loading the Terminal Emulator Add-in" on page 577
- "Connecting and Disconnecting from the Terminal Emulator Add-in" on page 577
- "Configuration and Settings" on page 578
- "Creating and Running Tests and Components" on page 579
- "Working with Terminal Emulator Controls" on page 581
- "Test Objects, Methods, and Properties" on page 581
- "Checkpoints and Output Values" on page 583
- "Multilingual Support" on page 583

Installing and Loading the Terminal Emulator Add-in

- ▶ When installing a Hummingbird HostExplorer terminal emulator or patches, make sure that QuickTest Professional is closed.
- ▶ If the QuickTest Professional Terminal Emulator Add-in is installed and loaded, but there is no terminal emulator installed on your computer, the following error message is displayed: QuickTest Terminal Emulator support is not configured correctly. Either the terminal emulator is not installed on your computer or the HLLAPI DLL was not found.

Workaround: When you open QuickTest, clear the **Terminal Emulators** check box in the Add-in Manager.

Note: You can prevent this message from appearing by adjusting your emulator's configuration settings. For more information, see the *HP QuickTest Professional Add-ins Guide*.

- ▶ You may experience unexpected behavior after you install an EXTRA! emulator. You may not be able to run QuickTest Professional or various features may stop working. This happens because the EXTRA! installation may have copied and registered an outdated version of the **atl.dll** file on your computer.

Workaround: Locate the **atl.dll** in your system folder (WINNT\system32). Its version should be 3.0 or higher. Register it with the **regsvr32** utility.

Connecting and Disconnecting from the Terminal Emulator Add-in

- ▶ If you have more than one terminal emulator session open, QuickTest does not recognize either session.

Workaround: While recording or running your test or component, make sure that only one terminal emulator session is connected at a time.

- ▶ If your test or component contains steps that disconnect the current emulator session during the run session, followed immediately by a **TeScreen.Sync** command, the test or component run might stop responding or take a long time to respond.

Workaround: Remove the **Sync** command from the test or component, or replace it with a **Wait** statement. For more information, see the **Utility Objects** section of the *HP QuickTest Professional Object Model Reference*.

- ▶ Inserting a checkpoint, creating a new test or component, or opening an existing test or component when the emulator session is busy may cause unexpected problems.

Workaround: Check the connection status of your emulator on the status line of the emulator screen before performing any of these operations.

- ▶ Unexpected behavior may occur after disconnecting from a Host On-Demand session while recording.

Workaround: Stop recording before disconnecting from the session. Then, manually add a step that disconnects from the session.

- ▶ You may experience unexpected behavior if the terminal emulator is closed while QuickTest is recording.

Configuration and Settings

- ▶ When working with an emulator that does not support HLLAPI, or with an emulator that has been configured as supporting text-only HLLAPI operations, do not change the size of the terminal emulator window after configuring the emulator settings.

- To enable support for a NetManage Web-To-Host Java Client session that is configured to open in a separate window, specify the title of your session window using the **Tools > Options > Terminal Emulator > Adjust Configuration > Object identification settings > Identify emulator window based on title bar prefix** option.

Tip: You may need to clear this value when switching to another configuration.

- When using the Terminal Emulator Configuration Wizard to configure the screen sizes of NetManage RUMBA Web-to-Host, you cannot use the **Mark Text Area** option to draw on top of the emulator window.

Workaround: Configure the text area position of the screen manually.

Creating and Running Tests and Components

- When using the OCR mechanism in order to perform steps requiring text recognition on non-HLLAPI emulators, the steps run slowly due to the required processing power of the OCR mechanism. Therefore, when testing non-HLLAPI emulators, it is recommended to select the default text recognition option: **First Windows API then OCR** in the Text Recognition pane of the Options dialog box. (For details on this option, see the *HP QuickTest Professional User Guide*.)
- The QuickTest Professional Terminal Emulator Add-in can identify emulator window objects only when the emulator is connected. For example, you cannot use the following statement to connect to an emulator session:

```
TeWindow("TeWindow").WinMenu("Menu").Select("Communication;Connect")
```

Workaround: You can record any steps that need to be performed prior to connection with the emulator. These steps are recorded as if the Terminal Emulator Add-in is not loaded. After the emulator is connected, stop the recording session and begin a new recording session to record terminal emulator objects.

- ▶ When using an emulator that supports HLLAPI, if your emulator session disconnects from the host while recording, QuickTest no longer recognizes the emulator, even after reconnecting.

Workaround: Stop recording, reconnect the session, and continue recording.

- ▶ When recording on a Hummingbird HostExplorer emulator, menu and toolbar operations in the emulator window are disabled.

Workaround: Stop recording, select the required menu item or click the required toolbar button, and continue recording.

- ▶ When using an emulator that supports HLLAPI, closing the emulator window while recording may cause unexpected results.

Workaround: Stop recording before closing the emulator window.

- ▶ The QuickTest Professional Terminal Emulator Add-in does not support recording operations on toolbar objects in terminal emulator applications.

Workaround: Record on the corresponding menu command for the toolbar button. Alternatively, you can use low-level recording to record operations on toolbars. For more information about low-level recording, see the *HP QuickTest Professional User Guide*.

- ▶ If you record a test or component using one terminal emulator, it may not run correctly on another terminal emulator. For example, tests recorded on RUMBA may not run on IBM PCOM.

- ▶ HostExplorer has a bug in the HLLAPI GetKey function. As a result, QuickTest will stop recording terminal emulator keyboard events after recording for a while, and the emulator might stop responding to keyboard events.

Workaround: Contact Hummingbird customer support to get the patch that fixes the problem with the HLLAPI GetKey function (where it stops responding after several calls).

- ▶ Clicking, typing, or moving objects in the terminal emulator window while QuickTest is running a test or component may cause unexpected results.

Workaround: Wait until the end of the test or component, or pause the test or component execution before using the emulator.

- ▶ To record and run tests or components on Hummingbird 9.0 5250 sessions, you need to install a patch for Hummingbird.

Workaround: Contact Hummingbird customer support to get the patch that fixes the problem with HLLAPI where all 5250 fields appear protected.

Working with Terminal Emulator Controls

- ▶ When working with Attachmate Terminal Viewer 3.1 5250 session, all of the fields that appear on the screen before the first unprotected field are recognized as a single field.
- ▶ QuickTest may not recognize a TeField object in a NetManage RUMBA session immediately after installing the emulator.

Workaround: Restart your computer after installing RUMBA, even if the installation does not request a restart.

Test Objects, Methods, and Properties

- ▶ When using the **SendKey** method to unlock a terminal emulator, for example, `TeWindow("TeWindow").TeScreen("screen5296").SendKey TE_RESET`, some emulators (such as Host On-Demand) may not be unlocked.

Workaround: Specify the keyboard event to send for the RESET command, using the **Tools > Options > Terminal Emulator > Adjust Configuration > Run Settings > Run steps containing special emulator keys using keyboard events > Keys for RESET function** option.

- ▶ By default, QuickTest uses the **attached text** and **protected** properties in TeField test object descriptions. If the attached text for a field changes from session to session, QuickTest cannot find the field during the run session.

Workaround: Open the Object Repository dialog box or the Object Properties dialog box for the object. Remove the **attached text** property from the field's description and add another property (or properties) such as **start row**, **start column**, or **index** to uniquely identify the object.

Tip: You can also create a smart identification definition for TeField objects so that your recorded test or component can run successfully even if the **attached text** property value for a particular TeField object changes. (Select **Tools > Object Identification > Enable Smart Identification** and click **Configure**.) For more information on Smart Identification, see the *HP QuickTest Professional User Guide*.

- ▶ You cannot use the **label** property in a programmatic description of the TeScreen object. However, since only one screen can exist in the given TeWindow at any one time, you can use TeScreen("MicClass:=TeScreen").

For example:

```
TeWindow("short  
name:=A").TeScreen("MicClass:=TeScreen").TeField("attached text:=User",  
"Protected:=False").Set "33333"
```

- ▶ The TeTextScreen properties **current column** and **current row** are available only for emulators that support HLLAPI.
- ▶ The **location** property is not recorded for TeField objects.

Workaround: Use the **index** property instead.

Checkpoints and Output Values

In some cases, a bitmap checkpoint on a TeScreen may fail because the cursor shows in the expected bitmap, and not in the actual bitmap (or the other way around).

Workaround: Set the emulator cursor to a slow blink rate, or not to blink at all. This enhances the probability that the cursor is not captured in the bitmap.

Multilingual Support

When working with the IBM PCOM emulator, QuickTest may ignore special European language characters while recording or running a test or component.

Workaround: Set the code page for your IBM PCOM emulator in QuickTest, using the **Tools > Options > Terminal Emulator > Adjust Configuration > Emulator settings > Code page number (IBM PCOM only)** option.

Tip: Try setting the **Code page number (IBM PCOM only)** option to 1252.

34

Enhancing Your Terminal Emulator Tests and Components

After you create your test or component, you can enhance it by adding checkpoints, retrieving output values, adding synchronization points, parameterizing values, and inserting terminal emulator objects, methods and properties into your test or component. For example, you can use standard checkpoints to check the number of protected or input fields in a screen, or you can check the content of a specific field and whether it is protected or visible.

This chapter includes:

- ▶ Working with Checkpoints and Output Values on page 586
- ▶ Synchronizing the Run Session on page 587
- ▶ Identifying Test Object Classes and Icons on page 591

Working with Checkpoints and Output Values

While recording your test, you can add text checkpoints for the TeScreen and TeTextScreen objects, for the status bar of the terminal emulator window, and for the dialog boxes that open after menu options are selected. While editing your test or component, you can add text checkpoints for TeScreen objects. You can also output property or text values from the objects in your terminal emulator application to use in your test or component.

Guidelines for Using Checkpoints and Output Values

- ▶ When working with tests, you can add text checkpoints for TeTextScreen objects while editing, if the test was recorded using an emulator with full HLLAPI support that has been configured to record in **Text screen** mode. For more information on changing the emulator mode, see Chapter 35, "Adjusting Your Terminal Emulator Configuration Settings."
- ▶ You can create bitmap checkpoints for TeWindow, TeScreen and TeTextScreen objects, but not for TeField objects.
- ▶ You can create text output values (tests only) only for TeScreen and TeTextScreen objects.
- ▶ In the terminal emulator window you can add text checkpoints or output values (tests only) and standard checkpoints and output values for the status bar and the dialog boxes that open from the menu options. QuickTest recognizes these as standard Windows objects. For more information on the properties of standard Windows objects, see the *HP QuickTest Professional Object Model Reference*.

For more information on standard, text, and bitmap checkpoints, and on standard and text output values, see the *HP QuickTest Professional User Guide*.

Synchronizing the Run Session

When testing a terminal emulator application, many factors can affect its speed of operation and therefore can potentially interfere with the run session. For example, host response time can vary depending on the system load.

Synchronizing your run session ensures that QuickTest performs the next step in the test or component only when your terminal emulator application is ready to continue. This prevents incidental differences in host response time and other factors from affecting successive run sessions.

For more information, see:

- "About Synchronizing the Run Session" on page 587
- "Synchronizing with the Host" on page 588
- "Waiting for a Specified Text String" on page 589
- "Setting Synchronization Timeouts" on page 590
- "Inserting a Synchronization Point for an Object" on page 591

About Synchronizing the Run Session

The QuickTest Professional Terminal Emulator Add-in provides various synchronization options that you can use to pace your run session. These can be inserted into your test or component automatically or with programming statements.

For all emulators, you can instruct QuickTest to delay the run session:

- For a specified period of time
- Until a specific string appears in a defined area
- Until a specified property achieves a defined value

For emulators that fully support HLLAPI, you can also synchronize the run session with the response time of the host.

Synchronizing with the Host

By default, when you record using a terminal emulator that fully supports HLLAPI, QuickTest automatically generates a Sync statement for the TeScreen object each time the emulator waits for a response from the host.

When you record using a terminal emulator that does not support HLLAPI, or that has been configured as supporting text-only HLLAPI operations, QuickTest automatically generates a Sync statement for the TeTextScreen object each time a specified key is pressed. The default is the ENTER key. QuickTest waits a specified period of time, to allow the host sufficient response time.



You can also insert a synchronization step at any point during a recording session. Select **Insert > Emulator Synchronization** or click the **Insert Emulator Synchronization Step** toolbar button.

You can optionally specify a timeout in milliseconds for the Sync statement, after which the run session continues regardless of the status of the emulator. If you do not specify a timeout value, QuickTest uses the default timeout interval, as described in "Setting Synchronization Timeouts" on page 590.

For the TeScreen object, the Sync method has the following syntax:

```
TeScreen(description).Sync [Timeout]
```

Using this method with a TeScreen object ensures that the run session is delayed until a response is received from the host and the emulator status is set to **Ready** (able to receive user input).

For the TeTextScreen object, the Sync method has the following syntax:

```
TeTextScreen(description).Sync [Timeout]
```

For `TeTextScreen` objects, the `Sync` statement simply instructs `QuickTest` to wait for the specified period of time before continuing to the next step in the test or component. You may find it more efficient to use the `WaitString` method for `TeTextScreen` objects, as described in "Waiting for a Specified Text String" on page 589.

Notes:

- ▶ Your emulator configuration can be adjusted to prevent `QuickTest` from automatically inserting `Sync` steps for `TeScreen` objects in your test or component.
 - ▶ It is also possible to specify the keys that generate `Sync` steps for `TeTextScreen` objects.
 - ▶ For more information, see Chapter 35, "Adjusting Your Terminal Emulator Configuration Settings."
-

Waiting for a Specified Text String

`QuickTest`'s `WaitString` method delays the run session until a specific text string appears in a specified rectangle on the terminal emulator screen. The specified text string can be a constant string or a regular expression.

To insert a `WaitString` statement while recording:



- 1** Select **Insert > Emulator WaitString** or click the **Insert Emulator WaitString Step** toolbar button. Your cursor becomes a crosshairs pointer.
- 2** Drag the pointer to draw a rectangle on your emulator screen containing the text string for which you want the run session to wait. `QuickTest` inserts a step into your test or component with the following syntax:

`TeScreen` object:

```
TeScreen(description).WaitString String [, TopRow, LeftColumn, BottomRow, RightColumn, Timeout, RegExp]
```

`TeTextScreen` object:

```
TeTextScreen(description).WaitString String [, TopRow, LeftColumn, BottomRow, RightColumn, Timeout, RegExp]
```

The position on the screen is defined by the values of the four corners of the rectangle, each corner with its own argument.

You can specify that the value specified in the **String** argument is a regular expression by setting the value of the **RegExp** argument to **True**. Regular expressions enable QuickTest to identify objects and text strings with varying values. For more information on regular expressions, see the *HP QuickTest Professional User Guide*.

You can also add an optional timeout value in milliseconds after which the run session continues regardless of whether the text string appears on the screen. If you do not specify this value, QuickTest uses the default timeout interval. For more information, see "Setting Synchronization Timeouts" on page 590.

The `WaitString` method returns a value of **True** if the string appears on the screen within the timeout period and **False** if the timeout expires before the string appears.

Setting Synchronization Timeouts

For tests, you can set the maximum interval (in milliseconds) that QuickTest waits before running each test step. You do this by setting the **Object Synchronization Timeout** in the Run pane of the Test Settings dialog box (**File > Settings > Run** node).

Note: This option is not available for components.

This setting is also used as the default timeout for the `Sync` and `WaitString` methods for both the `TeScreen` and the `TeTextScreen` objects if a timeout argument is not specified. For more information on the Test Settings dialog box, see the *HP QuickTest Professional User Guide*.

Inserting a Synchronization Point for an Object




To instruct QuickTest to pause the test or component until a particular object property achieves the value you specify, you can insert a *synchronization point*. When you insert a synchronization point into your test or component, QuickTest generates a **WaitProperty** statement in the Expert View. For example, if you want the run session to wait until the Text property of the Result field has a value of Successful, you insert the following statement:

```
TeScreen("LogOn").TeField("Result").WaitProperty "Text", "Successful"
```



For more information on synchronization points, see the *HP QuickTest Professional User Guide*.

Identifying Test Object Classes and Icons



The following test object classes and icons apply to terminal emulators that are configured as fully supporting HLLAPI:

Icon	Test Object Class
	TeField
	TeScreen
	TeWindow

The following test object classes and icons apply to terminal emulators that do not support HLLAPI or that have been configured as supporting text-only HLLAPI operations:

Icon	Test Object Class
	TeTextScreen
	TeWindow

The following test object classes and icons apply to the Windows objects for the terminal emulator window status bar and the dialog boxes that open from the menu options in the terminal emulator window:

Icon	Test Object Class
	Dialog
	WinObject

35

Adjusting Your Terminal Emulator Configuration Settings

In the majority of cases, QuickTest works successfully with terminal emulators using the supplied preconfigured settings and with the configuration settings defined using the Terminal Emulator Configuration Wizard. In exceptional circumstances it may be necessary to make minor adjustments to the configuration settings using the options in the Terminal Emulator Configuration Adjustments dialog box, which can be accessed by clicking the **Adjust Configuration** button on the **Tools > Options > Terminal Emulator** node.

Note: Do not change the configuration settings using the options in this dialog box unless you have a good understanding of your terminal emulator and of the impact that such changes may have on your tests or components.

The configuration adjustment options include settings for preconfigured emulators and user-defined emulator configurations, with and without HLLAPI support. The type of emulator selected determines which options are available.

For information on changing the configuration settings, see "Using the Terminal Emulator Configuration Adjustments Dialog Box" on page 594.

For an explanation of the purpose and effect of each of the options in this dialog box, see "Understanding the Configuration Adjustment Options" on page 597.

This chapter includes:

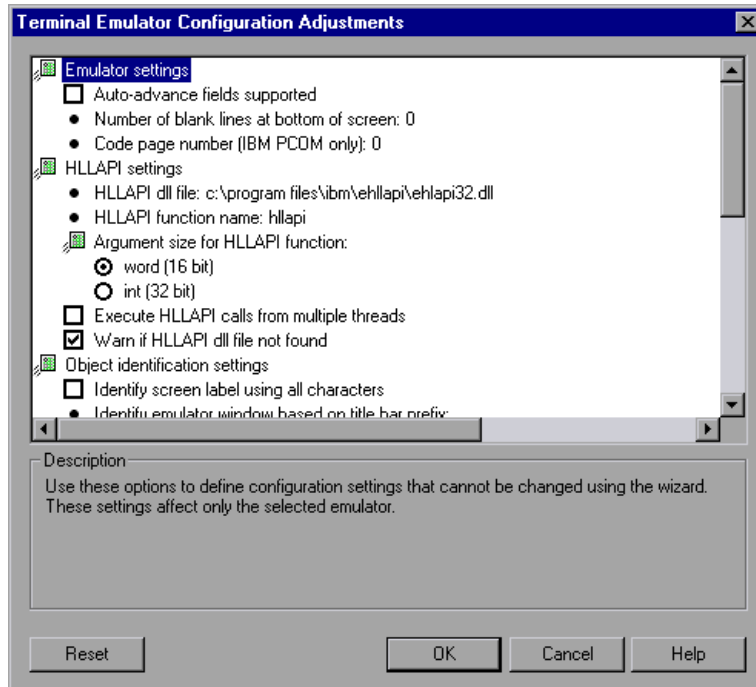
- ▶ Using the Terminal Emulator Configuration Adjustments Dialog Box on page 594
- ▶ Understanding the Configuration Adjustment Options on page 597

Using the Terminal Emulator Configuration Adjustments Dialog Box

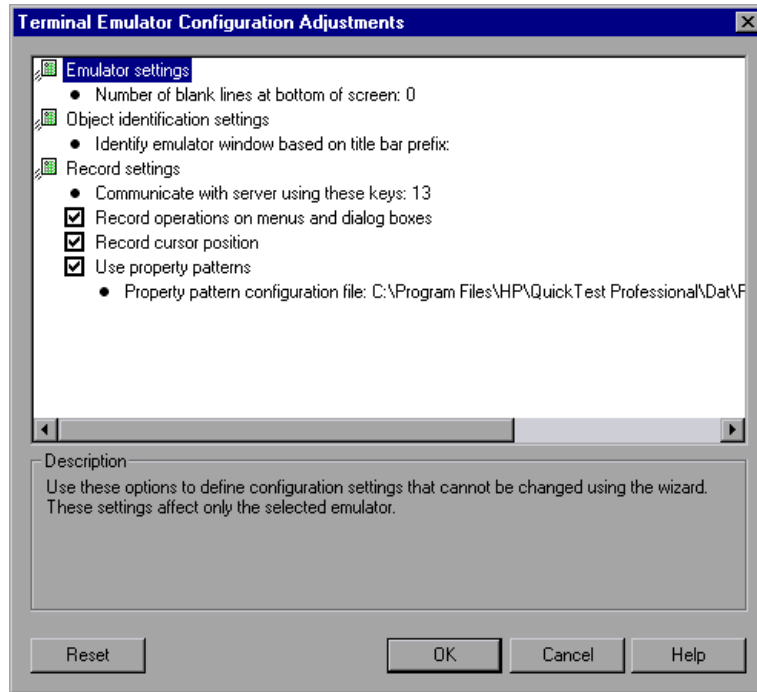
When you select an emulator in the Terminal Emulator pane and click **Adjust Configuration**, the Terminal Emulator Configuration Adjustments dialog box opens, displaying the current settings.

The choice of options displayed in this dialog box depends on the type of terminal emulator selected in the Terminal Emulator pane.

If you have selected an emulator that supports HLLAPI, the options displayed in the dialog box include the HLLAPI-specific options.



If you have selected an emulator that does not support HLLAPI or is configured for text-only support, the HLLAPI-specific options are not available.



Changing Configuration Settings

The Terminal Emulator Configuration Adjustments dialog box contains check boxes, radio buttons, and options that require a numeric or text value.

To enter a numeric or text value for an option:

- 1 Click the option once to highlight it.
- 2 Click the option again or press F2 to access the value to be changed.
- 3 Change the value as necessary.
- 4 When you have finished editing the value, click another location in the dialog box to set the value.

When you have made all the required changes, click **OK** to update the current terminal emulator configuration and close the dialog box.

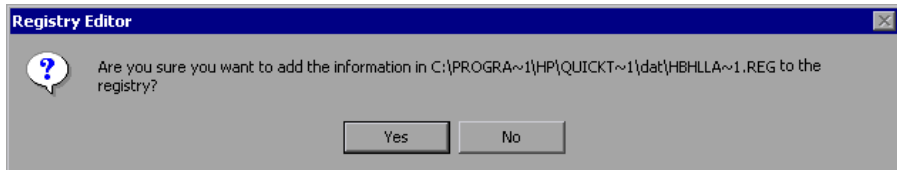
Restoring Configuration Settings

You can restore the default settings for the selected preconfigured emulator by clicking the **Reset** button in the Terminal Emulator Configuration Adjustments dialog box. This button is enabled only if a preconfigured emulator is selected.

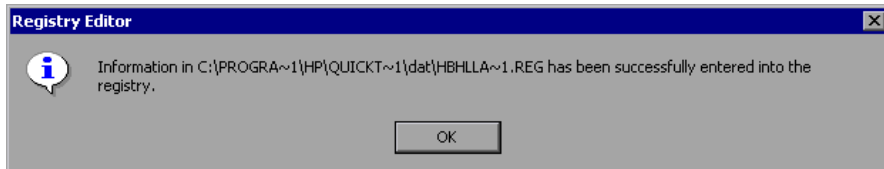
You can also restore the settings for a user-defined terminal emulator, if these settings were previously saved using the wizard. For more information on saving emulator settings, see "Completing the Terminal Emulator Configuration Wizard Screen" on page 549.

To restore the settings for a user-defined configuration:

- 1 Locate the saved registry file that contains the configuration settings in the <QuickTest installation folder>\dat folder on your computer. The file has a .reg extension. The path for the dat folder in a typical installation is: C:\Program Files\HP\QuickTest Professional\dat.
- 2 Double-click the registry file to activate the registry file. A confirmation message opens.



- 3 Click **Yes**. A message opens confirming that the information has been copied into the registry.



- 4 Click **OK**. The settings in the saved file have been restored.

Understanding the Configuration Adjustment Options

The majority of the options in the Terminal Emulator Configuration Adjustments dialog box are specific to the terminal emulator selected in the Terminal Emulator pane of the Options dialog. The values for these emulator-specific options are saved with the selected emulator. For example, if you specify an HLLAPI DLL file other than the default file, the specified file is used only for the selected emulator.

For a small number of options, the value is saved and applied regardless of the emulator selected in the Terminal Emulator pane of the Options dialog. For example, if you select not to record menus and dialog boxes, QuickTest keeps this setting even if you select a different emulator.

The Terminal Emulator Configuration Adjustments dialog box contains the following groups of options:

- Emulator Settings
- HLLAPI Settings
- Object Identification Settings
- Record Settings
- Run Settings

Some of the options may not be available for the selected terminal emulator. For example, if you have selected an emulator that does not support HLLAPI, the HLLAPI-specific options are not available.

Tip: You modify options that are displayed as bullets by clicking the text and modifying the value when the text becomes an editable box. For more information, see "Changing Configuration Settings" on page 595.

Emulator Settings

The following options can be used to define configuration settings that cannot be changed using the wizard:

- ▶ **Auto-advance fields supported.** Auto-advance fields allow an application to proceed automatically to the next screen or field after input of a predefined number of characters without pressing ENTER or another key.

If your emulator supports auto-advance fields, select this check box to enable QuickTest to record **Set** statements on these fields.

This option is available only for emulators that support HLLAPI.

- ▶ **Number of blank lines at bottom of screen.** Some emulators reserve blank lines at the bottom of the screen. If the screen size changes, these lines may distort QuickTest's calculation of field locations. This option enables you to specify the number of blank lines at the bottom of the emulator screen. It is recommended to use the Terminal Emulator Configuration Wizard to configure this setting, but you can also modify the setting using this option.

Enter the number of lines that your emulator reserves at the bottom of the screen. QuickTest includes this value in its algorithm for identifying field locations.

- ▶ **Code page number (IBM PCOM only).** If you are using an IBM PCOM emulator with a language other than English, enter the code page number for this language. For example, for the German language keyboard, enter the value **1252**. To use the default code page conversion, specify **0**. To view a list of languages and their code page numbers, select **Regional Options** in the Windows Control Panel and select the **Advanced** button in the General tab.

QuickTest uses this code page to correctly identify the keys you record.

HLLAPI Settings

The following options can be used to define configuration settings for emulators that support HLLAPI. These options are not available when an emulator is selected that does not support HLLAPI.

- ▶ **HLLAPI dll file.** QuickTest uses the HLLAPI DLL file specified for the selected emulator to connect to the emulator and to retrieve data concerning its current status.

If you are using a customized version of a preconfigured emulator, you may need to specify a different DLL file name.

- ▶ **HLLAPI function name.** The HLLAPI DLL for the selected emulator uses this function as the entry point for all HLLAPI calls.

If you are using a customized version of a preconfigured emulator, you may need to specify a different function name.

- ▶ **Argument size for HLLAPI function.** For most emulators, the HLLAPI function receives 16-bit (word) arguments. For some emulators, such as IBM PCOM, the HLLAPI function receives 32-bit (integer) arguments.

Select the correct argument size for the selected emulator: **word (16 bits)** or **integer (32 bits)**.

- ▶ **Execute HLLAPI calls from multiple threads.** Some emulators allow HLLAPI calls from multiple threads, while others require all HLLAPI calls to be executed from the same thread. For a preconfigured emulator configuration, this setting is selected by default.

Clear this check box to instruct QuickTest to open a separate process for HLLAPI calls and to execute all HLLAPI calls from this single thread.

- ▶ **Warn if HLLAPI dll file not found.** If this option is selected, QuickTest displays a warning message when the HLLAPI DLL file for the current configuration cannot be found. For example, QuickTest warns you if you try to use the Terminal Emulator Add-in before you have installed the emulator itself.

If you clear this check box and QuickTest cannot find the required DLL file, it may be difficult to determine why QuickTest is not recording successfully. It is therefore recommended that you leave this option selected.

Note: This setting applies to all terminal emulator configurations, regardless of the currently selected emulator.

Object Identification Settings

The following options can be used to configure the way QuickTest identifies objects for the selected terminal emulator:

- ▶ **Identify screen label using all characters.** The **label** property value is used to identify the TeScreen test object. The location and length of the label are defined for the selected emulator in the Terminal Emulator pane. For more information, see "Modifying Your Terminal Emulator Settings" on page 562.

By default, only the protected characters in the defined label area are captured for the **label** property value.

Select this option if you want QuickTest to capture all the characters in the label area for the **label** property, including any unprotected or hidden characters that may form part of the label.

- ▶ **Identify emulator window based on title bar prefix.** QuickTest normally identifies the emulator window by its object class. With a user-defined configuration, the class name may not be unique. For example, an emulator may use a generic class name, such as **Afx**. In such cases, you can instruct QuickTest to identify the window based on a static prefix in the window title bar.

To instruct QuickTest to use a prefix to identify the correct window, specify the text string for the prefix.

When no value is specified, QuickTest uses the object class to identify the emulator window.

Record Settings

The following options can be used to configure the way QuickTest records operations:

- **Communicate with server using these keys.** When recording without HLLAPI support, QuickTest inserts **Sync** steps after specified keys are pressed, to synchronize the communication between the emulator and the server. The keys are identified by their virtual key codes.

The default is the ENTER key—virtual key code value: **13 (0D Hex)**. You can specify different or additional keys. For example, you can add the CTRL key—virtual key code value: **17 (11 Hex)**.

Specify the **decimal** value of the virtual key code for each key, separated by a semicolon (;). QuickTest inserts a **Sync** step whenever one of these keys is pressed. For more information on synchronization, see "Synchronizing the Run Session" on page 587.

For a list of virtual key codes, see <http://msdn.microsoft.com/en-us/library/ms645540.aspx>. The list on the MSDN page displays the Hex values for each key code. You must convert the value to decimal and specify the decimal value of the key codes when you add them to the list for this option.

Note: This setting applies to all terminal emulator configurations, regardless of the currently selected emulator.

- **Record operations on menus and dialog boxes.** By default, QuickTest records operations on the menus of the terminal emulator window and the dialog boxes that open as a result of these menu option selections. For more information, see "TeWindow Object" on page 569.

Clear this check box if you do not want QuickTest to record these menu and dialog box operations. For example, you may not want emulator-specific menu and dialog box steps in your test or component if cross-emulator compatibility is important, or if these steps are not relevant to your test or component.

Note: This setting applies to all terminal emulator configurations, regardless of the currently selected emulator.

- **Record mode.** In **Text screen** mode, QuickTest records operations as TeTextScreen steps, based on the screen coordinates. In **Context-sensitive** mode, QuickTest records field operations as TeField steps. For more information on the TeField and TeTextScreen objects, see "Identifying Test Object Classes for Terminal Emulators" on page 569.

By default, all preconfigured terminal emulators and user-defined emulators configured as fully supporting HLLAPI are set to context-sensitive mode. Select **Text screen mode** if you are using an emulator that supports HLLAPI and you want to test in terms of coordinates instead of TeField objects.

You can use the wizard to change the mode for a user-defined terminal emulator. For more information, see "Using the Terminal Emulator Configuration Wizard" on page 536.

Note: For emulators that do not support HLLAPI and those configured as supporting text-only HLLAPI operations, QuickTest always uses **Text screen** mode and this option is not available.

- ▶ **Record steps without synchronizing.** By default, when QuickTest recognizes a user operation in the terminal emulator application, such as keyboard input or a mouse click, QuickTest suspends the processing of the user input in the application. After the recorded statement is added to the test or component script and the Active Screen information has been saved, QuickTest releases the emulator and allows it to process the user input.

Some emulators, such as IBM PCOM, do not support running HLLAPI while user input processing is suspended, and require QuickTest to release the emulator process before executing HLLAPI calls.

If you experience unexpected behavior while trying to record, you may need to select this option. For example, there may be no response from QuickTest or the emulator (or both). If you do select this option, make sure that you allow sufficient time for QuickTest to record each step before performing another operation.

- ▶ **Record cursor position.** When recording in a text screen or field, QuickTest uses `TeTextScreen.ClickPosition` or `TeField.SetCursorPos` to record the cursor position.

Clear this check box if you do not want to record the cursor position in your test or component.

Note: This setting applies to all terminal emulator configurations.

- ▶ **Trim trailing spaces in fields.** When recording in **Context-sensitive** mode, fields may contain trailing spaces or other "white characters", such as tab symbols.

Select this check box to instruct QuickTest to trim these characters. If you select this option, specify the minimum length of fields to be trimmed. Fields containing fewer than the specified number of characters remain unchanged. The default is **5** characters.

Clear the check box to leave the field content unchanged.

Note: This feature is available only for emulators that support HLLAPI.

- **Use property patterns.** Select this check box to use property patterns to record regular expressions in identification properties, such as for date and time values in a screen label.

For more information on property patterns, see *Using Property Patterns to Identify Objects (Advanced)* in **PropPattern.htm**. This file is located in the **help** subfolder of the QuickTest installation folder.

You can accept the default property pattern configuration file, change its contents, or specify a different property pattern configuration file. The default file is designed for applications where the current time forms part of the screen label. It defines regular expressions that replace the current time in the screen label, creating a reliable description and readable name for the screen.

Note: This setting applies to all terminal emulator configurations, regardless of the currently selected emulator.

Run Settings

The following options can be used to configure the way QuickTest runs tests or components for the selected terminal emulator, if the emulator supports HLLAPI:

- ▶ **Beep when Sync operations are performed.** Indicates whether QuickTest beeps after performing each **Sync** operation during a run session.

Note: This setting applies to all terminal emulator configurations, regardless of the currently selected emulator.

- ▶ **Run steps containing special emulator keys using keyboard events.** Instructs QuickTest to perform SendKey commands using keyboard events. If you do not use this option to specify key codes, QuickTest performs SendKey commands using the corresponding HLLAPI function.

Some emulators, for example, Attachmate Extra!, recognize the RESET command while the emulator is busy only when it is sent using keyboard events. In the **Keys for RESET function** option, specify the keyboard combination of the virtual key code by specifying the decimal value of each key in the code, separated by semicolons (;).

- ▶ **Time between emulator status checks (in milliseconds).** During a **Sync** step, QuickTest waits the specified period of time before checking the emulator status. QuickTest repeats these checks at the specified interval until the emulator status changes to **Ready** (or until the **Sync** timeout is reached), and then continues with the run session. For more information on synchronization, see "Synchronizing the Run Session" on page 587.

Specify the interval (in milliseconds) between emulator status checks. The default is **200**.

Note: Specifying a very long interval could significantly increase the time your tests or components take to run.

Part V

The Visual Basic Add-in

36

Using the Visual Basic Add-in

You can use the QuickTest Professional Visual Basic Add-in to test Visual Basic objects (controls).

For details on supported Visual Basic environments, see the **Visual Basic Add-in** section of the *HP QuickTest Professional Product Availability Matrix*, available from the Documentation Library Home page or the root folder of the QuickTest Professional DVD.

The Visual Basic Add-in provides test objects, methods, and properties that can be used when testing objects in Visual Basic applications. For more information, see the **Visual Basic** section of the *HP QuickTest Professional Object Model Reference*.

The following table summarizes basic information about the Visual Basic Add-in and how it relates to some commonly-used aspects of QuickTest.

General Information	
Add-in Type	This is a Windows-based add-in. Much of its functionality is the same as other Windows-based add-ins. See "Testing Windows-Based Applications" on page 87.
Checkpoints and Output Values	<ul style="list-style-type: none"> ▶ See the sections describing checkpoints and output values in the <i>HP QuickTest Professional User Guide</i> (for tests), and the <i>HP QuickTest Professional for Business Process Testing User Guide</i> (for components). ▶ See "Supported Checkpoints and Output Values Per Add-in" on page 769.
Prerequisites	
Opening Your Application	You can open your Visual Basic application before or after opening QuickTest.
Add-in Dependencies	None
Setting Preferences	
Options Dialog Box	Use the Windows Applications pane. (Tools > Options > Windows Applications node) See "The Options Dialog Box: Windows Applications Pane" on page 103.

<p>Record and Run Settings Dialog Box (tests only)</p>	<p>Use the Windows Applications tab. (Automation > Record and Run Settings)</p> <p>See "The Record and Run Settings Dialog Box: Windows Applications Tab" on page 90.</p> <p>Notes:</p> <ul style="list-style-type: none"> ▶ If you select the Record and Run only on radio button, the settings may also apply to (limit) the applications that are recognized for Object Spy and other pointing hand operations. ▶ QuickTest recognizes Visual Basic objects only in applications that are opened after changing the settings in the Windows Applications tab of the Record and Run Settings dialog box.
<p>Custom Active Screen Capture Settings Dialog Box (tests only)</p>	<p>Use the Windows applications section. (Tools > Options > Active Screen node > Custom Level)</p> <p>See the section on the Custom Active Screen Capture Settings dialog box in the <i>HP QuickTest Professional User Guide</i>.</p>
<p>Application Area Settings Dialog Box (components only)</p>	<p>Use the Applications pane. (File > Settings > Applications node)</p> <p>See the section on defining Application Settings for your application area in the <i>HP QuickTest Professional for Business Process Testing User Guide</i>.</p>

This chapter includes:

- ▶ Troubleshooting and Limitations - Visual Basic Add-in on page 612

Troubleshooting and Limitations - Visual Basic Add-in

This section describes troubleshooting and limitations for the Visual Basic Add-in.

- ▶ When working with the Visual Basic Add-in, it is recommended to select the **Record and run on these applications (opened on session start)** option and then to specify the application name in the Windows Applications tab of the Record and Run settings dialog box.

If you select the **Record and run test on any open Windows-based application** option, you should open the Visual Basic application after the first time you start recording.

- ▶ Combo box objects of style Simple ComboBox are not supported.

Part VI

The VisualAge Smalltalk Add-in

37

Using the VisualAge Smalltalk Add-in

You can use the QuickTest Professional VisualAge Smalltalk Add-in to test VisualAge Smalltalk objects (controls).

For details on supported VisualAge Smalltalk environments, see the **VisualAge Smalltalk Add-in** section of the *HP QuickTest Professional Product Availability Matrix*, available from the Documentation Library Home page or the root folder of the QuickTest Professional DVD.

The VisualAge Smalltalk Add-in uses a sub-set of the standard Windows test objects, methods, and properties, which can be used when testing objects in VisualAge Smalltalk applications. For more information, see the **VisualAge Smalltalk** section of the *HP QuickTest Professional Object Model Reference*.

The following table summarizes basic information about the VisualAge Smalltalk Add-in and how it relates to some commonly-used aspects of QuickTest.

General Information	
Add-in Type	This is a Windows-based add-in. Much of its functionality is the same as other Windows-based add-ins. See "Testing Windows-Based Applications" on page 87.
Checkpoints and Output Values	<ul style="list-style-type: none"> ▶ See the sections describing checkpoints and output values in the <i>HP QuickTest Professional User Guide</i> (for tests), and the <i>HP QuickTest Professional for Business Process Testing User Guide</i> (for components). ▶ See "Supported Checkpoints and Output Values Per Add-in" on page 769.
Prerequisites	
Opening Your Application	You can open your VisualAge Smalltalk application before or after opening QuickTest.
Add-in Dependencies	None
Other	You must configure your VisualAge Smalltalk environment by importing the qt-adapter.dat file and then recompiling your application. See "Configuring the VisualAge Smalltalk Add-in" on page 618.
Setting Preferences	
Options Dialog Box	Use the Windows Applications pane. (Tools > Options > Windows Applications node) See "The Options Dialog Box: Windows Applications Pane" on page 103.

<p>Record and Run Settings Dialog Box (tests only)</p>	<p>Use the Windows Applications tab. (Automation > Record and Run Settings)</p> <p>See "The Record and Run Settings Dialog Box: Windows Applications Tab" on page 90.</p> <p>Notes:</p> <ul style="list-style-type: none"> ▶ QuickTest can recognize only VisualAge applications that have been precompiled with the qt-adapter agent. For more information, see "Configuring the VisualAge Smalltalk Add-in" on page 618. ▶ The Record and Run only on radio button applies only to record and run sessions. QuickTest recognizes all VisualAge objects for Object Spy and pointing hand operations, regardless of the settings in the Record and Run Settings dialog box.
<p>Custom Active Screen Capture Settings Dialog Box (tests only)</p>	<p>Use the Windows applications section. (Tools > Options > Active Screen node > Custom Level)</p> <p>See the section on the Custom Active Screen Capture Settings dialog box in the <i>HP QuickTest Professional User Guide</i>.</p>
<p>Application Area Settings Dialog Box (components only)</p>	<p>Use the Applications pane. (File > Settings > Applications node)</p> <p>See the section on defining Application Settings for your application area in the <i>HP QuickTest Professional for Business Process Testing User Guide</i>.</p>

This chapter includes:

- ▶ Configuring the VisualAge Smalltalk Add-in on page 618

Configuring the VisualAge Smalltalk Add-in

Before you can use the VisualAge Smalltalk Add-in, you must import the **qt-adapter.dat** file to your VisualAge Smalltalk development environment. Then you must recompile your application to include the **qt-adapter** agent.

Note: The **qt-adapter** agent is similar to the agent provided with the WinRunner VisualAge Smalltalk Add-in. Therefore, some steps in the configuration procedure below include selecting values containing WinRunner or WR.

To configure your VisualAge Smalltalk environment:

- 1** Start VisualAge Smalltalk.
- 2** In the System Transcript window, select **Tools > Browse Configuration Maps**.
- 3** In the Configuration Maps Browser window, right-click the **Names** pane and select **Import**.
- 4** In the **Information Required** box, enter the IP address or host name of the server, or leave the text box blank to use the native (fileio) access. Click **OK**. The Selection Required dialog box opens.
- 5** In your file system, browse to the **<QuickTest installation folder>/dat** folder and select **qt-adapter.dat**.
- 6** In the Selection Required dialog box, do the following:
 - In the **Names** pane, select **WinRunner**.
 - In the **Versions** pane, select **WR Adapter 1.0.1**.
 - Click the **>>** button and click **OK**.

- 7** In the Configuration Maps Browser window, do the following:
 - In the **Names** pane, click **WinRunner**.
 - In the **Editions and Versions** pane, click **WR Adapter 1.0.1**. A list of available applications displays in the **Applications** pane.
 - Right-click the **Editions and Versions** pane and select **Load**.
- 8** To save your changes, select **File > Save Image**, or click **OK** in the Warning dialog box when closing the VisualAge Smalltalk application.
- 9** Recompile your VisualAge Smalltalk application with the **qt-adapter** agent.

You are now ready to create and run tests on VisualAge Smalltalk applications.

Part VII

The Web Add-in

38

Using the Web Add-in

You can use the Web Add-in to test HTML objects (controls).

For details on supported Web browsers and versions, see the *HP QuickTest Professional Product Availability Matrix*, available from the Documentation Library Home page or the root folder of the QuickTest Professional DVD.

The Web Add-in provides test objects, methods, and properties that can be used when testing objects in Web applications. For details, see the **Web** section of the *HP QuickTest Professional Object Model Reference*.

QuickTest also provides a set of add-ins that support testing specialized controls from a number of Web 2.0 toolkits using test object classes that were developed by HP using Web Add-in Extensibility (described on page 665). These add-ins are displayed as child nodes of the Web Add-in in the Add-in Manager. For details, see "Web 2.0 Toolkit Support" on page 659.

The following table summarizes basic information about the Web Add-in and how it relates to some commonly-used aspects of QuickTest. This information is also relevant for all child add-ins that extend the Web Add-in.

General Information	
Add-in Type	<p>Much of the functionality of this add-in is the same as other Web-based add-ins.</p> <ul style="list-style-type: none"> ▶ See "Testing Web-Based Applications" on page 45. ▶ See "Working with Web Browsers" on page 627.
Checkpoints and Output Values	<ul style="list-style-type: none"> ▶ See the sections describing checkpoints and output values in the <i>HP QuickTest Professional User Guide</i> (for tests), and the <i>HP QuickTest Professional for Business Process Testing User Guide</i> (for components). ▶ See "Checking Web Pages" on page 633. ▶ See "Supported Checkpoints and Output Values Per Add-in" on page 769.
Extending the Web Add-in	<p>Web Add-in Extensibility (described on page 665) enables you to develop support for testing third-party and custom Web controls that are not supported out-of-the-box by the QuickTest Professional Web Add-in.</p>
Other	<ul style="list-style-type: none"> ▶ When you load the Siebel Add-in in addition to the Web Add-in, the object identification settings are automatically customized. For this reason, the Web Add-in is not available in the Environment list in the Object Identification dialog box (Tools > Object Identification), even though the Web Add-in is loaded. For more information, see "Using the Siebel Add-in" on page 447. ▶ You can create steps on more than one browser tab, if your browser supports tabbed browsing.
Prerequisites	
Opening Your Application	<p>You must open QuickTest before opening your Web application.</p>
Add-in Dependencies	<p>None</p>

Setting Preferences	
Options Dialog Box	Use the Web pane. (Tools > Options > Web node) For more information, see "Setting Web Testing Options" on page 52.
Record and Run Settings Dialog Box (tests only)	Use the Web tab. (Automation > Record and Run Settings) See "Setting Web Record and Run Options" on page 46.
Test Settings Dialog Box (tests only)	Use the Web pane. (File > Settings > Web pane) See "Defining Web Settings for Your Test" on page 69.
Custom Active Screen Capture Settings Dialog Box (tests only)	Use the Web section. (Tools > Options > Active Screen node > Custom Level) See the section on the Custom Active Screen Capture Settings dialog box in the <i>HP QuickTest Professional User Guide</i> .
Application Area Settings Dialog Box (components only)	Use the Web pane. (File > Settings > Web node) <ul style="list-style-type: none"> ▶ See "Defining Web Settings for Your Application Area" on page 71. ▶ See the section on defining Application Settings for your application area in the <i>HP QuickTest Professional for Business Process Testing User Guide</i>.

This chapter includes:

- ▶ Considerations for Working with the Web Add-in on page 626
- ▶ Working with Web Browsers on page 627
- ▶ Checking Web Pages on page 633
- ▶ Accessibility Checkpoints - Checking Web Content Accessibility on page 646
- ▶ Accessing Password-Protected Resources in the Active Screen on page 650
- ▶ Activating Methods Associated with a Web Object on page 656

- ▶ Using Programmatic Descriptions for the WebElement Object on page 657
 - ▶ Registering Browser Controls on page 658
 - ▶ Web 2.0 Toolkit Support on page 659
 - ▶ Web Add-in Extensibility on page 665
 - ▶ Extensibility Accelerator for HP Functional Testing on page 666
- Troubleshooting and Limitations - Web Add-in** on page 667

Considerations for Working with the Web Add-in

- ▶ If QuickTest does not record Web events in a way that matches your needs, you can also configure the events you want to record for each type of Web object. For example, if you want to record events, such as moving the pointer over an object to open a sub-menu, you may need to modify the Web event configuration to recognize such events. For more information, see Chapter 39, "Configuring Web Event Recording for Web Objects."
- ▶ If you are recording on a list in an application, you must highlight 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). This is because QuickTest records a step only if the value in the list changes.
- ▶ Before you insert a checkpoint on a Web-based object while editing a test or component, make sure that one of the following is true:
 - ▶ The Web application is open, and the object you want to check is displayed.
 - ▶ The Active Screen for the currently selected step is open and contains the object you want to check (with the necessary level of information). For more information on Active Screen capture levels, see the section describing how to increase or decrease the Active Screen information saved with a test in the *HP QuickTest Professional User Guide*.

- If a Web element in an HTML page is set to be disabled or invisible, for example if a <DIV> element above it controls its appearance, but the elements on the page are available in the DOM, then QuickTest can perform operations on those objects even though a human user of the application could not.

Working with Web Browsers

QuickTest tests and components are generally cross-browser—you can record Web steps on Microsoft Internet Explorer or Mozilla Firefox, or you can create steps with the keyword-driven methodology using any supported browser. You can run Web steps in any supported browser. For specific items to consider, see the following sections. For information on supported browser versions, see the *HP QuickTest Professional Product Availability Matrix*, available from the Documentation Library Home page or the root folder of the QuickTest Professional DVD.

The following sections describe considerations and limitations for working with browsers:

- "General Considerations for All Web Browsers" on page 627
- "Microsoft Internet Explorer" on page 628
- "Mozilla Firefox" on page 629
- "Applications with Embedded Web Browser Controls" on page 631
- "Troubleshooting and Limitations for Working with Multiple Web Browsers" on page 632

General Considerations for All Web Browsers

- You select your browser in the Web tab of the Record and Run Settings dialog box. For more information, see "Setting Web Record and Run Options" on page 46.
- QuickTest does not support the option to zoom in and out of a Web page. If you use this option, some QuickTest functionality may not work as expected. For example, the Object Spy may be unable to correctly highlight objects or display object details. (These problems do not occur if the **Zoom Text Only** Firefox menu item is selected.)

Additionally, bitmap checkpoints will fail if a different zoom level is used when capturing the expected bitmap than the zoom level used when running the checkpoint step.

- ▶ By default, the name assigned to the Browser test object in the object repository is always the name assigned to the first Page object that is learned or recorded for the Browser object. The same Browser test object is used each time you learn an object or record in a browser with the same ordinal ID. Therefore, the name used for the Browser test object in the steps you record may not reflect the actual browser name.

Microsoft Internet Explorer

- ▶ QuickTest Professional Web support behaves as a browser extension in Microsoft Internet Explorer. Therefore, you cannot use the Web Add-in on Microsoft Internet Explorer without selecting the **Enable third-party browser extensions** option. To set the option, in Microsoft Internet Explorer select **Tools > Internet Options > Advanced** and select the **Enable third-party browser extensions** option.
- ▶ Creating and running steps that start an InPrivate Browsing session is supported only by using **Tools > InPrivate Browsing**. Using toolbars or extensions for this operation may cause Microsoft Internet Explorer to behave unexpectedly.
- ▶ Creating and running steps that are related to tabs, such as selecting a tab or creating a new tab is not supported when Microsoft Internet Explorer is in Full Screen mode.

Workaround: Add a **<Browser>.FullScreen** step before and after the desired step to toggle Full Screen mode.

Mozilla Firefox

Consider the following when using Mozilla Firefox:

- ▶ When working in operating systems with UAC (User Accounts Control), you must be logged-in with Administrator privileges (or have write permissions to the browser's installation folder) after installing Mozilla Firefox, since QuickTest needs to register support for Mozilla Firefox by adding a file to the browser's installation folder.
- ▶ Generally, steps that were recorded on Microsoft Internet Explorer will run on Mozilla Firefox without requiring any modification. However, there are several differences to consider:
 - ▶ QuickTest does not support Mozilla Firefox menus or sidebars.
 - ▶ QuickTest supports specific Browser menu operations that are represented by the following toolbar buttons:
 - ▶ **Back**
 - ▶ **Forward**
 - ▶ **Home**
 - ▶ **Refresh**
 - ▶ **Stop**

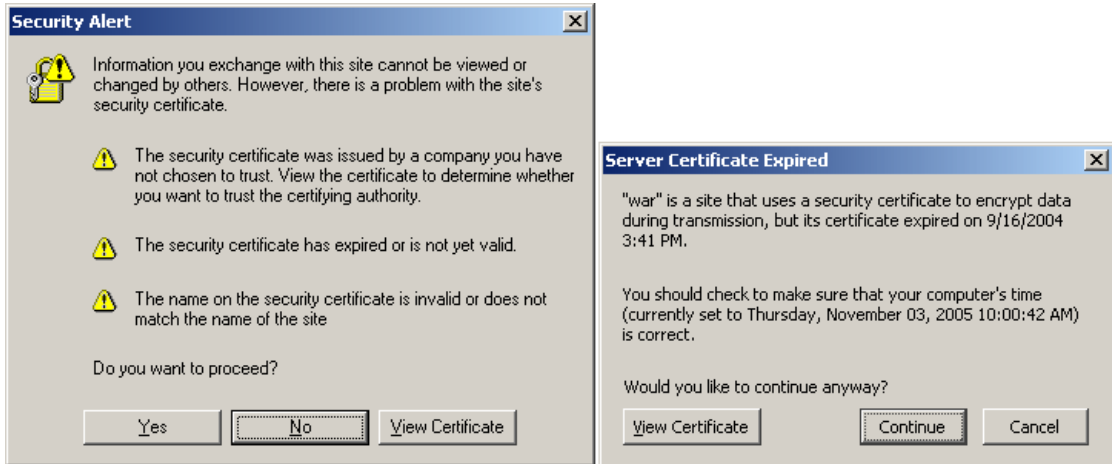
All other toolbars and toolbar buttons are not supported. If you record steps on any unsupported menu or toolbar objects when working with Microsoft Internet Explorer, you may need to remove or replace the steps before running the test or component on Mozilla Firefox.

- ▶ The following checkpoints and output values may be affected by cross-browser QuickTest operations:
 - ▶ Standard or page checkpoints for links and images that are created on Internet Explorer using Record, or using the Active Screen, may not pass when run using Mozilla-based browsers, even if the checkpoints pass when the test is run using Internet Explorer.
 - ▶ Standard checkpoints for links and images created on Active Screen captures that were captured from a Mozilla-based browser, may not pass when run using Internet Explorer, even if the checkpoints pass when the test is run using the Mozilla-based browser.

You can use regular expressions if you want to create checkpoints for links and images that run on both Internet Explorer and Mozilla-based browsers. For more information on regular expressions, see the section on understanding and using regular expressions in the *HP QuickTest Professional User Guide*.

- ▶ Standard checkpoints that use the **inner_html** property may not pass when run using Mozilla-based browsers, because blanks, slashes, backslashes or other special characters are handled differently in the different browser types.
- ▶ Before running text/text area checkpoint or output value steps, you must set the text recognition options to use only OCR, by selecting the **Use Only OCR** option in the **General > Text Recognition** pane of the Options dialog box.
- ▶ Due to the difference in standard dialog boxes, pop-up recovery scenarios that use the **Click button with label** recovery operation and were built for Microsoft Internet Explorer will not work for Mozilla Firefox, and vice versa.
- ▶ Mozilla Firefox uses different standard dialog boxes than the Windows standard dialog boxes used by Microsoft Internet Explorer. If you create steps on such dialog boxes, you should create additional steps to be used when running on Mozilla Firefox, and precede them with an **If** statement to check which browser is running.

For example, the following two dialog boxes are a security alert of the same Web site, the one on the left is from Microsoft Internet Explorer and the one on the right is from Mozilla Firefox. Although they both look like a Windows dialog box, the Mozilla Firefox one is actually a browser window.



Applications with Embedded Web Browser Controls

Working with applications that contain embedded Web browser controls is similar to working with Web objects in a Web browser.

Note: Embedded browser controls are supported only for Microsoft Internet Explorer.

To test objects in embedded browser controls, ensure that:

- ▶ The Web Add-in is loaded.
- ▶ The application opens only after QuickTest is open.
- ▶ (For tests) In the Web tab of the Record and Run Settings dialog box, the **Record and run test on any open browser** option is selected. (This option is not relevant for components.)

After these conditions are met, you can start adding steps or running your test or component.

Troubleshooting and Limitations for Working with Multiple Web Browsers

Problem

When running steps that are intended to be performed on different browsers, and QuickTest tries to perform the step intended for the second browser before the second browser has finished loading, QuickTest will perform the step on the first browser, and the step may fail.

Solution

Insert a **Wait()** statement before the first step on the second browser to enable the second browser to finish loading.

Reason

By default, a Browser test object does not have any identification properties in its description. When only one browser is open, the open browser matches the (empty) description for any Browser test objects. When multiple browsers are open, QuickTest uses smart identification or the ordinal identifier property value stored with the relevant Browser test object to distinguish between the browsers and to select the correct browser.

However, if a second browser has not fully loaded when QuickTest tries to perform a step intended for that browser, QuickTest will assume that only one browser is open and it will try to perform the step on the first browser without reverting to smart identification or ordinal identifiers.

Checking Web Pages

When working with tests, you can check statistical information about your Web pages by adding page checkpoints to your test. These checkpoints check the links and the sources of the images on a Web page. You can also instruct page checkpoints to include a check for broken links.

Page checkpoints are not supported for business components.

This section includes:

- ▶ "Automatic Page Checkpoints" on page 633
- ▶ "Creating Individual Page Checkpoints" on page 634
- ▶ "Understanding the Page Checkpoint Properties Dialog Box" on page 636
- ▶ "Filtering Hypertext Links" on page 642
- ▶ "Filtering Image Sources" on page 644

Automatic Page Checkpoints

You can instruct QuickTest to create automatic page checkpoints for every page in all tests by selecting the **Create a checkpoint for each Web page while recording** check box in the Web > Advanced pane (click the **Advanced** node under the Web node of the Options dialog box). By default, the automatic page checkpoint includes the checks that you select from among the available options in the Web > Advanced pane.


You can also instruct QuickTest not to perform automatic page checkpoints when you run your test by selecting the **Ignore automatic checkpoints while running tests** check box in the Web > Advanced pane of the Options dialog box.

For more information, see "Setting Web Testing Options" on page 52.

Creating Individual Page Checkpoints



You can manually add a page checkpoint to your test to check the links and the image sources on a selected Web page either while recording or editing your test.

To add a page checkpoint while recording:

- 1 Navigate to the page to which you want to add a checkpoint.
- 2  Select **Insert > Checkpoint > Standard Checkpoint**, or click the **Insert Checkpoint or Output Value** button in the **Insert** toolbar and select **Standard Checkpoint**.

The QuickTest window is minimized and the pointer turns into a pointing hand.
- 3 Click in the page you want to check. The Object Selection dialog box opens. For more information, see the *HP QuickTest Professional User Guide*.
- 4 Select the **Page** item and click **OK**. The Page Checkpoint Properties dialog box opens.
- 5 Modify the settings for the checkpoint in the Page Checkpoint Properties dialog box, as described in "Understanding the Page Checkpoint Properties Dialog Box" on page 636.
- 6 Click **OK** to close the dialog box. A checkpoint step is added to your test.

To add a page checkpoint while editing your test:

- 1  Make sure the **Active Screen** button is selected.
- 2 Click a step in your test where you want to add a checkpoint. The Active Screen displays the Web page or part of the Web page corresponding to the highlighted step.
- 3 Right-click anywhere on the Active Screen and select **Insert Standard Checkpoint**. The Object Selection dialog box opens.
- 4  Select the **Page** item you want to check from the displayed object tree.

- 5 Click **OK**. The Page Checkpoint Properties dialog box opens.

Note: You can also right-click a **Page** item in the Keyword View and select **Insert Standard Checkpoint** to open the Page Checkpoint Properties dialog box.

- 6 Specify the settings for the checkpoint. For more information, see "Understanding the Page Checkpoint Properties Dialog Box" on page 636.
- 7 Click **OK** to close the dialog box. A checkpoint step is added to your test.

Note: You cannot select the **HTML Verification** options while creating a page checkpoint from the Keyword View or Active Screen. You can select these options only when creating a Page checkpoint while recording.

Understanding the Page Checkpoint Properties Dialog Box

The Page Checkpoint Properties dialog box enables you to choose which properties to check.

The ABC icon indicates that the value of the property to check is a constant.

The selected check box indicates that this property will be checked.

This icon indicates that the value of the property to check is a Data Table parameter.

Page Checkpoint Properties

Name: Find a Flight: Mercury
Class: Page

Type	Property	Value
<input checked="" type="checkbox"/> ABC	load time	0
<input checked="" type="checkbox"/> ABC	number of images	13
<input checked="" type="checkbox"/> DT	number of links	<Find_a_Flight_Mercury_n

Configure value

Constant 12

Parameter
DataTable("Find_a_Flight_Mercury_number_of_links")

HTML verification

HTML source Edit HTML Source...

HTML tags Edit HTML Tags...

All objects in page

Links Filter Link Check...

Images Filter Image Check...

Broken links


Checkpoint timeout: 0 seconds

Insert statement: Before current step After current step

OK Cancel Help





Identifying the Object

The top part of the dialog box displays information on the checkpoint:

Information	Description
Name	<p>The name that QuickTest assigns to the checkpoint. By default, the checkpoint name is the title of the Web page on which the checkpoint is being performed, as defined in the HTML code. You can specify a different name for the checkpoint or accept the default name.</p> <p>If you rename the checkpoint, make sure that the name:</p> <ul style="list-style-type: none"> ▶ is unique ▶ does not begin or end with a space ▶ does not contain " (double quotation mark) ▶ does not contain the following character combinations: <ul style="list-style-type: none"> := @@
Class	<p>The type of object. This is always Page.</p>
Find in Repository button  (Located to the right of the Name box)	<p>Displays the checkpoint in its repository.</p> <p>Note: This option is not available when creating a new checkpoint. It is available only when editing an existing checkpoint.</p>

Choosing Which Property to Check

The default properties for the object are listed in the Properties pane of the dialog box. The pane includes the properties, their values, and their types:

Pane Element	Description
Check box	<p>For each object class, QuickTest recommends default property checks. You can accept the default checks or modify them accordingly.</p> <p>To check a property, select the corresponding check box.</p> <p>To exclude a property check, clear the corresponding check box.</p>
Type	<p>The  icon indicates that the value of the property is currently a constant.</p> <p>The  icon indicates that the value of the property is currently a test 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>
Property	The name of the property.
Value	<p>The value of the property. Note that the value in the page will be the expected value of the property when you run your test unless you edit this value. For information on editing the value of a property, see the section on configuring values in the <i>HP QuickTest Professional User Guide</i>.</p>

Note: By default, page checkpoints include a check on the page load time. The load time displayed in the Page Checkpoint Properties dialog box is the amount of time it took the page to load during recording. To add to the time that QuickTest allows for pages to load without causing page checkpoints to fail, increase the value of the **Add seconds to page load time** option in the Web pane of the Options dialog. For more information, see the section on setting global testing options in the *HP QuickTest Professional User Guide*.

Configuring the Value of a Page Property

In the **Configure value** area, you can define the expected value of the property to check as a **Constant** or **Parameter**. For information on modifying property values, see the section on setting values in the Configure Value area in the *HP QuickTest Professional User Guide*.

Checking the HTML Verification

In the HTML verification area, you can use the following options to check the HTML source and tags of the page:

Option	Description
HTML source	Checks that the source in the Web page being tested matches the expected HTML code (the source code of the page at the time that the test is recorded). Available only when creating a page checkpoint while recording.
Edit HTML Source (enabled only when the HTML Source check box is selected)	<p>Opens the HTML Source dialog box, which displays the expected HTML code. Edit the expected HTML source code and click OK. Note that you can also use regular expressions when editing the expected HTML source code if you click the regular expression check box at the bottom of the page. For more information on regular expressions, see the section on understanding and using regular expressions in the <i>HP QuickTest Professional User Guide</i>.</p> <p>You can search and replace text strings in the HTML Source dialog box by right-clicking and choosing Find or Replace. For more information on the Find dialog box, see the section on finding text strings in the <i>HP QuickTest Professional User Guide</i>. For more information on the Replace dialog box, see the section on replacing text strings in the <i>HP QuickTest Professional User Guide</i>.</p>
HTML tags	Checks that the HTML tags in the Web page being tested match the expected HTML tags (the HTML tags on the page at the time that the test is recorded). Available only when creating a page checkpoint while recording.

Option	Description
<p>Edit HTML Tags (enabled only when the HTML Tags check box is selected)</p>	<p>Opens the dialog box that displays the expected HTML tags. Edit the expected HTML tags and click OK. Note that you can also use regular expressions when editing the HTML tags if you click the regular expression check box at the bottom of the page. For more information on regular expressions, see the section on understanding and using regular expressions in the <i>HP QuickTest Professional User Guide</i>.</p> <p>You can search and replace text strings in the Edit HTML Tags dialog box by right-clicking and choosing Find or Replace. For more information on the Find dialog box, see the section on finding text strings in the <i>HP QuickTest Professional User Guide</i>. For more information on the Replace dialog box, see the section on replacing text strings in the <i>HP QuickTest Professional User Guide</i>.</p>

Checking All the Objects in a Page

In the **All objects in page** area, you can check all the links, images, and broken links in a page. You can use the following options to check the objects in a page:

Option	Description
<p>Links</p>	<p>Checks the functionality of the links in the page according to your selections in the Filter Link Check dialog box.</p>
<p>Filter Link Check (enabled only when the Links check box is selected)</p>	<p>Opens the Filter Link Check dialog box, which enables you to specify which hypertext links to check in the page. For more information, see "Filtering Hypertext Links" on page 642.</p>
<p>Images</p>	<p>Checks that the images are displayed on the page according to your selections in the Filter Images Check dialog box.</p>

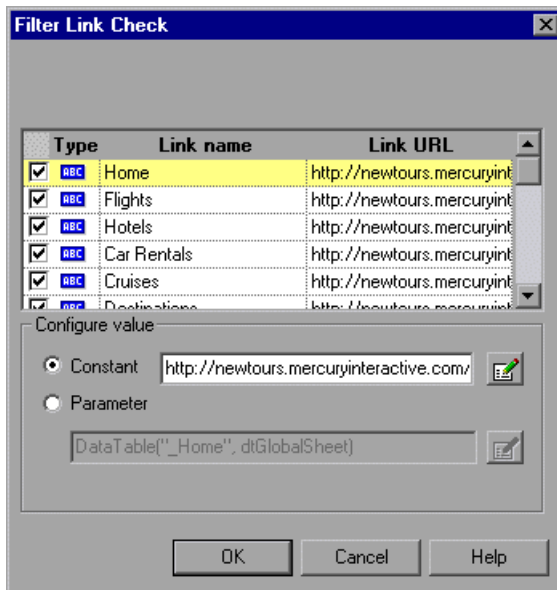
Option	Description
Filter Image Check (enabled only when the Images check box is selected)	Opens the Filter Image Check dialog box, which enables you to specify which image sources to check in the page. For more information, see "Filtering Image Sources" on page 644.
Broken links	Instructs QuickTest to check for broken links. Note that if you want to check only links that are targeted to your current host, you should select the Broken links - checks only links to current host option in the Web pane of the Options dialog box. For more information, see the section on setting Web testing options in the <i>HP QuickTest Professional User Guide</i> .

Note: The **Insert statement** option is not available when adding a page checkpoint during recording or when modifying an existing page checkpoint. It is available only when adding a new page checkpoint to an existing test while editing your test.

Filtering Hypertext Links








You can filter which hypertext links to check in a page checkpoint using the Filter Link Check dialog box. You open this dialog box by clicking **Filter Link Check** in the Page Checkpoint Properties dialog box. If you select the **Links** check box in the Page Checkpoints Properties dialog box, then by default all the links on the page are selected. To instruct QuickTest not to check a particular hypertext link, clear the link's check box.

For more information, see "Checking All the Objects in a Page" on page 640.



Choosing Which Hypertext Links to Check

You can use the following options to choose which hypertext links to check in a page checkpoint:

Pane Element	Description
Check box	<p>Each link on the page has a corresponding check box.</p> <p>To check a link, select the corresponding check box (by default all links are selected).</p> <p>To exclude a link from the page checkpoint, clear the corresponding check box.</p>
Type	<p>The  icon indicates that the target URL is currently a constant.</p> <p>The  icon indicates that the value of the property is currently a Data Table parameter.</p> <p>The  icon indicates that the value of the property is currently an environment variable parameter.</p> <p>The  icon indicates that the value of the property is currently a random number parameter.</p> <p>The  icon indicates that the value of the property is currently a test or action parameter. (Relevant only for tests)</p> <p>The  icon indicates that the value of the property is currently a component parameter. (Relevant only for components)</p> <p>The  icon indicates that the value of the property is currently a local parameter. (Relevant only for components)</p>
Link name	The text in the hypertext link.
Link URL	The target URL.

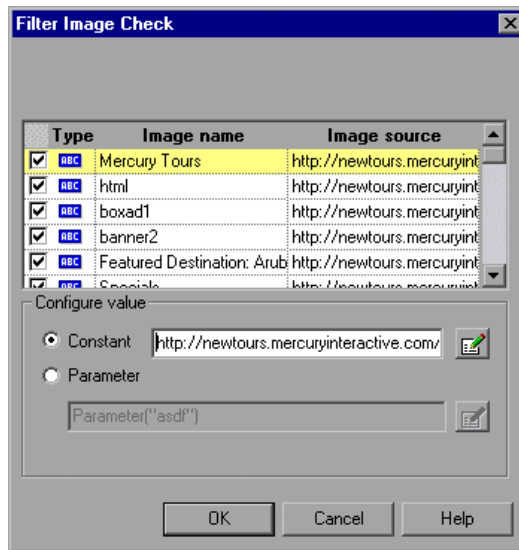
Configuring the Value of the Target URL

In the **Configure value** area, you can define the expected value of the target URL to which the hypertext links as a **Constant** or **Parameter**. For information on modifying values, see the section on setting values in the Configure Value area in the *HP QuickTest Professional User Guide*.

Filtering Image Sources

You can filter which image sources to check in a page checkpoint using the Filter Images Check dialog box. You open this dialog box by selecting **Filter Image Check** in the Page Checkpoint Properties dialog box. If you select the **Images** check box in the Page Checkpoints Properties dialog box, then, by default, all image sources on the page are selected. To instruct QuickTest not to check a particular image source, clear the image's check box.








For more information, see "Checking All the Objects in a Page" on page 640.



Choosing Which Image Sources to Check

You can use the following options to choose which image sources to check in a page checkpoint:

Pane Element	Description
Check box	<p>Each image source on the page has a corresponding check box.</p> <p>To check an image source, select the corresponding check box (by default all image sources are selected).</p> <p>To exclude an image source from the page checkpoint, clear the corresponding check box.</p>

Pane Element	Description
Type	<p>The  icon indicates that the image source is currently a constant.</p> <p>The  icon indicates that the value of the property is currently a Data Table parameter.</p> <p>The  icon indicates that the value of the property is currently an environment variable parameter.</p> <p>The  icon indicates that the value of the property is currently a random number parameter.</p> <p>The  icon indicates that the value of the property is currently a test or action parameter. (Relevant only for tests)</p> <p>The  icon indicates that the value of the property is currently a component parameter. (Relevant only for components)</p> <p>The  icon indicates that the value of the property is currently a local parameter. (Relevant only for components)</p>
Image name	The name of the image.
Image source	The image source file and path.

Configuring the Value of the Path of the Image Source File

In the **Configure value** area, you can define the path of the image source file as a **Constant** or **Parameter**. For information on modifying values, see the section on setting values in the Configure Value area in the *HP QuickTest Professional User Guide*.

Accessibility Checkpoints - Checking Web Content Accessibility

The Section 508 criteria for Web-based technology and information systems are based on access guidelines developed by the Web Accessibility Initiative of the World Wide Web Consortium (W3C). You can add accessibility checkpoints to help you quickly identify areas of your Web site that may not conform to the W3C Web Content Accessibility Guidelines. You can add automatic accessibility checkpoints to each page in your test, or you can add individual accessibility checkpoints to individual pages or frames.

Accessibility checkpoints are designed to help you easily locate the areas of your Web site that require special attention according to the W3C Web Content Accessibility Guidelines. They do not necessarily indicate whether or not your Web site conforms to the guidelines.

Accessibility checkpoints are not supported for business components.

This section includes:

- ▶ "Setting Accessibility Checkpoint Preferences" on page 647
- ▶ "Automatic Accessibility Checkpoints" on page 647
- ▶ "Creating Individual Accessibility Checkpoints" on page 647
- ▶ "Reviewing Accessibility Checkpoint Results" on page 650
- ▶ "Troubleshooting and Limitations for Accessibility Checkpoints" on page 650

Setting Accessibility Checkpoint Preferences

You can set accessibility checkpoint preferences in the **Web > Advanced** pane of the Options dialog box (**Tools > Options > Web node > Advanced node**) and view them in the Accessibility Checkpoint Properties dialog box. All accessibility checkpoints in your test use the options that are selected in the Advanced Web Options dialog box at the time of the run session. For information on the accessibility checkpoint options, see "Accessibility Checkpoint Options in Tests" on page 63.

Automatic Accessibility Checkpoints

You can instruct QuickTest to create automatic accessibility checkpoints for every page in all tests by selecting the **Add Automatic accessibility checkpoint to each Web page while recording** check box in the **Web > Advanced** pane of the Options dialog box (**Tools > Options > Web node > Advanced node**). If you select this option, an accessibility checkpoint is inserted for each page as you record.

Creating Individual Accessibility Checkpoints

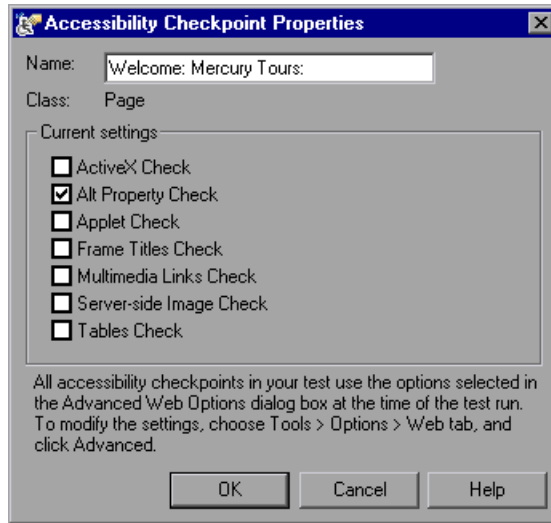
If you did not choose to add accessibility checkpoints automatically while recording, you can add an accessibility checkpoint to help you quickly identify areas of a particular Web page or frame that may not conform to the W3C Web Content Accessibility Guidelines. You can add accessibility checkpoints either while recording or editing your test.

To add an accessibility checkpoint while recording:

- 1 Navigate to a page where you want to add an accessibility checkpoint.
- 2 Select **Insert > Checkpoint > Accessibility Checkpoint**, or click the **Insert Checkpoint or Output Value** button and select **Accessibility Checkpoint**.
- 3 Click in the page or frame you want to check:
 - If the page contains frames, the Object Selection dialog box opens. Select the **Page** or **Frame** item you want to check and click **OK**. The Accessibility Checkpoint Properties dialog box opens.



- ▶ If the page does not contain frames, the Accessibility Checkpoint Properties dialog box opens. The dialog box displays the object's name, class (this is always Page or Frame), and the options that are currently selected. You can modify the option settings in the Web > Advanced pane of the Options dialog box (**Tools > Options > Web node > Advanced node**). For more information, see "Accessibility Checkpoint Options in Tests" on page 63.



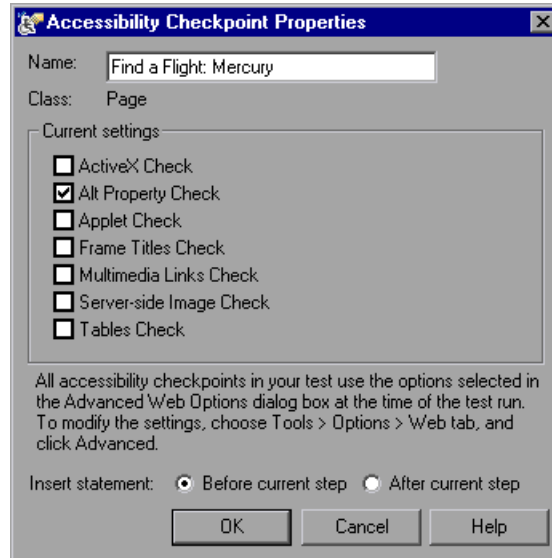
- 4 Click **OK** to close the dialog box. A checkpoint step is added to your test.

To add an accessibility checkpoint while editing your test:



- 1 Make sure the **Active Screen** button is selected.
- 2 Click a step in your test where you want to add a checkpoint. The Active Screen displays the Web page or part of the Web page corresponding to the highlighted step.
- 3 Right-click anywhere on the Active Screen, and select **Insert Accessibility Checkpoint**.
 - ▶ If the page contains frames, the Object Selection dialog box opens. Select the **Page** or **Frame** item and click **OK**. The Accessibility Checkpoint Properties dialog box opens.

- If the page does not contain frames, the Accessibility Checkpoint Properties dialog box opens. The dialog box displays the options that are currently selected. You can modify the option settings in the Web > Advanced pane of the Options dialog box (**Tools > Options > Web node > Advanced node**). For more information, see the section on advanced Web options in the *HP QuickTest Professional User Guide*.



Select **Before current step** if you want to check the accessibility elements before the highlighted step is performed. Select **After current step** if you want to check the accessibility elements after the highlighted step is performed.

Note: The **Insert statement** option is not available when adding a page checkpoint during recording or when modifying an existing page checkpoint. It is available only when adding a new page checkpoint to an existing test while editing your test.

- 4 Click **OK** to close the dialog box. A checkpoint step is added to your test.

Reviewing Accessibility Checkpoint Results

When you include accessibility checkpoints in your test, the Run Results window displays the results of each accessibility option that you checked.

The Run Results window displays a separate step for each accessibility option that was checked in each checkpoint. The results details provide information that can help you pinpoint parts of your Web site or application that may not conform to the W3C Web Content Accessibility Guidelines. The information provided for each check is based on the W3C requirements.

For more information on accessibility checkpoint results, see the section on analyzing accessibility checkpoint results in the *HP QuickTest Professional User Guide*.

Troubleshooting and Limitations for Accessibility Checkpoints

In Quality Center, you can view a comparison of accessibility checkpoints in the Asset Comparison Tool only if both QuickTest and the QuickTest Professional Add-in for ALM/QC are installed on the Quality Center computer.

Accessing Password-Protected Resources in the Active Screen

When QuickTest creates an Active Screen page for a Web-based application, it stores the path to images and other resources on the page, rather than downloading and storing the images with your test.

Note: The Active Screen pane is not available when working with business components. Therefore, this section is not relevant for business components.

Storing the path to images and other resources ensures that the disk space used by the Active Screen pages captured with your test is not affected by the file size of the resources displayed on the page.

For this reason, a page in the Active Screen (or in your run results) may require a user name and password to access certain images or other resources within the page. If this is the case, a pop-up login window may open when you select a step corresponding to the page, or you may note that images or other resources are missing from the page.

For example, the formatting of your page may look very different from the actual page on your Web site if the cascading style sheet (CSS) referenced in the page is password-protected, and therefore could not be downloaded to the Active Screen.

You may need to use one or both of the following methods to access your password-protected resources, depending on the password-protection mechanism used by your Web server:

- ▶ **Standard Authentication.** If your server uses a standard authentication mechanism, you can enter the login information in the Web pane of the Test Settings dialog box. QuickTest saves this information with your test and automatically enters the login information each time you select to display an Active Screen page that requires the information. For more information, see "Using the Standard Authentication Mechanism" on page 652.
- ▶ **Advanced Authentication.** If your server uses a more complex authentication mechanism, you may need to log in to the Web site manually using the Advanced Authentication dialog box. This gives the Active Screen access to password-protected resources in your Active Screen pages for the duration of your QuickTest session. When using this method, you must log in to your Web site in the Advanced Authentication dialog box each time you open the test in a new QuickTest session.

In most cases, the automatic login is sufficient. In some cases, you must use the manual login method. In rare cases, you may need to use both login mechanisms to enable access to all resources in your Active Screen pages. For more information, see "Using the Advanced Authentication Mechanism" on page 654.

Note: If your Web site is not password-protected, but you are still unable to view images or other resources on your Active Screen, you may not be connected to the Internet, the Web server may be down, or the source path that was captured with the Active Screen page may no longer be accurate.

Using the Standard Authentication Mechanism

If you select a step in your test or results, and an Active Screen login window opens over the Active Screen or results details display, then one or more images or other resources in the Active Screen may be password-protected.

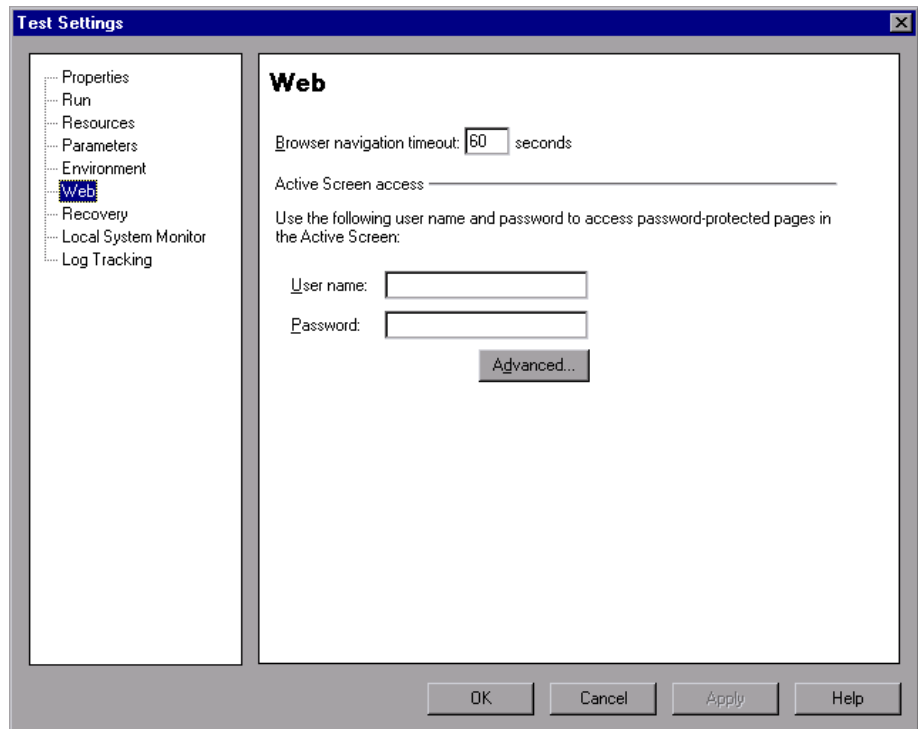


To prevent the pop-up login window from opening and to ensure that all images and resources are displayed in the Active Screen and results each time you open the test, you can use the automatic Active Screen login mechanism.

To enable the mechanism, you can select **Save the User Name and Password** in the pop-up login window the first time it opens. This adds the login information to the **Active Screen access** area in the Web pane of the Test Settings dialog box. Alternatively, you can add the login information manually in the Web pane of the Test Settings dialog box.

To set Active Screen access information in the Test Settings dialog box:

- 1** Select **File > Settings**. The Test Settings dialog box opens.
- 2** Click the **Web** node.



- 3** Enter the **User name** and **Password** for the Web site or Web page containing the password-protected resources.
- 4** Click **OK** to save your changes and close the dialog box.



- 5 Refresh the Active Screen by selecting a new step in the Keyword View or toggle the **Active Screen** button to redisplay the Active Screen. Confirm that the page is displayed correctly.

If one or more resources are still missing or displayed incorrectly, you may need to use the Advanced Authentication mechanism. For more information, see "Using the Advanced Authentication Mechanism" on page 654.

For more information on the Web pane of the Test Settings dialog box, see "Defining Web Settings for Your Test" on page 69.

Using the Advanced Authentication Mechanism

Depending on the authentication mechanisms used to password-protect resources on a Web site, the automatic Active Screen login mechanism may not be sufficient.

To enable the Active Screen to access the resources on such a site, you must log in to your site using the Advanced Authentication dialog box. When you log in this way, you remain logged in to the site for the duration of the QuickTest session. If you close and reopen QuickTest and then reopen your test, you must log in again.

Note: If the site to which you log in has an inactivity timeout after which you are automatically logged out of the Web site, you may need to log in using the Advanced Authentication dialog box more than once while editing your test to re-enable access to your Active Screen pages.

To log in to your Web site using the advanced authentication mechanism:


- 1 Select **File > Settings**. The Test Settings dialog box opens.
- 2 Click the **Web** node.

- 3 Click the **Advanced** button. The Advanced Authentication dialog box opens.



The browser window in the dialog box displays the default Web page for the test according to the following guidelines:

- ▶ The first time you open this dialog box for a given test, the browser window displays the URL address set for the test in the Web tab of the Record and Run Settings dialog box.
 - ▶ If you navigate to a new URL address using this dialog box, that address becomes the default Advanced Authentication page for this test.
- 4 If the displayed Web page is not the correct page for logging in to your site, enter the correct URL address in the **Address** box and click **Go**. Otherwise, proceed to step 5.

- 5 Enter your login information in the page displayed in the Advanced Authentication browser window.
- 6 When the login process is complete, click **Close**. The Advanced Authentication dialog box closes, but the login session remains open for the remainder of your QuickTest session (or until the Web site's inactivity timeout is exceeded).
-  7 Refresh the Active Screen by selecting a new step in the Keyword View or toggle the **Active Screen** button to redisplay the Active Screen. Confirm that the pages are displayed correctly.

If you still cannot view images or other resources on your Active Screen, you may not be connected to the Internet, the Web server may be down, or the source path that was captured with the Active Screen page may no longer be accurate.

Activating Methods Associated with a Web Object

In the Expert View, you can use the **Object** property to activate the method for a Web object. Activating the method for a Web object has the following syntax:

```
WebObjectName.Object.Method_to_activate( )
```

For example, suppose you have the following statement in your script:
document.MyForm.MyHiddenField.value = "My New Text"

The following example achieves the same thing by using the **Object** property, where **MyDoc** is the DOM's document:

```
Dim MyDoc  
Set MyDoc = Browser(browser_name).page(page_name).Object  
MyDoc.MyForm.MyHiddenField.value = "My New Text"
```


In the following example, LinksCollection is assigned to the link collection of the page through the Object property. Then, a message box opens for each of the links, with its innerHTML text.

```
Dim LinksCollection, link
Set LinksCollection = Browser(browser_name).Page(page_name).Object.links
For Each link in LinksCollection
    MsgBox link.innerHTML
Next
```

For more information on the **Object** property (**.Object**), see the section on retrieving and setting identification property values in the *HP QuickTest Professional User Guide*.

For a list of a Web object's internal properties and methods, see:

<http://msdn2.microsoft.com/en-us/library/ms531073.aspx>

Using Programmatic Descriptions for the WebElement Object

When QuickTest recognizes an object as a Web object that does not fit into any other HP QuickTest test object class, it learns the object as a WebElement object. You can also use a programmatic description with a WebElement test object to perform methods on any Web object in your Web site.

For example, when you run either of the examples below, QuickTest clicks the first Web object in the Mercury Tours page with the name UserName.

```
Browser("Mercury Tours").Page("Mercury Tours").
    WebElement("Name:=UserName", "Index:=0").Click
or
```

```
set WebObjDesc = Description.Create()
WebObjDesc("Name").Value = "UserName"
WebObjDesc("Index").Value = "0"
Browser("Mercury Tours").Page("Mercury Tours").WebElement(WebObjDesc).
    Click
```

For more information on the WebElement object, see the *HP QuickTest Professional Object Model Reference*. For more information on programmatic descriptions, see the section on programmatic descriptions in the *HP QuickTest Professional User Guide*.

Registering Browser Controls

A browser control adds navigation, document viewing, data download, and other browser functionality to a non-Web application. This enables the user to browse the Internet as well as local and network folders from within the application.

QuickTest Professional cannot automatically recognize the objects that provide browser functionality in your non-Web application as Web objects. For QuickTest to record or run on these objects, the application hosting the browser control must be registered.

Note: You can register applications developed in different environments, such as those written in Java, .NET, and so on.

You use the Register Browser Control utility to define the path of your Web application hosting the browser control. After registration, QuickTest will recognize Web objects in your application when recording or running tests.



To open the Register New Browser Control utility, select **Start > Programs > QuickTest Professional > Tools > Register New Browser Control**.

Enter the absolute path to the `.exe` file of the application hosting the browser control, and click **Register**. To remove a registered application, enter the absolute path and click **Unregister**.

After you register an application hosting a browser control using this utility, you must restart QuickTest Professional before you test your application.

Web 2.0 Toolkit Support

The Complexities of Testing Web 2.0 Controls

Web 2.0 sites often include a feature-rich, user-friendly interface based on client-side interactivity frameworks. The controls in these sites are generally created using a combination of HTML and client-side JavaScript code that create complex, interactive application objects.

Many groups and organizations have published Web 2.0 toolkits. These toolkits comprise open source JavaScript libraries that define Web 2.0 controls. Developers can use or customize these toolkits to build Web 2.0 applications instead of developing Web 2.0 controls from scratch.

The QuickTest Web Add-in does not recognize these complex controls and, instead, relates to the HTML elements that comprise them. This results in low-level steps on generic Web test objects. Such steps may be difficult to create, read, and maintain.

Testing Web 2.0 Controls with QuickTest Web 2.0 Add-in Support

QuickTest Web Add-in Extensibility makes it possible to develop Web-based add-ins that can identify the controls in a Web 2.0 application in a way that better matches the intended purpose and functionality of those controls.

QuickTest provides built-in Web Add-in Extensibility support for several public Web 2.0 toolkits. The support for each toolkit is packaged as a child add-in of the Web Add-in. If you install the Web 2.0 Toolkit Support, you can load this support by selecting the relevant toolkit name in the Add-in Manager. The Web 2.0 Toolkit Support Setup is available from the **Add-in Extensibility and Web 2.0 Toolkits** option in the QuickTest Professional setup.

The operations supported for each Web 2.0 test object class are a combination of custom operations developed for that test object class and operations directly inherited from the corresponding (base) Web Add-in test object class.

You work with a Web 2.0 toolkit add-in much the same way as you work with the regular Web Add-in. When the toolkit support is loaded, you can learn, record, create checkpoints, run steps, and use all standard QuickTest functionality on controls from these toolkits.

QuickTest provides support for the following toolkits:

- ▶ ASP .NET Ajax - <http://www.asp.net/ajax/>
- ▶ Google Web Toolkit (GWT) - <http://code.google.com/webtoolkit/>
- ▶ Dojo - <http://www.dojotoolkit.org>
- ▶ Yahoo User Interface (Yahoo UI) - <http://developer.yahoo.com/yui/>

For details on the test objects and operations supported for these toolkits, see the **Web 2.0 Toolkits** section of the *HP QuickTest Professional Object Model Reference*.

Considerations for Working with Web 2.0 Add-ins

- **jQuery Library Injection.** The Web 2.0 Add-in support is based on the jQuery JavaScript library. Therefore, if you load any Web 2.0 add-in, QuickTest injects the jQuery JavaScript library into every Web page that opens in a browser while QuickTest is open (unless a jQuery library is already included in the page).

The specific jQuery file injected for each Web 2.0 add-in is specified in the the add-in's toolkit XML file, located in: `<QuickTest installation>\dat\Extensibility\Web\Toolkits\<<ToolkitName>\<ToolkitName>.xml`

- **F1 Help Support.** When you press F1 on a test object operation that was inherited from the Web Add-in, the Help displays information about that operation for the Web Add-in test object class from which the operation was inherited, and not for the extensibility-based test object class used in your step.

Additionally, the details in the Help file reflect the behavior of the test objects and operations in the XML files provided with QuickTest. If these files were customized or modified in any way, the details in the Help files supplied with QuickTest may no longer be accurate.

In general, when the content of the extensibility files for a Web 2.0 toolkit is modified, the Help file should also be changed as described in "Customization Guidelines" on page 663. In these cases, you should contact the person or organization who customized the files as your first contact point for support.

- **Checkpoints and Output Values.** Inserting checkpoints and output values on Web 2.0 objects is supported only when recording steps.
- **Container Objects.** Some Web 2.0 objects that visually or behaviorally seem to contain other objects in a Web application are not learned as container objects in terms of the test object hierarchy. For example, this is the case for the YUIDialogBox and GWTDialogBox test objects.

- **Identification property values.** When working in Mozilla Firefox, the value of the **selected item** or **selected** identification property is not available in the Object Spy for some Web 2.0 test object classes. The same is true when updating property values from the application in the object repository. This is because the value is only retrievable when the browser is in focus.

Workaround: Retrieve the property value without removing focus from the browser. For example:

```
Browser("Dijit Tree Test").Page("Dijit Tree Test").DojoTree("mytree").Select
"Continents;Africa"

msgbox Browser("Dijit Tree Test").Page("Dijit Tree Test").DojoTree("mytree").
GetROProperty("selected item")
```

- **Object Type Identification.** In the toolkit XML file, the **<HTMLTags>** and **<Conditions>** elements in the **<Identification>** section for the relevant test object class define how QuickTest maps Web controls to that class.

In the example below, QuickTest identifies a control as a GWTToggleButon test object (when the GWT Add-in is loaded) if it has a <div> HTML tag and a className HTML property with a value that matches the regular expression: `.*gwt-ToggleButton.*`

```
<Control TestObjectClass="GWTToggleButon">
  <Settings>
    <Variable name="default_imp_file" value="JavaScript\GWTToggleButon.js"/>
  </Settings>
  <Identification>
    <Browser name="*">
      <HTMLTags>
        <Tag name="div"/>
      </HTMLTags>
      <Conditions type="IdentifyIfPropMatch">
        <!-- The search string in this condition is treated as a regular expression
and
is therefore equivalent to .*gwt-ToggleButton.* -->
        <Condition prop_name="className" expected_value="gwt-
ToggleButton" is_reg_exp="true"/>
      </Conditions>
    </Browser>
```

In some cases (for example, when `<Conditions type="CallIDFuncIfPropMatch">`), a JavaScript function that contains identification criteria is also used to help map controls to a test object class.

Keep in mind that the support provided in the HP-furnished Web 2.0 add-ins is dependent on the HTML and DOM structure of the controls. If developers of a Web 2.0-based application change the values of a control's properties, then the values defined for the `<HTMLTags>` and `<Conditions>` elements of the toolkit XML files (or JavaScript files) may not enable QuickTest to correctly identify those controls.

If QuickTest is not identifying an object in your application as you expect, you can view or adjust these values in the relevant toolkit support files.

The toolkit XML files are located in: `<QuickTest installation>\dat\Extensibility\Web\Toolkits\<ToolkitName>\<ToolkitName>.xml`

The JavaScript files are in a **JavaScript** folder under the above folder.

If you modify this (or any) HP-furnished toolkit support set file, follow the guidelines described in "Customization Guidelines" on page 663.

For more details on the way QuickTest identifies supported controls and for information on the implementation of the supported operations, see the comments provided in the XML and JavaScript files for the relevant toolkit support set.

Customization Guidelines

If you are familiar with Web Add-in Extensibility, then you can customize or further extend the built-in Web 2.0 support to match the needs of the Web 2.0 toolkit application you are testing.

Additionally, if you have installed **Extensibility Accelerator**, you can use this Visual Studio-like IDE to make it faster and easier to design and develop the required extensibility XML files so that you can invest your main efforts in the development of the JavaScript functions that will enable QuickTest to work with your custom Web controls.

Extensibility Accelerator also comes with built-in projects for the QuickTest Web 2.0 add-ins. You can use these projects to help you learn the Extensibility Accelerator features or to more easily add to or modify the provided support files.

If you customize or further extend any of the HP-furnished Web Add-in Extensibility files, you should also do the following:

- ▶ Make a copy of, or otherwise back up, the original HP-provided files.
- ▶ Change the name and description that are displayed in the Add-in Manager for the toolkit. Include the text: "Provided by <YourOrganization>" in the Add-in Manager description (in the **Controls\Description** element of the toolkit XML file).
- ▶ Create your own Help file to be opened for the customized test object classes or operations. You must use a different file name than the HP-provided Help file. (Change the file name in the **HelpInfo** element of the Test Object XML file.)

Note: When installing the Web 2.0 add-ins, if a previous version of a selected add-in is installed on your computer, Setup stores the previous files in a backup folder before installing. You may need to merge any customizations you made to the previous version with the new version.

For more information on how to make these changes and how to customize the support files, see the QuickTest Professional Web Add-in Extensibility documentation, available in the **<QuickTest installation folder>\help\Extensibility** folder.

For information on working with Extensibility Accelerator, see the *HP Extensibility Accelerator for HP Functional Testing User Guide*.

Web Add-in Extensibility

QuickTest Professional Web Add-in Extensibility enables you to develop support for testing third-party and custom Web controls that are not supported out-of-the-box by the QuickTest Professional Web Add-in.

If the test object class that QuickTest uses to represent a control does not provide the operations and properties necessary to operate on your control, you can use Web Add-in Extensibility to create a new test object class.

You can then map the control to the new test object class, and design the test object class behavior in JavaScript. You can program how operations are performed on the control, how properties are retrieved, and more.

You can also teach QuickTest to treat a control that contains a set of lower-level controls as a single functional control, instead of relating to each lower-level control separately.

To implement Web Add-in Extensibility, you need to be familiar with:

- ▶ QuickTest Professional and its Object Model Reference
- ▶ The behavior of the custom control (operations, properties, events)
- ▶ Web programming (HTML and JavaScript)
- ▶ XML (basic knowledge)

Extensibility Accelerator for HP Functional Testing (described on page 666) is a Visual Studio-like IDE that facilitates the design, development, and deployment of Web Add-in Extensibility support. You can install it from:

- ▶ The **Add-in Extensibility and Web 2.0 Toolkits** option in the QuickTest Professional setup program.
- ▶ www.hp.com/go/functionaltestingWeb2

Extensibility Accelerator also provides samples of support developed using Web Add-in Extensibility, which you can use to gain a better understanding of how to create your own support.

For details on implementing Web Add-in Extensibility, see the Web Add-in Extensibility Help, available from the QuickTest Professional Extensibility Documentation program group (**Start > Programs > QuickTest Professional > Extensibility > Documentation**).

A printer-friendly (PDF) version of the *HP QuickTest Professional Web Add-in Extensibility Developer Guide* is available in the **<QuickTest Professional installation folder>\help\Extensibility** folder.

Extensibility Accelerator for HP Functional Testing

An increasing number of Web applications are making use of Web 2.0-based toolkits, such as ASP.NET AJAX, Dojo, YahooUI, and GWT to add dynamic and interactive content to their sites. The controls in these toolkits are complex and require sophisticated and flexible testing capabilities.

QuickTest Professional Web Add-in Extensibility enables you to extend the Web Add-in to customize how QuickTest recognizes and interacts with different types of controls. Until now, using Web Add-in Extensibility consisted of manually developing and maintaining toolkit support sets.

Extensibility Accelerator for HP Functional Testing is a Visual Studio-like IDE that facilitates the design, development, and deployment of these support sets. It makes it faster and easier to create the required extensibility XML files so that you can invest your main efforts in the development of the JavaScript functions that will enable QuickTest to work with your custom Web controls.

The Extensibility Accelerator user interface helps you define new test object classes, operations, and properties. It also provides a point-and-click mechanism you can use to map the test object classes you defined to controls in your application. Extensibility Accelerator deployment capabilities enable you to automatically deploy your new toolkit support set to QuickTest or to package it so that you can share it with other QuickTest users.

The Extensibility Accelerator for HP Functional Testing installation is available from:

- ▶ The **Add-in Extensibility and Web 2.0 Toolkits** option in the QuickTest Professional setup program.
- ▶ www.hp.com/go/functionaltestingWeb2

Troubleshooting and Limitations - Web Add-in

This section contains general troubleshooting and limitation information about the Web add-in, and includes the following sections:

- ▶ "User Account Control (Where Applicable)" on page 667
- ▶ "Test Objects, Methods, and Properties" on page 668
- ▶ "Creating and Running Testing Documents" on page 669
- ▶ "Running Tests or Components in Internet Explorer" on page 669
- ▶ "Running Tests or Components in Mozilla Firefox" on page 671
- ▶ "Recognition of WebTable Test Objects" on page 671
- ▶ "Multi-Lingual Support for Web Browsers" on page 674

User Account Control (Where Applicable)

- ▶ If you are working on a computer where the UAC (User Account Control) option is set to ON and the Internet Explorer **Enable Protected Mode** option is selected, then QuickTest cannot open the Internet Explorer browser at the beginning of record or run sessions as instructed by the **Open the following browser when a record or run session begins** option in the Record and Run Settings dialog box.

Workaround: Clear the **Enable Protected Mode** option in Internet Explorer (**Tools > Internet Options > Security**), apply the changes, and close the browser.

- ▶ If you are working on a computer where the UAC (User Account Control) option is set to ON, QuickTest does not support testing on Mozilla Firefox browsers that were installed (or upgraded to a new version) after you installed QuickTest Professional.

Workaround: After installing Mozilla Firefox on the environment described above, log in as an administrator and open QuickTest. This enables QuickTest to install files that are required for Mozilla Firefox support.

Test Objects, Methods, and Properties

- ▶ Web test objects do not support the **Class Name** identification property. If you try to run a **ChildObjects**(*Descr*) step on a Web object, and the *Descr* argument includes the **Class Name** property, a General Run Error message is displayed.

Workaround: Use the **micclass** property in the *Descr* argument.

- ▶ If you record drag and drop steps on a Web element within the same frame, the test steps may fail during the run session if the screen resolution is not identical to the screen resolution during the recording session. This is because the target location coordinates may be different for different screen resolutions.

Workaround: If this problem occurs, adjust the **Drop** coordinates according to the new location.

- ▶ QuickTest Professional records changes in the edit field only on <input type="file"> tags. Browsing operations are not recorded.
- ▶ Clicks on form tags of type POST may not run correctly.

Workaround: If this problem occurs, change the replay type before the click to **Run by mouse operations** using:
Setting.WebPackage("ReplayType") = 2. It is recommended to return the replay type to the default (**Run by Events**) setting after the click step:
Setting.WebPackage("ReplayType") = 1.

- QuickTest does not record **Drag** and **Drop** operations performed on an ASP.NET Ajax drag panel control.

Workaround: Make sure that the relevant object exists in your object repository (or learn the object), and then insert the required **Drag** and **Drop** operations manually.

Creating and Running Testing Documents

- If you use the Tab key when recording password fields in the AutoComplete dialog box, QuickTest may record incorrectly.

Workaround: Press ENTER after entering the user name or click the button for logging in.

- In QuickTest version 9.2 and earlier, when you learned the objects on a Web page or ran a **Page.ChildObjects** step on a page with embedded hierarchies, (containing objects such as Java applets, ActiveX controls, or .NET Windows Forms controls), only the Web elements on the page were returned.

The learn behavior for QuickTest 9.5 and later changed such that when you learn all objects on a page, objects from the embedded hierarchies are also learned. However, to preserve backward compatibility, the behavior of the **ChildObjects** method continues to retrieve only the Web objects.

- When QuickTest opens a browser, it may not correctly recognize multiple tabs that were opened and saved from a previous browser session.

Workaround: If multiple tabs are required, open them during the run session by adding the relevant steps to your test or component.

Running Tests or Components in Internet Explorer

- When using Microsoft Internet Explorer 7.0, QuickTest cannot switch to tabs that are not visible on the tab-band without scrolling.

Workaround: Perform either of the following:

- Maximize the browser to increase the number of tabs that are visible in the tab-band without scrolling.
- Increase the screen resolution to allow more tabs to be visible in the tab-band.

- ▶ If you record a click on an area of an image map that is not mapped to a URL in Microsoft Internet Explorer, QuickTest Professional will perform a click on the first mapped area of that map during the run session.
- ▶ QuickTest Professional does not record on customized toolbar buttons in Microsoft Internet Explorer. (It records only on the toolbar buttons that are displayed by default in the browser.)
- ▶ When working with Internet Explorer 7.0 or later on a Windows Vista or Windows 7 operating system with UAC enabled, QuickTest Professional may not recognize Web objects, even though the Web Add-in is installed and loaded.

Workaround: Check and modify your Internet Explorer settings.

- ▶ Modify the Internet Explorer security settings, if needed.

For example, in Internet Explorer 7.0, select **Tools > Internet Options**. In the Security tab, clear the **Enable Protected Mode** check box and click **OK**.

- ▶ Enable the BHOManager Class Add-on if it is disabled. (QuickTest installs this add-on in Internet Explorer 7.0 or later. The BHOManager Class Add-on must be set to **Enabled** for QuickTest to interact with the browser and its objects.)

For example, in Internet Explorer 7.0, select **Tools > Manage Add-ons > Enable or Disable Add-ons** (or **Tools > Internet Options > Programs tab > Manage add-ons** button if the **Manage Add-ons** menu item is not listed). In the Manage Add-ons dialog box, click the **BHOManager Class** to highlight it. Then, in the **Settings** area, click the **Enable** radio button and click **OK**.

- ▶ Turn off **UAC (User Account Control)**, if needed. This disables Internet Explorer's protected mode. For details, see the *HP QuickTest Professional Installation Guide*.
- ▶ QuickTest Professional does not record on the Search frame of the Microsoft Internet Explorer browser.
- ▶ QuickTest Professional does not record on the Find window of the Microsoft Internet Explorer browser.

- ▶ QuickTest Professional may respond slowly during a recording session if the drop-down boxes in a Web page contain a lot of data.

Workaround: Learn the objects on a Web page that contains a lot of data (instead of recording).

Running Tests or Components in Mozilla Firefox

- ▶ The Object Spy and Checkpoint Properties dialog boxes do not retrieve the current value of edit boxes in Mozilla Firefox dialog boxes.
- ▶ The **Type** property of the **WebButton** test object has a different default value in Microsoft Internet Explorer and Mozilla Firefox. In Microsoft Internet Explorer the default value is **Button**, but in Mozilla Firefox the default value is **Submit**.

Workaround: Do not use the **Type** property in the description of a **WebButton** test object.

- ▶ If two minor versions of Mozilla Firefox are installed on the same computer, and the earlier version (for example, Firefox 1.5.0.3) was installed after the later version (for example, Firefox 1.5.0.8), QuickTest may not recognize which is the latest version.
- ▶ QuickTest does not support the **showModalDialog** command in Mozilla Firefox.
- ▶ QuickTest does not support anonymous content elements in non-XUL frames. (For example, the buttons in the Mozilla Firefox SSL exception page.)
- ▶ When recording steps in Mozilla Firefox, additional steps may be recorded.

Workaround: Manually remove the extraneous steps after the recording session ends.

Recognition of WebTable Test Objects

By default, when using the QuickTest Web Add-in, QuickTest recognizes any HTML table as a **WebTable** test object.

However, in QuickTest 9.5 or 10.00, the default behavior was to ignore HTML tables with one row and one column during Object Spy, learn, and record sessions.

In specific situations, this changed default behavior may result in differences when learning new test objects or when running steps containing Web test objects that were learned in QuickTest 9.5 or 10.00. For example, the **ChildObjects** method may return a different value for parent objects that contain Web tables.

If necessary, you can revert to the previous behavior by enabling (and optionally modifying) **abstract table** support.

Abstract tables are defined in a built-in Web Add-in Extensibility toolkit support set called **HPInternal**. By default, this toolkit support set is not loaded.

To activate the abstract table support:

- 1 Open
<QuickTest Professional installation folder>\dat\Extensibility\Web\Toolkits\HPInternal\loadalways.ind
- 2 Change the single line in the file to: load=true

To modify which types of tables QuickTest treats as an abstract table:

Edit the **IsHPAbstractTable** JavaScript located in:
<QuickTest Professional installation folder>\dat\Extensibility\Web\Toolkits\HPInternal\HPAbstractTable.js.

The sample **IsHPAbstractTable** JavaScript function below causes QuickTest to treat Web table elements containing one row and one column as abstract tables:

```
function IsHPAbstractTable()
{
    // Treat all tables with only one cell as abstract tables
    if ( _elem.rows.length == 1 && _elem.rows[0].cells.length == 1 )
    {
        return true;
    }
    return false;
}
```


To instruct QuickTest to ignore additional types of Web table elements, modify the **IsHPAbstractTable** JavaScript function to return **true** for those types based on their HTML properties or other information. (Use the token **_elem** to represent the Web element QuickTest is currently handling.).

Caution: HPAbstractTable.js affects the way QuickTest identifies Web objects and can cause problems if modified incorrectly. Edit this file only if you are an experienced JavaScript programmer and are familiar with the implementation of your Web controls. Make sure to create a backup copy of the file before making changes.

Checkpoints and Output Values

- ▶ Checkpoints on page source/HTML tags cannot be inserted from the Active Screen and must be inserted while recording. These checkpoints may fail during the first run session.

Workaround: Perform an update run (**Automation> Update Run Mode**) of your test or component before you run a test or component that includes a page source/HTML tag checkpoint.

- ▶ If you insert checkpoints from the Active Screen when you are working with an application containing a browser control rather than with a Web browser, your checkpoints may fail.

Workaround: Insert checkpoints while recording.

Multi-Lingual Support for Web Browsers

- ▶ In Internet Explorer, the AutoComplete operation on edit fields is not recorded.

Workaround: You can disable the AutoComplete feature in Microsoft Internet Explorer by selecting **Tools > Internet Options > Advanced** and deselecting the **Use inline AutoComplete** under the **Browsing** options in Microsoft Internet Explorer.

- ▶ If a test or component contains a step that closes a Mozilla Firefox browser, QuickTest may behave unexpectedly when that step is reached during a run session.

Workaround: Do not include a step that closes a Mozilla Firefox browser.

39

Configuring Web Event Recording for Web Objects

If QuickTest does not record Web events on your Web test objects in a way that matches your needs, you can configure the events you want to record for each type of Web object using the Web Event Recording Configuration dialog box.

This chapter includes:

- About Configuring Web Event Recording on page 676
- Selecting a Predefined Event Recording Configuration on page 678
- Customizing the Web Event Recording Configuration on page 680
- Recording Right Mouse Button Clicks on page 690
- Saving and Loading Custom Event Configuration Files on page 694
- Resetting Event Recording Configuration Settings on page 696

About Configuring Web Event Recording

When you record on a Web application, QuickTest generates steps by recording the events you perform on the Web objects in your application. An **event** is a notification that occurs in response to an operation, such as a change in state, or as a result of the user clicking the mouse or pressing a key while working in a Web application. You may find that you need to record more or fewer events than QuickTest automatically records by default.

You can modify the default event recording settings for Web objects using the Web Event Recording Configuration dialog box to use one of three predefined configurations, or you can customize the individual event recording configuration settings to meet your specific needs.

For example, QuickTest does not generally record mouseover events on link objects. If, however, you have mouseover behavior connected to a link, it may be important for you to record the mouseover event. In this case, you could customize the configuration to record mouseover events on link objects whenever they are connected to a behavior.

Additional Considerations

When configuring Web Event recording, consider the following:

- ▶ Event configuration is a global setting and therefore affects all steps that are recorded after you change the settings.
- ▶ Changing the event configuration settings does not affect steps that have already been recorded. If you find that QuickTest recorded more or less than you need, change the event recording configuration and then re-record the steps that are affected by the change.
- ▶ Changes to the custom Web event recording configuration settings do not affect open browsers. To apply your changes, make the changes you need in the Web Event Recording Configuration dialog box, refresh any open browsers, and then start a new recording session.

- ▶ The settings in the Web Event Recording Configuration dialog box affect recording only for objects that QuickTest recognizes as Web test objects. The recording configuration for other Web-based objects (such as Siebel, PeopleSoft, .NET Web Forms, and SAP Web controls) is defined by environment-specific XML configuration files.

Note: For the purposes of Web event recording, QuickTest treats Web test objects that are child objects of a PSFrame test object as PeopleSoft objects and thus applies the settings in the PeopleSoft event configuration XML file when recording those objects.

For more information, see "Web Event Recording Configurations" on page 74.

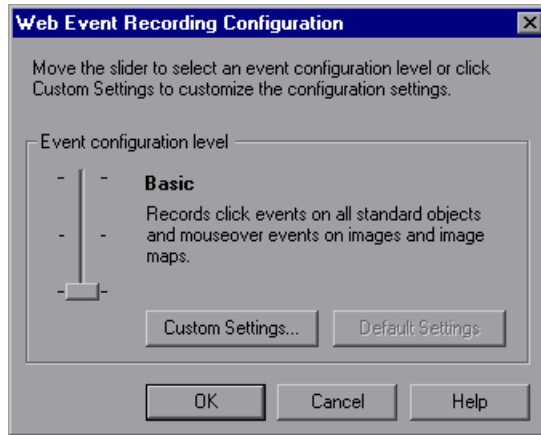
Selecting a Predefined Event Recording Configuration

The Web Event Recording Configuration dialog box offers three predefined event-configuration levels. By default, QuickTest uses the **Basic** level. If QuickTest does not record all the events you need, you may require a higher level.

Level	Description
Basic	<p>Default</p> <ul style="list-style-type: none"> ▶ Always records click events on Web objects that commonly support clicking, such as images, buttons, and radio buttons. ▶ Always records the submit event within forms. ▶ Records click events on other Web objects with an Internet Explorer handler or behavior connected. For more information on handlers and behaviors, see "Listening Criteria" on page 686. ▶ Records the mouseover event on images and image maps only if the event following the mouseover is performed on the same object.
Medium	Records click events on the <DIV>, , and <TD> HTML tag objects, in addition to the events recorded in the basic level.
High	Records mouseover, mousedown, and double-click events on Web objects with handlers or behaviors attached, in addition to the events recorded in the basic level. For more information on handlers and behaviors, see "Listening Criteria" on page 686.

To set a predefined event-recording configuration:

- 1** Select **Tools > Web Event Recording Configuration**. The Web Event Recording Configuration dialog box opens.



- 2** Use the slider to select your preferred predefined event recording configuration.

Tip: You can click the **Custom Settings** button to open the Custom Web Event Recording dialog box where you can customize the event recording configuration. For more information, see "Customizing the Web Event Recording Configuration" on page 680.

You can click the **Default Settings** button to return to the **Basic** level.

- 3** Click **OK**.

Customizing the Web Event Recording Configuration

If the predefined Web event configuration levels do not exactly match your recording needs, you can customize the event recording configuration using the Custom Web Event Recording Configuration dialog box.

The Custom Web Event Recording Configuration dialog box enables you to customize event recording in several ways. You can:

- ▶ Add or delete objects to which QuickTest should apply special listening or recording settings. For more information, see "Adding and Deleting Objects in the Custom Configuration Object List" on page 683.
- ▶ Add or delete events for which QuickTest should listen. For more information, see "Adding and Deleting Listening Events for an Object" on page 685.
- ▶ Modify the listening or recording settings for an event. For more information, see "Modifying the Listening and Recording Settings for an Event" on page 686.

To customize the event recording configuration:

- 1** Select **Tools > Web Event Recording Configuration**. The Web Event Recording Configuration dialog box opens.
- 2** Click the **Custom Settings** button. The Custom Web Event Recording Configuration dialog box opens.

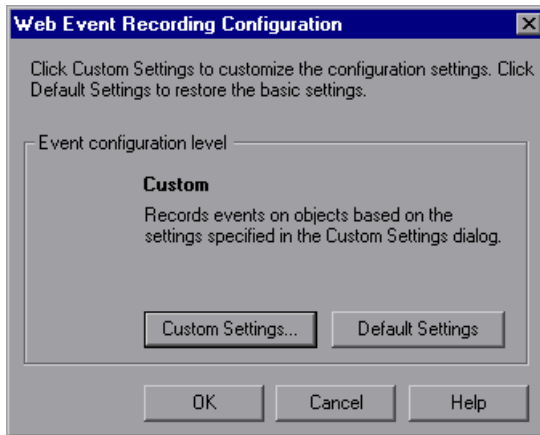


3 Customize the event recording configuration using the following options:

Option	Description
Objects pane	<p>Displays a list of Web test object classes and HTML tag objects. Only HTML tag objects can be added or deleted.</p> <ul style="list-style-type: none"> ▶ To add an HTML tag object, select Object > Add. ▶ To delete an HTML object from the list, select Object > Delete. <p>For more information, see "Adding and Deleting Objects in the Custom Configuration Object List" on page 683.</p>
Events pane	<p>Displays a list of events associated with the object.</p> <ul style="list-style-type: none"> ▶ To add an event to the Events pane, select Event > Add. ▶ To delete an event, select Event > Delete. <p>For more information, see "Adding and Deleting Listening Events for an Object" on page 685.</p>
Event Name	<p>Specifies the name of the event to which QuickTest and/or records, depending on the settings you specify.</p>
Listen	<p>Specifies the criteria that instructs QuickTest when to listen to the event.</p> <ul style="list-style-type: none"> ▶ Always. Always listens to the event. ▶ If Handler. Listens to the event if a handler is attached to it. A handler is code in a Web page, typically a function or routine written in a scripting language, that receives control when the corresponding event occurs. ▶ If Behavior. Listens to the event if a DHTML behavior is attached to it. A DHTML behavior encapsulates specific functionality or behavior on a page. When applied to a standard HTML element on a page, a behavior enhances that element's default behavior. ▶ If Handler or Behavior. Listens to the event if a handler or behavior is attached to it. ▶ Never. Never listens to the event. <p>For more information, see "Modifying the Listening and Recording Settings for an Event" on page 686 and "Tips for Working with Event Listening and Recording" on page 688.</p>

Option	Description
Record	Enables or disables recording of the event for the selected object, or enables recording of the event only if the subsequent event occurs on the same object. See "Recording Status" on page 687 and "Tips for Working with Event Listening and Recording" on page 688.
Reset	Enables you to reset your settings to a preconfigured level.

- 4 Click **OK**. The Custom Web Event Recording Configuration dialog box closes. The slider scale on the Web Event Recording Configuration dialog box is hidden and the configuration description displays **Custom**.



- 5 Click **OK** to close the Web Event Recording Configuration dialog box.

Adding and Deleting Objects in the Custom Configuration Object List

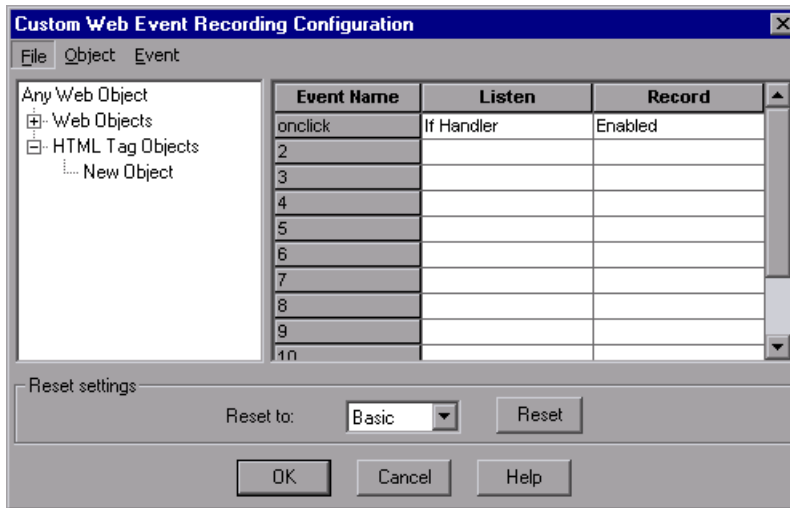
The Custom Web Event Recording Configuration dialog box lists objects in an object hierarchy. The top of the hierarchy is **Any Web Object**. The settings for **Any Web Object** apply to any object on the Web page, for which there is no specific event recording configuration set. Below this are the **Web Objects** and **HTML Tag Objects** categories, each of which contains a list of objects.

When working with the objects in the Custom Web Event Recording Configuration dialog box, keep the following principles in mind:

- ▶ If an object is listed in the Custom Web Event Recording Configuration dialog box, then the settings for that object override the settings for **Any Web Object**.
- ▶ You cannot delete or add to the list of objects in the **Web Objects** category, but you can modify the settings for any of these objects.
- ▶ You can add any HTML Tag object in your Web page to the **HTML Tag Objects** category.

To add objects to the event configuration object list:

- 1 In the Custom Web Event Recording Configuration dialog box, select **Object > Add**. A **New Object** object is displayed in the HTML Tag Objects list.



- 2 Click **New Object** to rename it. Enter the exact HTML Tag name.

By default the new object is set to listen and record **onclick** events with handlers attached.

For more information on adding or deleting events, see "Adding and Deleting Listening Events for an Object" on page 685. For more information on listening and recording settings, see "Modifying the Listening and Recording Settings for an Event" on page 686.

To delete objects from the HTML Tag Objects list:

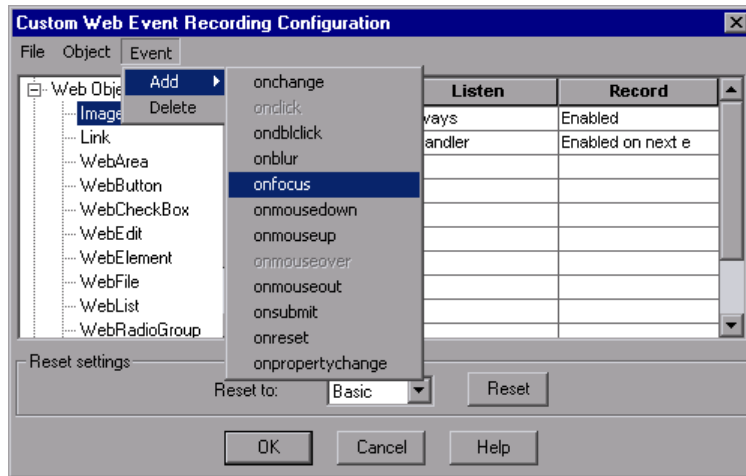
- 1 From the Custom Web Event Recording Configuration dialog box, select the object in the HTML Tag Objects category that you want to delete.
- 2 Select **Object > Delete**. The object is deleted from the list.

Adding and Deleting Listening Events for an Object

You can add or delete events from the list of events that trigger QuickTest to listen to an object.

To add listening events for an object:

- 1 In the Custom Web Event Recording Configuration dialog box, select the object to which you want to add an event, or select **Any Web Object**.
- 2 Select **Event > Add**. A list of available events opens.



- 3 Select the event you want to add. The event is displayed in the **Event Name** column in alphabetical order. By default, QuickTest listens to the event when a handler is attached and always records the event (as long as it is listened to at some level).

For more information on listening and recording settings, see "Modifying the Listening and Recording Settings for an Event" on page 686.

To delete listening events for an object:

- 1 In the Custom Web Event Recording Configuration dialog box, select the object from which you want delete an event, or select **Any Web Object**.
- 2 Select the event you want to delete from the **Event Name** column.
- 3 Select **Event > Delete**. The event is deleted from the **Event Name** column.

Modifying the Listening and Recording Settings for an Event

You can select the listening criteria and set the recording status for each event listed for each object.

Note: The listen and record settings are mutually independent. This means that you can choose to listen to an event for a particular object, but not record it, or you can choose not to listen to an event for an object, but still record the event. For more information, see "Tips for Working with Event Listening and Recording" on page 688.

Listening Criteria

For each event, you can instruct QuickTest to listen every time the event occurs on the object, only if an event handler is attached to the event, only if a DHTML behavior is attached to the event, if either an event handler or DHTML behavior are attached to the event, or to never listen to the event.

An event **handler** is code in a Web page, typically a function or routine written in a scripting language, that receives control when the corresponding event occurs.

Note: QuickTest supports event handlers that are attached using an **on*** attribute (such as **onclick** or **onmouseover**). It does not support other event handlers, such as those attached using an **addEventListener** or **attachEvent** command.

A DHTML **behavior** encapsulates specific functionality or behavior on a page. When applied to a standard HTML element on a page, a behavior enhances that element's default behavior.

To specify the listening criterion for an event:

- 1 From the Custom Web Event Recording Configuration dialog box, select the object for which you want to modify the listening criterion or select **Any Web Object**.
- 2 In the row of the event you want to modify, select the listening criterion you want from the **Listen** column.

Event Name	Listen	Record
onclick	If Handler	Enabled
onkeydown	Always	Enabled
onmouseover	If Handler	Disabled
4	Always	
5	If Handler	
6	If Behavior	
7	If Handler or Behavior	
8	Never	
9		
10		

You can select **Always**, **If Handler**, **If Behavior**, **If Handler or Behavior**, or **Never**.

Recording Status

For each event, you can enable recording, disable recording, or enable recording only if the next event is dependent on the selected event.

- **Enabled.** Records the event each time it occurs on an object as long as QuickTest listens to the event on the selected object, or on another object to which the event bubbles.

Bubbling is the process whereby, when an event occurs on a child object, the event can travel up the chain of hierarchy within the HTML code until it encounters an event handler to process the event.

- **Disabled.** Does not record the specified event and ignores event bubbling where applicable.
- **Enabled on next event.** Same as **Enabled**, except that it records the event only if a subsequent event occurs on the same object.

For example, suppose a mouseover behavior modifies an image link. You may not want to record the mouseover event each time you happen to move the pointer over this image. It is essential, though, that the mouseover event be recorded before a click event on the same object because only the image that is displayed after the mouseover event enables the link event. This option applies only to the Image and WebArea objects.

To set the recording status for an event:

- 1 From the Custom Web Event Recording Configuration dialog box, select the object for which you want to modify the recording status or select **Any Web Object**.
- 2 In the row of the event you want to modify, select a recording status from the **Record** column.

Event Name	Listen	Record
onclick	Always	Enabled
onmouseover	If Handler	Enabled on next ev
3		Disabled
4		Enabled
5		Enabled on next ever
6		
7		
8		
9		

Tips for Working with Event Listening and Recording

It can sometimes be difficult to find the ideal listen and recording settings. When defining these settings, keep in mind the following guidelines:

- If settings for different objects in the Objects pane conflict, QuickTest gives first priority to settings for specific **HTML Tag Objects** and second priority to **Web Objects** settings. QuickTest applies the settings for **Any Web Object** only to Web objects that do not belong to any other loaded Web-based environment and were not defined in the **HTML Tag Object** or **Web Objects** areas.

- To record an event on an object, you must instruct QuickTest to listen for the event, and to record the event when it occurs. You can listen for an event on a child object, even if a parent object contains the handler or behavior, or you can listen for an event on a parent object, even if the child object contains the handler or behavior.

However, you must enable recording for the event on the **source object** (the object on which the event actually occurs, regardless of which parent object contains the handler or behavior).

For example, suppose a table cell with an **onmouseover** event handler contains two images. When the mouse moves over either of the images, the event also bubbles up to the cell, and the bubbling includes information on the image that the mouse moved over. You can record this mouseover event by:

- Setting **Listen** on the <TD> tag mouseover event to **If Handler** (so that QuickTest "hears" the event when it occurs), while disabling recording on it, and then setting **Listen** on the tag mouseover event to **Never**, while setting **Record** on the tag to **Enable** (to record the mouseover event on the image after it is listened to at the <TD> level).
- Setting **Listen** on the tag mouseover event to **Always** (to listen for the mouseover event even though the image tag does not contain a behavior or handler), and setting **Record** on the tag to **Enabled** (to record the mouseover event on the image).
- Instructing QuickTest to listen for many events on many objects may lower performance, so it is recommended to limit **Listen** settings to the required objects.
- In Internet Explorer, listening to the object on which the event occurs (the source object) can, in rare situations, interfere with the event.

If you find that your application works properly until you begin recording on the application using QuickTest, your **Listen** settings may be interfering.

- If this problem occurs with a mouse event, try selecting the appropriate **Use standard Windows mouse events** options in the Web > Advanced pane of the Options dialog box (**Tools > Options > Web** node > **Advanced** node). For more information, see "Advanced Web Options" on page 62.

- ▶ If this problem occurs with a keyboard or internal event, or the **Use standard Windows mouse events** option does not solve your problem, set the **Listen** settings for the event to **Never** on the source object (but keep the record setting enabled on the source object), and set the **Listen** settings to **Always** for a parent object.

Recording Right Mouse Button Clicks

QuickTest enables you to record clicks made using left, center, and right mouse buttons. By default, only left clicks are recorded, but you can modify the configuration to record clicks from the right and center buttons, as well.

QuickTest records the Click statement when the **OnClick** event is triggered. QuickTest differentiates between the mouse buttons by listening for events configured for each of the mouse buttons. By default, it listens for the **OnMouseUp** event, but you can also configure it to listen for the **OnMouseDown** event using the Web Event Recording Configuration dialog box.

Notes:

- ▶ Recording of simultaneous clicking of more than one mouse button is not supported.
 - ▶ QuickTest does not record the right-click that opens the browser context menu, or the selection of an item from the context menu. For more information on modifying the script manually to enable these options, visit the HP Software Self-solve knowledge base and search for document ID KM185231.
-

Configuring QuickTest to Record Right Mouse Clicks

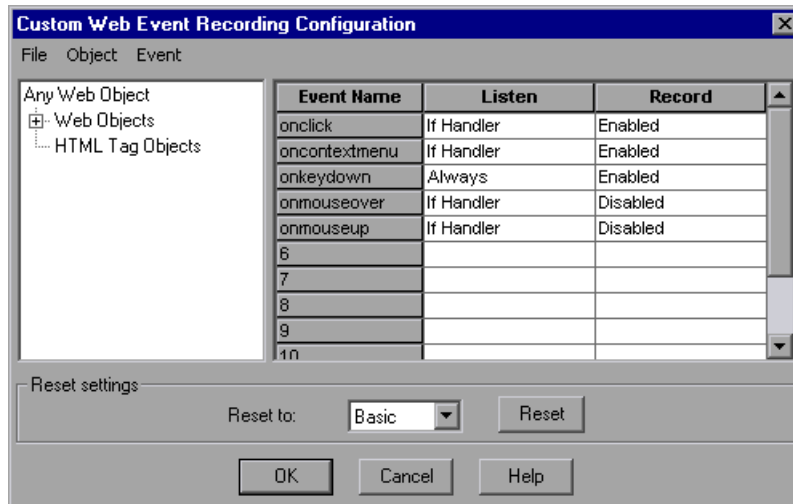
You instruct QuickTest to record right mouse clicks by modifying the configuration file manually and then loading it.

To configure QuickTest to record right mouse clicks:

- 1 Select **Tools > Web Event Recording Configuration**. The Web Event Recording Configuration dialog box opens.



- 2 Click the **Custom Settings** button. The Custom Web Event Recording Configuration dialog box opens.



- 3 In the Custom Web Event Recording Configuration dialog box, select **File > Save Configuration As**. The Save As dialog box opens.
- 4 Navigate to the folder in which you want to save the Web event recording configuration file, and enter a configuration file name. The extension for configuration files is **.xml**.
- 5 Click **Save** to save the file and close the dialog box.
- 6 Open the saved configuration file for editing in any text editor. The configuration file uses a defined structure. For more information on the XML file structure, see "Understanding the Web Event Recording Configuration XML Structure" on page 695.

The following example illustrates the beginning of an exported configuration file:

```
- <XML>
- <Object Name="Any Web Object">
  <Event Name="onclick" Listen="2" Record="2" />
  <Event Name="oncontextmenu" Listen="2" Record="2" />
  <Event Name="onkeydown" Listen="1" Record="2" />
  <Event Name="onmouseover" Listen="2" Record="1" />
- <Event Name="onmouseup" Listen="2" Record="1">
  <Property Name="button" Value="2" Listen="2" Record="2" />
```

The **Property Name** element controls the recording of the mouse buttons. The values of the mouse buttons are defined as follows:

- 1. Left
- 2. Right
- 4. Middle

7 Edit the file as follows:

- ▶ To record a left mouse click for the **onmouseup** event, add the following line:

```
<Property Name="button" Value="1" Listen="2" Record="2"/>
```

- ▶ To record right and left mouse clicks for the **onmousedown** event, add the following lines:

```
<Event Name="onmousedown" Listen="2" Record="1">
```

```
  <Property Name="button" Value="2" Listen="2" Record="2"/>
```

```
  <Property Name="button" Value="1" Listen="2" Record="2"/>
```

```
</Event>
```

Note: Only one event, either **onmouseup** or **onmousedown**, should be used to handle mouse clicks. If both events are used, QuickTest will record two clicks instead of one. By default, QuickTest listens for the **onmouseup** event.

8 Save the file.

- 9** In the Custom Web Event Recording Configuration dialog box, select **File > Load Configuration**. The Open dialog box opens.

- 10** Navigate to the folder in which you saved the edited configuration file, select the file, and click **Open**. The Custom Web Recording Configuration dialog box reopens.

- 11** Click **OK**. The new configuration is loaded, with all preferences corresponding to those you defined in the XML configuration file. Any Web objects you now record will be recorded according to these new settings.

Saving and Loading Custom Event Configuration Files

You can save the changes you make in the Custom Web Event Recording Configuration dialog box, and load them at any time.

You can also modify the XML file before loading it. For more information on the XML file structure, see "Understanding the Web Event Recording Configuration XML Structure" on page 695.

To save a custom configuration:

- 1** Customize the event recording configuration as desired. For more information on how to customize the configuration, see "Customizing the Web Event Recording Configuration" on page 680.
- 2** In the Custom Web Event Recording Configuration dialog box, select **File > Save Configuration As**. The Save As dialog box opens.
- 3** Navigate to the folder in which you want to save your event configuration file and enter a configuration file name. The extension for configuration files is **.xml**.
- 4** Click **Save** to save the file and close the dialog box.

To load a custom configuration:

- 1** Select **Tools > Web Event Recording Configuration** and then click **Custom Settings** to open the Custom Web Event Recording Configuration dialog box.
- 2** Select **File > Load Configuration**. The Open dialog box opens.
- 3** Locate the event configuration file (**.xml**) that you want to load and click **Open**. The dialog box closes and the selected configuration is loaded.

Understanding the Web Event Recording Configuration XML Structure

The Web event recording configuration XML file is structured in a certain format. If you are modifying the file, or creating your own file, you must ensure that you adhere to this format for your settings to take effect.

Following is a sample XML file:

```
<XML>
  <Object Name="Any Web Object">
    <Event Name="onclick" Listen="2" Record="2"/>
    <Event Name="onmouseup" Listen="2" Record="1">
      <Property Name="button" Value="2" Listen="2" Record="2"/>
    </Event>
  </Object>
  ...
  ...
  ...
  <Object Name="WebList">
    <Event Name="onblur" Listen="1" Record="2"/>
    <Event Name="onchange" Listen="1" Record="2"/>
    <Event Name="onfocus" Listen="1" Record="2"/>
  </Object>
</XML>
```

You define the listening criteria and recording status options in the XML using the following possible attributes:

Attribute	Possible Values
Listen	<ul style="list-style-type: none"> 1. Always 2. If Handler 4. If Behavior 6. If Handler or Behavior 0. Never
Record	<ul style="list-style-type: none"> 1. Disabled 2. Enabled 6. Enabled on Next Event

Resetting Event Recording Configuration Settings

You can restore predefined settings after you set custom settings by resetting the event recording configuration settings to the basic level from the Web Event Recording Configuration dialog box. You can also restore the default custom level settings from the Custom Web Event Recording Configuration dialog box.

Note: When you choose to reset predefined settings, your custom settings are cleared completely. If you do not want to lose your changes, be sure to save your settings in an event configuration file. For more information, see "Saving and Loading Custom Event Configuration Files" on page 694.

To reset basic level configuration settings from the Web Event Recording Configuration dialog box:

- 1 Select **Tools > Web Event Recording Configuration**. The Web Event Recording Configuration dialog box opens.
- 2 Click **Default**. The configuration slider is displayed again and all event settings are restored to the **Basic** event recording configuration level.
- 3 If you want to select a different predefined configuration level, see "Selecting a Predefined Event Recording Configuration" on page 678.

You can also restore the settings to a specific (base) custom configuration from within the Custom Web Event Recording Configuration dialog box so that you can begin customizing from that point.

To reset the settings to a custom level from the Custom Web Event Recording Configuration dialog box:

- 1 Select **Tools > Web Event Recording Configuration**. The Web Event Recording Configuration dialog box opens.
- 2 Click the **Custom Settings** button. The Custom Web Event Recording Configuration dialog box opens.
- 3 In the **Reset to** box, select the predefined event recording level you want.

- 4 Click **Reset**. All event settings are restored to the defaults for the level you selected.

Part VIII

The Web Services Add-in

40

Using the Web Services Add-in

You can use the Web Services Add-in to test the operations that your Web service supports by sending and receiving messages to and from your Web service.

For details on supported Web service toolkits, protocols, and other relevant version support information, see the **Web Services Add-in** section of the *HP QuickTest Professional Product Availability Matrix*, available from the Documentation Library Home page or the root folder of the QuickTest Professional DVD.

The Web Services Add-in provides test objects, methods, and properties that can be used when testing Web services. For more information, see the **Web Services** section of the *HP QuickTest Professional Object Model Reference*.

The following table summarizes basic information about the Web Services Add-in and how it relates to some commonly-used aspects of QuickTest.

General Information	
Checkpoints and Output Values	<ul style="list-style-type: none">▶ See the sections describing checkpoints and output values in the <i>HP QuickTest Professional User Guide</i> (for tests), and the <i>HP QuickTest Professional for Business Process Testing User Guide</i> (for components).▶ See "Supported Checkpoints and Output Values Per Add-in" on page 769.
Prerequisites	
Opening Your Application	You can open your Web Services application before or after opening QuickTest.
Add-in Dependencies	None

Setting Preferences	
Options Dialog Box	Use the Web Services pane. (Tools > Options > Web Services node) See "Setting Web Services Test Options" on page 733.
Test Settings Dialog Box (tests only)	Use the Web Services pane. (File > Settings > Web Services node) See "Defining Web Service Test or Component Settings" on page 736.
Application Area Settings Dialog Box (components only)	Use the Web Services pane. (File > Settings > Web Services node) See the section on defining Application Settings for your application area in the <i>HP QuickTest Professional for Business Process Testing User Guide</i> .

This chapter includes:

- ▶ About the Web Services Add-in on page 703
- ▶ Considerations for Working with the Web Services Add-in on page 704
- ▶ Understanding the Web Service Testing Wizard on page 706
- ▶ Checking that Your WSDL Meets WS-I Standards on page 723
- ▶ Using the Web Service Add Object Wizard on page 727
- ▶ Specifying the Web Services Toolkit on page 731
- ▶ Setting Web Services Test Options on page 733
- ▶ Defining Web Service Test or Component Settings on page 736
- ▶ Working with Web Service Operations on page 737
- ▶ Working with Business Process Testing on page 742
- ▶ Analyzing the Results of a Web Service Test on page 743
- ▶ Introduction to HP Service Test and HP Service Test Management on page 746
- ▶ Troubleshooting and Limitations - Web Services on page 747

About the Web Services Add-in

Web services are self-contained, modular, dynamic applications that can be described, published, located, or invoked over the network to create products, processes, and supply chains. They can be local, distributed, or Web-based. Using Web service-specific features and operations help to make Web service scripts easier to read, maintain, enhance, and parameterize, enabling both advanced and novice users to create sophisticated tests on Web services. Web services usually provide a description to ensure that the client uses the data format expected by the server using a language known as *Web Services Definition Language (WSDL)*.

You can use the QuickTest Professional Web Services Add-in to test your Web service using familiar QuickTest functionality, without the need for extensive knowledge of your Web service architecture. For example, you can use QuickTest Professional to invoke the operations of your Web service and verify returned XML data using special functionality that has been customized for Web services. You can use the specialized `WebService` test object operations to control the way in which QuickTest communicates with your service, including working with security, configuration, headers, and attachments.

You can also use the Web Services Add-in together with any other QuickTest add-ins to create tests or components that test both the direct communications with your Web service and the front-end application that reflects the results of these communications.

In addition to the functionality available in the QuickTest Web Services Add-in, HP provides full SOA testing capabilities with HP Service Test and HP Service Test Management. For more information, see "Introduction to HP Service Test and HP Service Test Management" on page 746.

This chapter explains how to use QuickTest to create and run tests and components on Web services. Additionally, you can review the Web service-related articles in the HP Software Self-solve knowledge base.

Considerations for Working with the Web Services Add-in

- ▶ You can add a WebService test object to the object repository using the Web Service Add Object Wizard (described on page 727) or the Web Service Testing Wizard (described on page 706).
 - ▶ The Web Service Add Object Wizard enables you to automatically create a WebService test object. You can then add steps manually.
 - ▶ When working with tests, you can use the Web Service Testing Wizard to automatically create a WebService test object and to generate steps to test the operations that your Web service supports based on a supplied WSDL file. The Web Service Testing Wizard also enables you to automatically generate checkpoints for these steps.
- ▶ QuickTest Professional supports keyword-driven testing on your Web services. This means that after you add a WebService test object to your object repository, you can access all of the operations that the Web service supports from the **Operation** column of the Keyword View, the **Operation** box in the Step Generator, or by using IntelliSense in the Expert View.

In IntelliSense and in the Step Generator, you can view the operations supported by the Web service in the upper part of the box. These are the operations that were derived from the WSDL source. QuickTest WebService test object operations are displayed in the lower part of the box.

- ▶ You can enhance your test manually by entering operations in the Expert View. For more information on working with the Expert View, see the *HP QuickTest Professional User Guide*.
- ▶ When working with components, QuickTest provides commonly used Web service functions in the **Web_Services.txt** function library. You can use these functions if the function library is associated with your component's application area or your test.
- ▶ You (or an Automation Engineer) can wrap additional Web service-specific functionality in other function libraries associated with the component's application area. For more information on working with function libraries and application areas, see the *HP QuickTest Professional for Business Process Testing User Guide*.

- ▶ When working with multiple toolkits, it is recommended to create a separate test or component for use with each required toolkit. For information on specifying a toolkit for your test or component, see "Setting Web Services Test Options" on page 733 and "Defining Web Service Test or Component Settings" on page 736.
- ▶ If your Web service is protected by a firewall, or your Web service requests are routed via a proxy server, you may need to insert a **SetProxy** statement to set the required proxy information for the service before the first step containing an operation on the corresponding WebService test object. For more information, see the **Web Services** section of the *HP QuickTest Professional Object Model Reference*. If you are working with components, you or an Automation Engineer may need to wrap this functionality in a user-defined function and associate the function library with your application area.
- ▶ The Web Services Add-in is compliant 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>

Understanding the Web Service Testing Wizard

The Web Service Testing Wizard helps you select the WSDL source, set security options, and specify the Web service, port, and operations you want to test. For tests, you can also choose to automatically insert checkpoints with the proper syntax for each selected operation. When you finish, the wizard creates a WebService test object that represents the Web service and port you want to test and inserts the relevant steps directly into your test or component.

You can then update the generated steps of your test or component by replacing the generated argument values with valid values, updating the expected values, and selecting the nodes you want to check in your checkpoints (tests only). After you perform these steps, you can update the data by performing an update run (**Automation > Update Run Mode**). For more information, see the section on updating a test in the *HP QuickTest Professional User Guide*. You can also enhance the generated test or component by inserting additional steps or, for tests, adding programming logic.

Note: QuickTest includes Maintenance Run Mode, which is not supported for applications such as Web services, which do not have a user interface.

After you associate the **Web_Services.txt** function library with your component's application area or your test, you can also perform verification operations similar to checkpoints, and you can insert a wide range of Web service security operations in your component steps. For more information, see "Working with Web Service Operations" on page 737.

Note: If you want to automatically add a WebService test object to your object repository, and then generate steps manually for the operations it supports, you can use the Web Service Add Object Wizard (described on page 727).

To open the Web Service Testing Wizard:

Click the **Web Services Wizard** button or select **Automation > Web Service Testing Wizard**. The Web Service Testing Wizard opens.

The Web Service Testing Wizard includes the following screens:

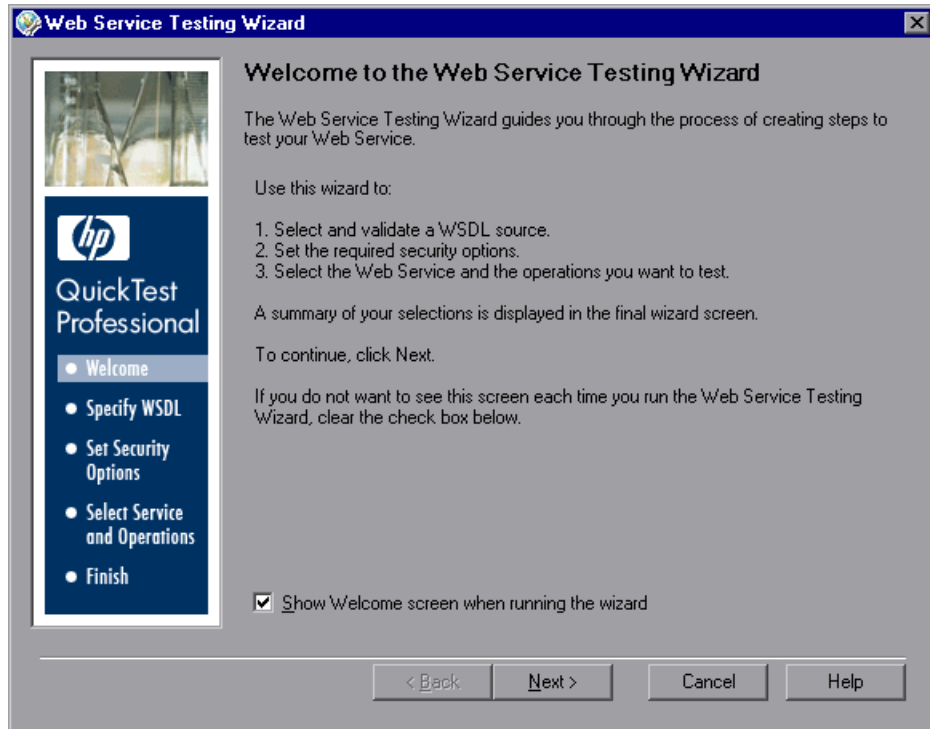
- ▶ Welcome to the Web Service Testing Wizard Screen. Provides an overview of the wizard steps.
- ▶ Web Service Testing Wizard - Specify WSDL for Scanning Screen. Enables you to select the source of the test object you want to include in the generated steps.
- ▶ Web Service Testing Wizard - Set Security Options Screen. Enables you to select the security tokens required for communication with the Web service you want to test, and to set their property values.
- ▶ Web Service Testing Wizard - Select Service and Operations Screen. Enables you to select the WSDL service you want to test and the service operations you want to include in your generated test.
- ▶ Web Service Testing Wizard - Summary Screen. Displays a summary of your selections and for tests, enables you to choose whether to add checkpoints after each relevant operation step.

When you click **Finish**, the basic test is created. You complete it by entering relevant operation argument values and by setting checkpoint expected values (for tests) and other preferences. For more information, see "Completing and Enhancing Your Generated Test" on page 722.

Note: Running your test or component without entering valid argument values may cause the test to fail.

Welcome to the Web Service Testing Wizard Screen

The Welcome to the Web Service Testing Wizard screen is the first screen displayed when you open the wizard. The screen describes the steps in the wizard.

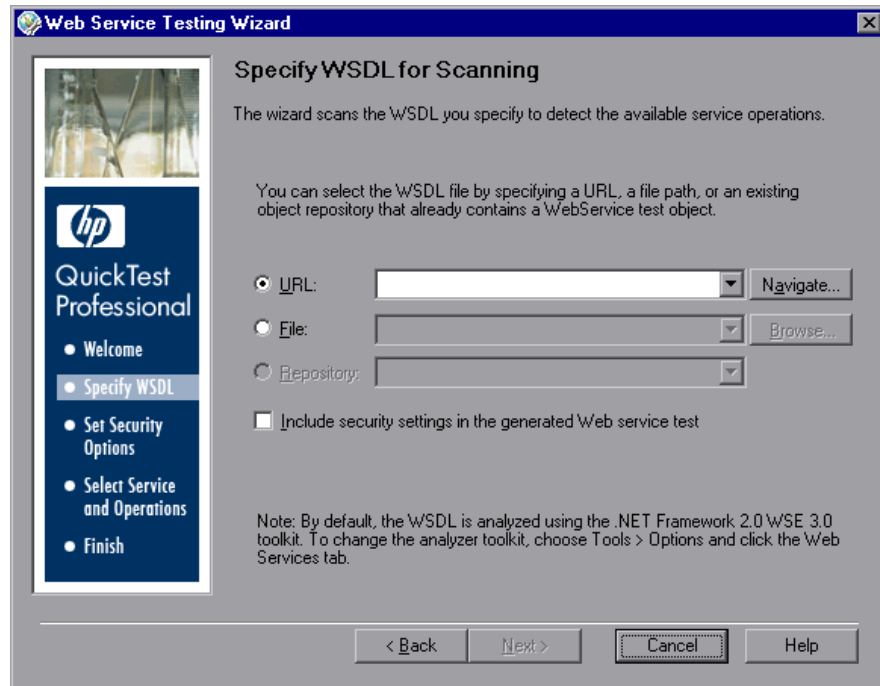


You can choose not to display the welcome screen when you open the wizard. Clear the **Show Welcome screen when running the wizard** check box in this screen, or select the **Tools > Options > Web Services** node in the QuickTest window and then clear **Show Welcome step in Web Service Testing Wizard** in the Web Services pane of the Options dialog box. For more information on the Web Services pane, see "Setting Web Services Test Options" on page 733.

Click **Next** to continue to the Web Service Testing Wizard - Specify WSDL for Scanning Screen.

Web Service Testing Wizard - Specify WSDL for Scanning Screen

The Specify WSDL for Scanning screen enables you to select the required WSDL source. The source can be a URL, a WSDL file, or an existing WebService test object from the object repository.



Notes:

- The first time you open the Web Service Testing Wizard, the URL box is empty. On subsequent uses of the wizard, the Specify WSDL for Scanning screen opens with the same settings as those set in the previous wizard session.
- By default, the WSDL source is analyzed using the .NET Framework 2.0 WSE 3.0 toolkit. You can change the toolkit if required. For more information, see "Specifying the Web Services Toolkit" on page 731.

In the Specify WSDL for Scanning screen, specify the WSDL source and whether you want to include security settings in your Web service test, as follows:

Select a radio button according to the WSDL source you want to test. Then enter the source. You can click the down arrow next to each box to view and select recently used items.

- ▶ If you want to locate a URL source on a Web server, click the **Navigate** button next to the **URL** box to open Microsoft Internet Explorer. The button name changes to **Capture**. Navigate to the required URL (WSDL file). Minimize the browser and click **Capture** or close the browser. The URL address is automatically entered in the **URL** box.
- ▶ If you want to locate a WSDL file, click **Browse** next to the **File** box to open the Browse for WSDL File dialog box. Browse to the required file.

If you are currently connected to a Quality Center project, you can toggle between the file system and the test plan tree for the Quality Center project by clicking the **File System** or **Quality Center** buttons.

Tip: From the **Attachments of type** list in the Browse for WSDL File dialog box (or **Files of type** list when choosing a file in the file system), you can choose to view only **.wsdl** files, only **.xml** files, or view all the files in the selected location.

- ▶ If you want to create steps for a service (and port) that has already been defined as a test object in one of the repositories associated with the current action or component (via its application area), select **Repository**, and then select the relevant test object. This option enables QuickTest to access the WebService object directly, without processing the WSDL file, thus saving processing time as QuickTest identifies the service and its operations. (This in turn enables the next screen to open faster.)

Select the **Include security settings in the generated Web service test** check box if you want to specify the security tokens that are required for communication with the Web service you want to test.

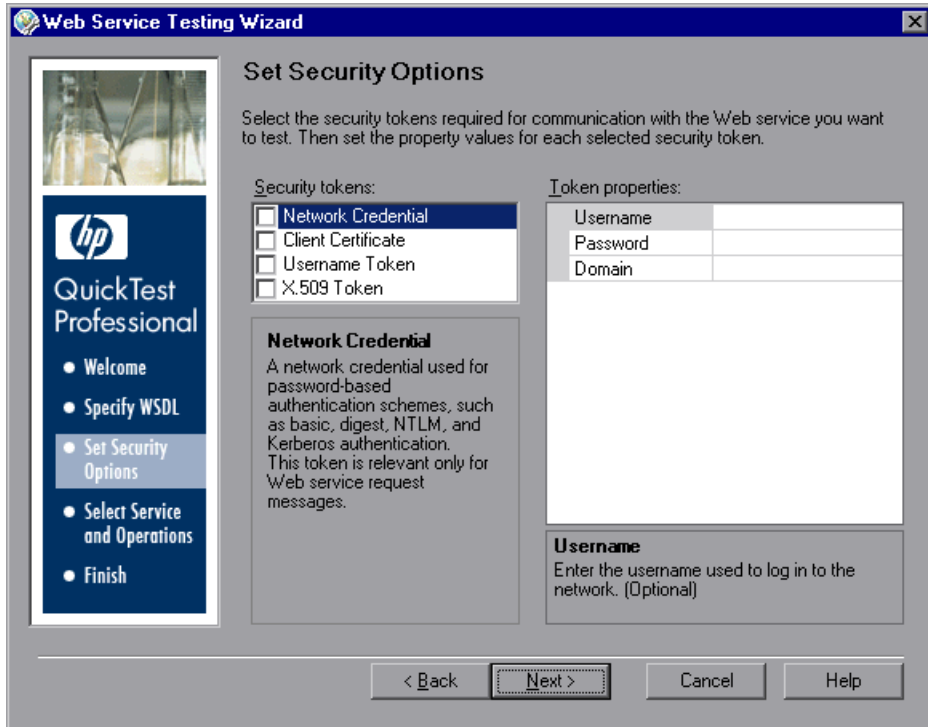
Click **Next**. Depending on your selections in the Specify WSDL for Scanning screen, one of the following occurs:

- ▶ If you specified a secure WSDL, the Network Credentials dialog box opens. Enter the login details required to access the WSDL and click **OK**. The Web Service Testing Wizard - Set Security Options Screen opens. The Network Credentials security token is selected and the token properties contain the credentials you specified. When QuickTest runs the test created by this wizard, the credentials saved with the test are used to access both the WSDL and the service.
- ▶ If you selected the **Include security settings in the generated Web service test** check box, the Web Service Testing Wizard - Set Security Options Screen opens.
- ▶ If you do not need to specify security settings, the Web Service Testing Wizard - Select Service and Operations Screen opens (described on page 718). Note that opening the next screen may take some time, depending upon the selected option. While scanning the WSDL, QuickTest displays a cyclic progress bar that runs until the scanning process is complete.

Note: The Set Security Options screen is available only for tests. Therefore, if you are working with components, when you click **Next**, the Web Service Testing Wizard - Select Service and Operations Screen opens (described on page 718). If required, you can insert steps containing security operations after you complete the wizard. These operations are available from the associated **Web_Services.txt** function library. For information on associating this function library with your component's application area or with your test, see the *HP QuickTest Professional for Business Process Testing User Guide* or the *HP QuickTest Professional User Guide* respectively.

Web Service Testing Wizard - Set Security Options Screen

When working with tests, the Set Security Options screen enables you to specify the security tokens that are needed for communication with the Web service you want to test. For each selected security token, you also set its property values.



After you complete the wizard, QuickTest converts your selections into one or more test steps, containing the corresponding operations and property values. For example, if you select **X.509 Token**, QuickTest might add steps similar to the following:

```
tokenID = WebService("FlightNetWebService").Security.AddX509Token
(micRequestToken, XMLWarehouse("Certificate"))
```


If you select **Client Certificate** or **X.509 Token** in the Security tokens area, the wizard loads the certificate that you specify in the Token properties area and saves it with the test. The wizard stores the raw data of the certificate as a binary string in a new XML structure. During a run session, QuickTest uses this certificate and does not load it from the external source that you specified. You can access the certificate from the XML Warehouse pane in the Settings dialog box (**File > Settings > XML Warehouse** node). For more information on XML structures, see "Working with XML Structures" on page 753.

Considerations for Working with Security Tokens

- ▶ If you specify a certificate installed on the computer (**StoreType = Store**), and the certificate contains a non-exportable private key, the wizard cannot save the certificate with the test. Instead, the wizard saves a reference to the location of the certificate, enabling QuickTest to locate the certificate during a run session. Therefore, before you run the test, you must make sure that the certificate is installed in the location that you specified when creating the test.
- ▶ If you need to replace the certificate, run the wizard again to create a new XML structure containing the required certificate and add the relevant steps. Edit the test or component manually to remove the steps that used the old certificate.
- ▶ To instruct QuickTest to load the certificate from a file or a certificate store for each run session, manually create steps that use the **LoadX509CertificateFromFile** or **LoadX509CertificateFromStore** methods. These methods return an XMLData object that you can use as an argument for subsequent steps. For more information, see the **Web Services** section of the *HP QuickTest Professional Object Model Reference*.

The Set Security Options

In the **Security tokens** area, select the check boxes for the security tokens needed for communication with the Web service you want to test. You can highlight any token to view its description in the **Description** area. When you highlight a token, its properties are displayed in the **Token properties** area. (Note that highlighting the token does not select its check box.) You can select as many security tokens as needed.

The following security tokens are available:

- ▶ Network Credential, described on page 714
- ▶ Client Certificate, described on page 714
- ▶ Username Token, described on page 715
- ▶ X.509 Token, described on page 716

Network Credential

A network credential used for password-based authentication schemes, such as basic, digest, NTLM, and Kerberos authentication. This token is relevant only for Web service request messages.

The **Network Credential** token includes the following properties:

- ▶ **Username.** The user name used to log in to the network.
- ▶ **Password.** The password used to log in to the network.
- ▶ **Domain.** The network domain name. (Optional)

Client Certificate

A client certificate used mostly when a client uses the SSL3.0/PCT1 protocol to connect to a server, and the server requires client certificates for mutual authentication.

The **Client Certificate** token includes the following properties:

- ▶ **StoreType.** Indicates whether the certificate is located in the file system or installed on the computer.
Available options: **File** and **Store**.

If you select **File**, the following **FileCertificate** properties are displayed:

- ▶ **FileName.** The path of the file that contains the certificate data. Enter the file name or click the browse button to locate the certificate file.
- ▶ **Password.** The password required to access the certificate file.
(Optional)

If you select **Store**, the following **StoreCertificate** properties are displayed:

- ▶ **Location.** Indicates whether the certificate is installed for the current user or for anyone using the computer.
Available options: **CurrentUser** and **LocalMachine**.
- ▶ **Store.** The store in which the certificate is located. QuickTest displays a list of standard locations for installed certificates. If the certificate that you need to use in the test is located elsewhere, use the **LoadX509CertificateFromStore** method. For more information see the **Web Services** section of the *QuickTest Professional Object Model Reference* (**Help > QuickTest Professional Help**).
- ▶ **Certificate.** The client certificate to use for authentication. To select a certificate, click the browse button. The Select Certificate dialog box opens, displaying all of the certificates installed in the store you specified. Select a certificate and click **OK**.

Username Token

Username and password security credentials.

The **Username** token includes the following properties:

- ▶ **TokenDirection.** Indicates whether the security token should be appended to all subsequent request messages or all subsequent response messages.
Available options: **Request** and **Response**.
- ▶ **Username.** The user name used to sign or encrypt SOAP messages.
- ▶ **Password.** The password used to sign or encrypt SOAP messages.

- ▶ **SendMode.** Indicates how the password should be sent.
Available options: **None**, **PlainText**, and **Hashed**.
- ▶ **ProtectionMode.** Indicates the protection mode to be applied to all subsequent Web service operations.
Available options: **None**, **Signature**, **Encryption**, and **Both**.

X.509 Token

An X.509 certificate used for signing and/or encrypting Web service request messages when a server's public certificate is required.

The **X.509 token** includes the following properties:

- ▶ **TokenDirection.** Indicates whether the security token should be appended to all subsequent request messages or all subsequent response messages.
Available options: **Request** and **Response**.
- ▶ **StoreType.** Indicates whether the certificate is located in the file system or installed on the computer.
Available options: **File** and **Store**.

If you select **File**, the following **FileCertificate** properties are displayed:

- ▶ **FileName.** The path of the file that contains the certificate data. Enter the file name or click the browse button to locate the certificate file.
- ▶ **Password.** The password required to access the certificate file.
(Optional)

If you select **Store**, the following **StoreCertificate** properties are displayed:

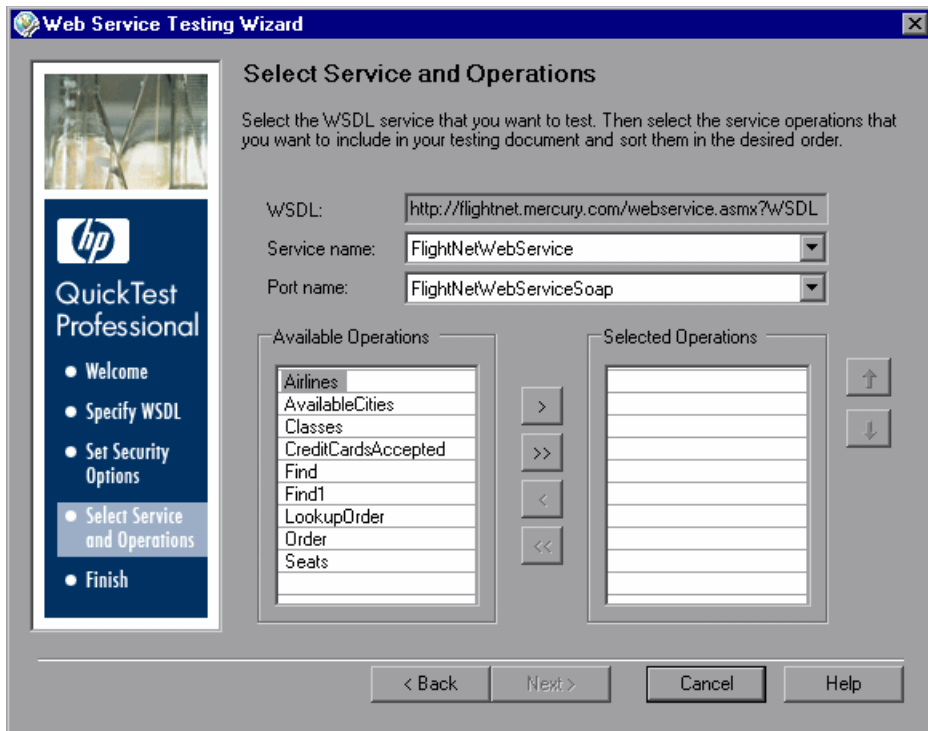
- ▶ **Location.** Indicates whether the certificate is installed for the current user or for anyone using the computer.
Available options: **CurrentUser** or **LocalMachine**.
- ▶ **Store.** The store in which the certificate is located. QuickTest displays a list of standard locations for installed certificates. If the certificate that you need to use in the test is located elsewhere, use the **LoadX509CertificateFromStore** method. For more information see the **Web Services** section of the *QuickTest Professional Object Model Reference* (**Help > QuickTest Professional Help**).
- ▶ **Certificate.** The client certificate to use for authentication. To select a certificate, click the browse button. The Select Certificate dialog box opens, displaying all of the certificates installed in the store you specified. Select a certificate and click **OK**.
- ▶ **ProtectionMode.** Indicates the protection mode to be applied to all subsequent Web service operations.
Available options: **None**, **Signature**, **Encryption**, and **Both**.

Set the properties for each selected security token by highlighting the token in the **Security tokens** area and modifying its property values in the **Token properties** area. When you highlight a property, its description is displayed below the **Token properties** area.

Click **Next** to continue to the Web Service Testing Wizard - Select Service and Operations Screen.

Web Service Testing Wizard - Select Service and Operations Screen

The Select Service and Operations screen enables you to select a service and a port from the source WSDL you specified in the Specify WSDL screen (described on page 709). The test object that the wizard generates represents this specific service and port. You can then select the operations to be included in the test or component steps from the list of available operations supported by that Web service. You can also arrange them in the required testing order.



From the **Service name** list and **Port name** list, select the service and port that you want to test. The name of the service you select is also used as the default name of the created test object. If a description of the Web service is available, it is displayed as a tooltip when the cursor is positioned over the service name.

Tip: You can rename a WebService test object in the Object Repository. Select **Resources > Object Repository** to open the repository. Then select the test object, right-click, and select **Rename** from the menu.

Note: The list of ports displays all of the ports in the selected service that work with a supported protocol. For a list of supported protocols, see the *HP QuickTest Professional Product Availability Matrix*, available from the Documentation Library Home page or the root folder of the QuickTest Professional DVD.

In the **Available Operations** list, double-click each operation you want to include, or select an operation and click the right arrow button > to add it to the **Selected Operations** list. You can add the same operation more than once.

To remove an operation from the **Selected Operations** list, you can double-click it in the **Selected Operations** list or select the operation and click the left arrow button <.

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



Use the up and down arrows to sort the **Selected Operations** list into the required testing order.

Important Information for Microsoft .NET Framework WSE Toolkit Users

If you are working with a .NET Framework WSE toolkit, and one or more Web service operations in your WSDL require you to provide parameter data in a SOAP header, you can use the **SetHeaderField_<FieldName>Value** operation to do so. (<FieldName> represents the name of the .NET field in which the parameter data is stored and **Value** is appended to the field name.)

During a run session, this operation instructs QuickTest to save the header field value you specify, as if you are working directly with a .NET client. Thereafter, QuickTest inserts this header field value every time it sends a message requiring this header. If a response message updates this header field value, the updated header field value is saved and is used for all subsequent method calls.

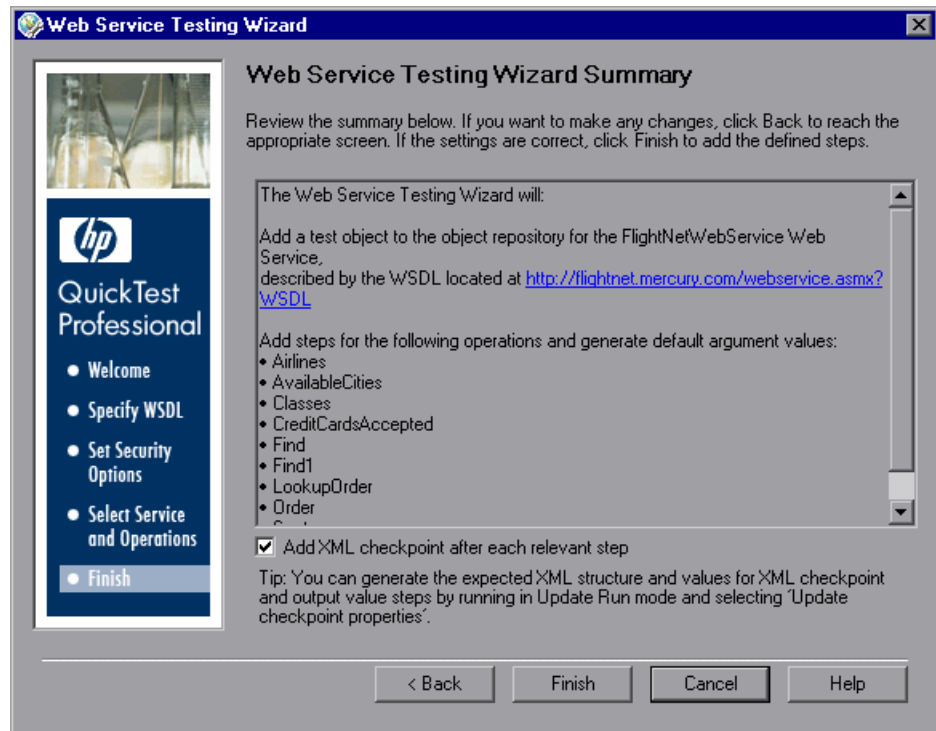
In the **Available Operations** list, select any operation beginning with **SetHeaderField_** and move it to the **Selected Operations** list. Then move this operation to the top of the **Selected Operations** list (or at least above any operations that require parameterized header data).

For more information on the **SetHeaderField_** operation, see "Accessing Operations Supported by the Web Service You Are Testing" on page 740.

Click **Next** to continue to the Web Service Testing Wizard - Summary Screen.

Web Service Testing Wizard - Summary Screen

The Web Service Testing Wizard Summary screen provides a summary of the operations for which the wizard will add steps according to your selections.



When working with tests, you can automatically insert XML checkpoints by selecting **Add XML checkpoint after each relevant step** (selected by default). Selecting this check box adds an XML checkpoint for each step in the test that has a return value or an output argument. For more information on XML checkpoints, see the *HP QuickTest Professional User Guide*.

Review the summary. If the settings are correct, click **Finish**. The WebService test object is stored in the local object repository, and the defined steps are converted to the proper syntax and inserted into your test or component.

Completing and Enhancing Your Generated Test

The Web Service Testing Wizard accelerates the process of designing a basic test that checks the operations that your Web service supports.

For each operation, the wizard generates default argument values with the proper value types. For each automatically created checkpoint (tests only), the wizard generates a generic XML structure as a place holder for the expected XML return values. Before you can run your test or component, you must replace the default values with the appropriate values for your test.

Review each of the argument values and supply a value that is appropriate for the operation. For argument values of type XML, the wizard generates an XML structure. Use the **Configure the value** button in the Keyword View **Value** column to open the XML structure and edit the generated values within the structure. Alternatively, you can open the XML structure from the XML Warehouse pane in the Test Settings or Business Component Settings dialog box (**File > Settings > XML Warehouse** node). For more information on XML structures, see "Working with XML Structures" on page 753. After you modify the required values, you need to update the data by performing an update run (**Automation > Update Run Mode**). For more information, see the section on updating a test in the *HP QuickTest Professional User Guide*.

Open each automatically generated checkpoint, populate the XML tree with the expected return values, and select the items that you want to check. For more information, see "Checking XML" on page 750 and the Checking XML chapter in the *HP QuickTest Professional User Guide*.

You may want to output values returned by one step for use as input in another step. For more information, see "Outputting XML Values" on page 752, and the XML Output Values section in the *HP QuickTest Professional User Guide*.

You can also add steps using the test object operations (methods and properties) that the QuickTest Professional Web Services Add-in supplies to test the behavior of your Web service. For more information, see "Using QuickTest WebService Test Object Methods and Properties" on page 738, and the **Web Services** section of the *HP QuickTest Professional Object Model Reference*.

Checking that Your WSDL Meets WS-I Standards

You can instruct QuickTest to analyze a specific WSDL source to check that it conforms to WS-I standards and complies with the WS-I Basic Profile.

Note: QuickTest performs the validation using the WS-I validation tool, which is a third-party application that is not provided with QuickTest. You can download **Interoperability Testing Tools 1.1** from the Web Services Interoperability Organization Web site at <http://www.ws-i.org>. Note that it must be installed locally.

You can run the validation tool manually in QuickTest, selecting **Tools > Validate WSDL** to open the Validate WSDL dialog box. You can also validate that the WSDL source conforms to WS-I standards programmatically using either the **WebService.ValidateWSDL** method or the **WSUtil.ValidateWSDL** method. For more information, see the **Web Services** section of the *HP QuickTest Professional Object Model Reference*.

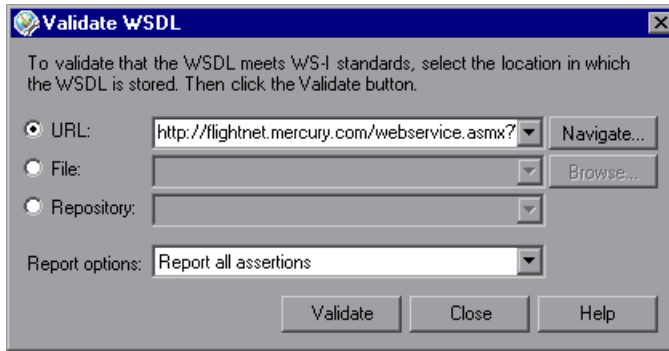
When the validation is complete, you can view the validation results, which indicate whether the file conforms to WS-I profile guidelines. Note that it is possible that your Web service tests or components on the specified WSDL will run successfully even if they do not conform to these guidelines.

Before you can use the WS-I validation tool, you need to specify the location of the WS-I validation tool. You do this in the Web Services pane of the Options dialog box (**Tools > Options > Web Services** node). For more information, see "Setting Web Services Test Options" on page 733.

Note: When you run the WS-I validation tool, QuickTest accesses the WSDL. If the WSDL you are validating is located on a secure server, or if the network connection is secure, you must save the WSDL and any additional resources referenced by the WSDL to a non-secure location (such as a local drive) before running the WS-I validation tool.

To check that your WSDL meets WS-I standards:

- 1 Select **Tools > Validate WSDL** or press ALT+T+L. The Validate WSDL dialog box opens.



- 2 Specify the location in which the WSDL is stored by selecting a radio button according to the WSDL source you want to test. Then enter the source. You can click the down arrow next to each box to view and select recently used items.

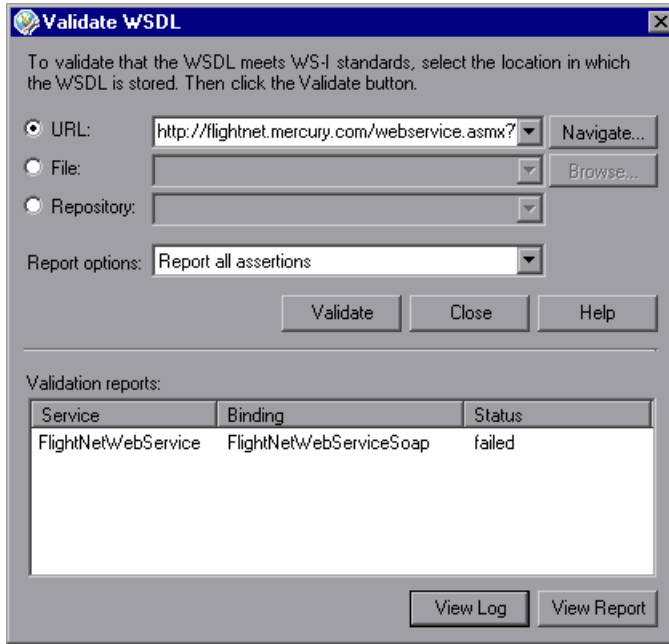
- ▶ To specify a WSDL from a URL source on a Web server, select **URL**. Enter the URL address manually or click the **Navigate** button next to the **URL** box to open your default browser. The button name changes to **Capture**. Navigate to the required URL. Minimize the browser and click **Capture** or close the browser. The URL address is automatically entered in the **URL** box.
- ▶ To specify a WSDL file, select **File**. Enter the file path manually or click **Browse** next to the **File** box to open the Browse for WSDL File dialog box. Browse to the required file.

If you are currently connected to a Quality Center project, you can toggle between the file system and the test plan tree of the Quality Center project by clicking the **File System** or **Quality Center** buttons.

Tip: From the **Attachments of type** list in the Browse for WSDL File dialog box (or **Files of type** list when choosing a file in the file system), you can choose to view only **.wsdl** files, only **.xml** files, or view all the files in the selected location.

- To specify a WSDL that defines a service for which a test object has already been created in one of the repositories associated with the current action (or component), select **Repository**, and then select the relevant test object. QuickTest locates the WSDL according to the location specified in the **wsdl** property of the WebService test object.
- 3** Click the **Report options** down arrow and select the criteria for the assertion results to be included in the validation report. (Test assertions are used by the validation tool to analyze whether a Web service conforms to WS-I standards.)
 - **Report all assertions.** Reports the results of all assertions.
 - **Report all assertions but passed assertions.** Reports the results of all assertions except those that have a "passed" result.
 - **Report only failed assertions.** Reports the results of assertions that have a "failed" result only.
 - 4** Click the **Validate** button to analyze the specified WSDL source and check that it conforms to WS-I standards. The validation is performed for each binding defined in the WSDL. This process might take some time.

When the check is complete, the Validate WSDL dialog box expands to display the results of the validation check in the **Validation reports** area. This area lists the reports generated by the WS-I testing tool and indicates the status of each report.



5 Highlight a report in the **Validation reports** area:

- ▶ Click the **View Log** button to view the WS-I testing tool log for the selected report item.
- ▶ Click the **View Report** button to open the WS-I Profile Conformance Report for the selected report item in a Web browser.

Using the Web Service Add Object Wizard

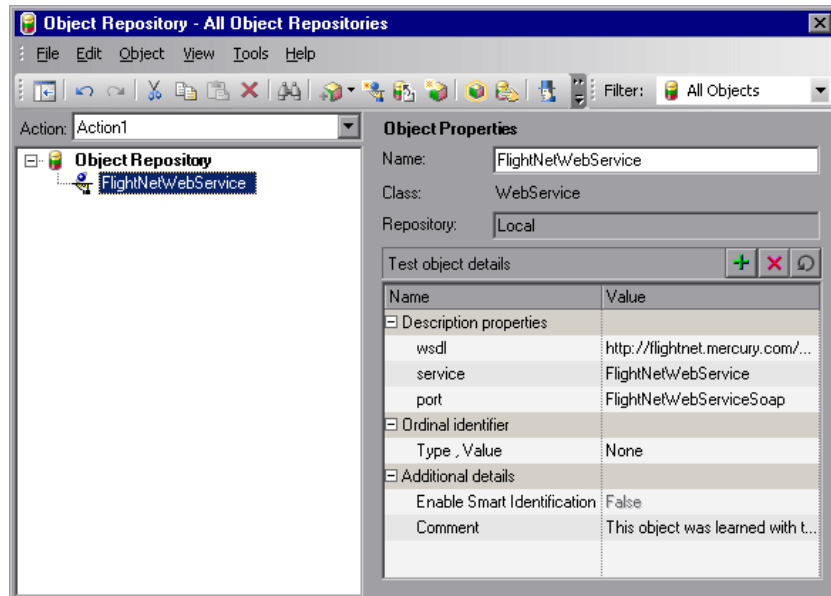
The Web Service Add Object Wizard enables you to add WebService test objects to your object repository. You select the WSDL source, the Web service, and the port you want to test, and the wizard creates a WebService test object. You can then use the new test object to add steps to your test or component.

Note: If you want to create a WebService test object and automatically generate steps for the operations it supports, you can use the Web Service Testing Wizard. For more information, see "Understanding the Web Service Testing Wizard" on page 706.

To open the Web Service Add Object Wizard:



- 1 Click the **Object Repository** toolbar button, or select **Resources > Object Repository**. The Object Repository dialog box opens.





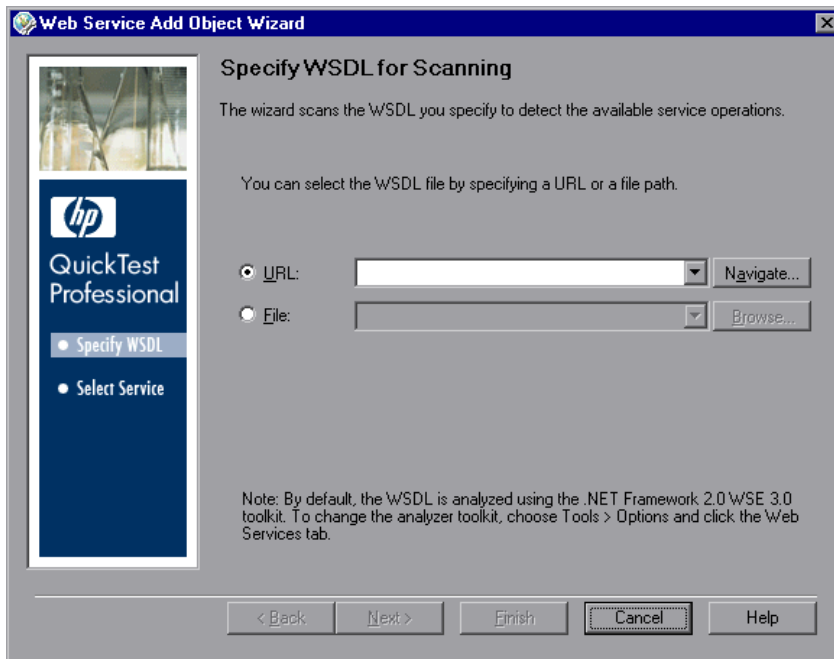
- 2 Click the **Web Service Add Object Wizard** button or select **Object > Web Service Add Object Wizard**. The Web Service Add Object Wizard opens to show the Specify WSDL for Scanning screen.

The Web Service Add Object Wizard includes the following screens:

- ▶ Add Object Wizard - Specify WSDL for Scanning Screen. Enables you to select and validate the source of the test object you want to create.
- ▶ Add Object Wizard - Select Service Screen. Enables you to select the WSDL service for which you want to create a test object.

Add Object Wizard - Specify WSDL for Scanning Screen

The Specify WSDL for Scanning screen is the first screen displayed when you open the Add Object wizard. This screen enables you to specify a URL or file as the WSDL source.



Notes:

- ▶ The first time you open the Web Service Add Object Wizard, the URL box is empty. On subsequent uses of the wizard, the Specify WSDL for Scanning screen opens with the same settings as those set in the previous wizard session.
 - ▶ By default, the WSDL source is analyzed using the .NET Framework 2.0 WSE 3.0 toolkit. You can change the toolkit if required. For more information, see "Specifying the Web Services Toolkit" on page 731.
-

Select a radio button according to the WSDL source you want to test. Then enter the source. You can click the down arrow next to each box to view and select recently used items.

- ▶ If you want to locate a URL source on a Web server, click the **Navigate** button next to the **URL** box to open your default browser. The button name changes to **Capture**. Navigate to the required URL. Minimize the browser and click **Capture** or close the browser. The URL address is automatically entered in the **URL** box.
- ▶ If you want to locate a WSDL file, click **Browse** next to the **File** box to open the Browse for WSDL File dialog box. Browse to the required file.

If you are currently connected to a Quality Center project, you can toggle between the file system and the test plan tree for the Quality Center project by clicking the **File System** or **Quality Center** buttons.

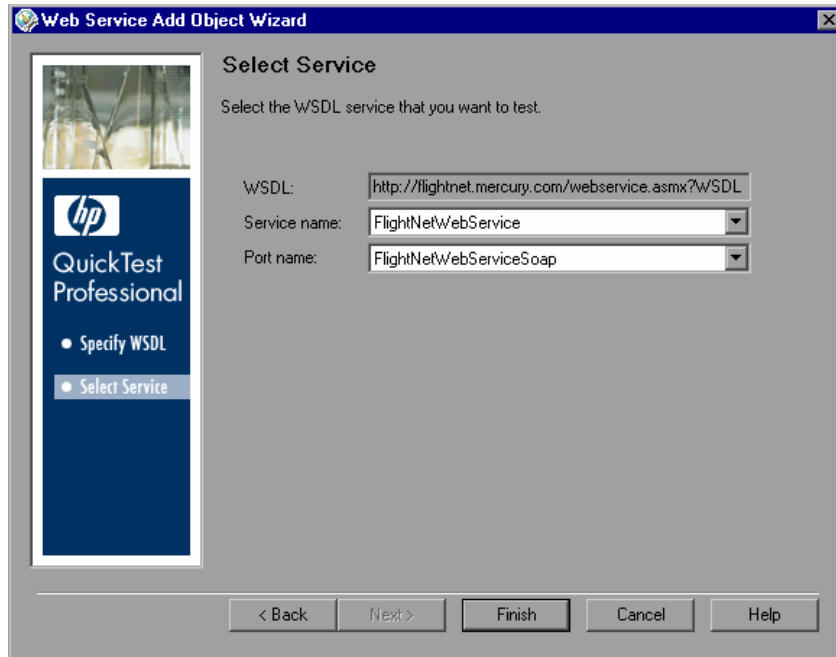
Tip: From the **Attachments of type** list in the Browse for WSDL File dialog box (or **Files of type** list when choosing a file in the file system), you can choose to view only **.wsdl** files, only **.xml** files, or view all the files in the selected location.

Click **Next** to continue to the Add Object Wizard - Select Service Screen or click **Finish** if you want the Wizard to create your test object using the first Web service and the first port described in the WSDL.

Note: If you specified a secure WSDL, the Network Credentials dialog box opens. Enter the login details required to access the WSDL and click **OK**.

Add Object Wizard - Select Service Screen

The Select Service screen in the Add Object wizard enables you to select the Web service and port that you want to test.



From the **Service name** list and the **Port name** list, select the service and port that you want to test and click **Finish**. The wizard adds a WebService test object to the object repository, representing the specific Web service and port. If a description of the Web service is available, it is displayed as a tooltip when the cursor is positioned over the service name.

Note: The list of ports displays all of the ports in the selected service that work with a supported protocol. For a list of supported protocols, see the *HP QuickTest Professional Product Availability Matrix*, available from the Documentation Library Home page or the root folder of the QuickTest Professional DVD.

Close the Object Repository dialog box. You can now use the test object and its operations in the steps of your test.

Specifying the Web Services Toolkit

Before creating WebService test objects and steps, you can specify the Web services toolkit that you want QuickTest Professional to use when learning new test objects and when running your Web service steps. If you are working with multiple toolkits, it is recommended to create a unique test or component for each toolkit.



The toolkit you specify for learning WebService objects is a global setting used for all Web service tests. You set this option in the Web Services pane of the Options dialog box (click the **Options** toolbar button, or select **Tools > Options > Web Services** node). By default, QuickTest uses the Microsoft .NET Framework 2.0 WSE 3.0 toolkit when learning WebService objects and when running Web service tests and components. You can change the default, as described in "Setting Web Services Test Options" on page 733.



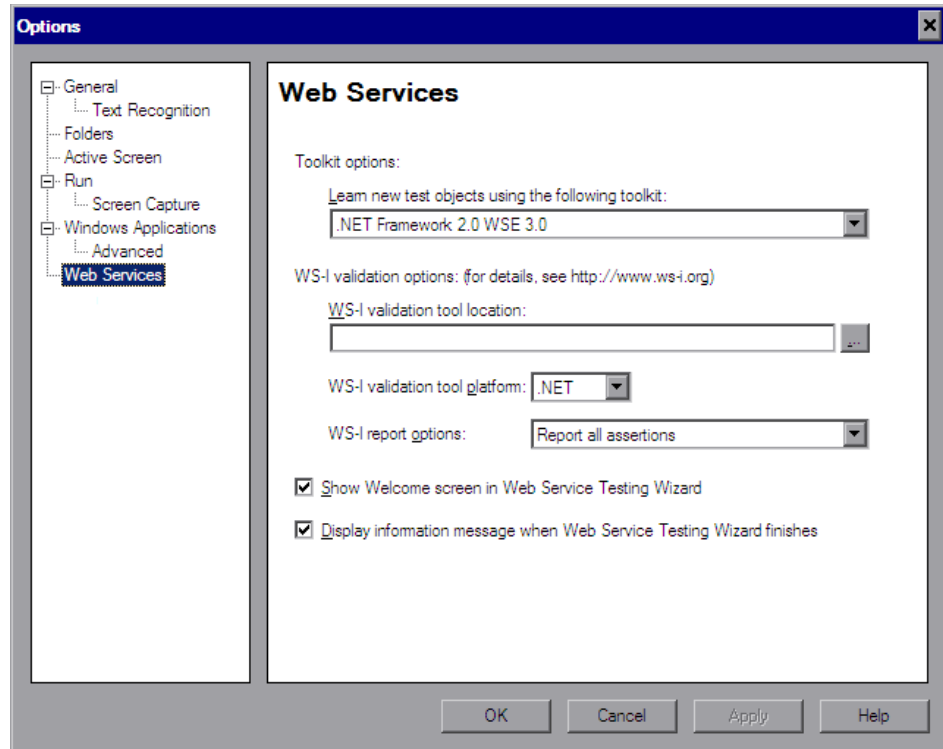
The toolkit you specify for running Web service tests and components is a local setting that is specific to a test or an application area. It is recommended to use the same toolkit you specify for learning WebService objects, otherwise your test may fail. For tests, you specify the toolkit in the Web Services pane of the Test Settings dialog box (click the **Settings** toolbar button, or select **File > Settings > Web Services** node). For components, you specify the toolkit in the Web Services pane of the Application Area Settings dialog box (select **File > Settings**, or click the **Settings** toolbar button, or click the **Additional Settings** button in the Application Area General pane).

Before you run a Web service test or component, make sure that the specified toolkit is installed on the computer on which QuickTest is installed. For example, if the test or component is set to run using the Microsoft .NET WSE 2.0 toolkit, make sure that this toolkit is installed on your QuickTest computer.

For more information, see "Defining Web Service Test or Component Settings" on page 736.

Setting Web Services Test Options

The Web services options enable you to specify the toolkit that QuickTest uses when learning WebService test objects, specify WS-I validation preferences, and set additional display settings for the Web Service Testing Wizard.



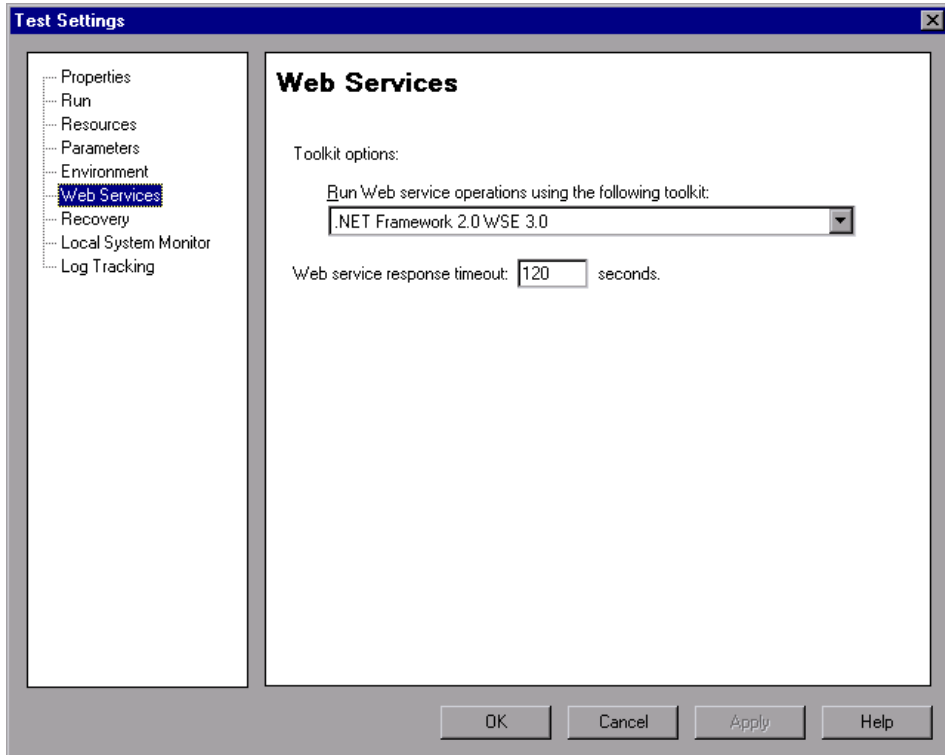
The Web Services pane (**Tools > Options > Web Services** node) includes the following options:

Option	Description
<p>Learn new test objects using the following toolkit</p>	<p>Specifies the toolkit you want QuickTest Professional to use when learning new WebService test objects:</p> <ul style="list-style-type: none"> ➤ .NET Framework 1.1 WSE 2.0 ➤ .NET Framework 2.0 WSE 3.0 ➤ Apache Axis 1.x <p>Note: After you install Microsoft .NET Framework 1.1, you can download and install .NET Framework 1.1 WSE 2.0 SP3 from http://www.microsoft.com/downloads/details.aspx?familyid=8070e1de-22e1-4c78-ab9f-07a7fcf1b6aa&displaylang=en.</p> <p>Tip: It is recommended to run Web service tests using the same toolkit with which the test was created. For information on setting run options, see "Defining Web Service Test or Component Settings" on page 736.</p>
<p>WS-I validation tool location</p>	<p>Indicates the path containing the WS-I validation tool.</p> <p>Specify the root path of the ws-i-test-tools folder (and not the bin folder containing the executable file). The ws-i-test-tools folder must be stored on the computer on which the WSDL is scanned.</p> <p>You can download Interoperability Testing Tools 1.1 for Java and .NET from http://www.ws-i.org.</p> <p>The WS-I validation tool tests Web service interoperability by checking compliance with the WS-I Basic Profile.</p> <p>QuickTest uses the path specified here to perform validation checks in the Validate WSDL dialog box. For more information, see "Checking that Your WSDL Meets WS-I Standards" on page 723.</p>
<p>WS-I validation tool platform</p>	<p>Indicates the Web service client platform type—.NET or Java.</p>

Option	Description
WS-I report options	<p>Specifies the criteria for the assertion results to be included in the validation report. Test assertions are used by the WS-I validation tool to analyze whether a Web service conforms to WS-I standards.</p> <p>The following options are available:</p> <ul style="list-style-type: none"> ▶ Report all assertions. Reports the results of all assertions. ▶ Report all assertions but passed assertions. Reports the results of all assertions except those that have a "passed" result. ▶ Report only failed assertions. Reports only the results of assertions that have a "failed" result. <p>Note: You can use the WS-I validation tool (accessed from the Validate WSDL dialog box—Tools > Validate WSDL) to test Web service interoperability. It does this by checking compliance with the WS-I Basic Profile. The WS-I validation tool, Interoperability Testing Tools 1.1, is a third-party application and is not provided with QuickTest Professional. The tool can be downloaded from the Web Services Interoperability Organization Web site at http://www.ws-i.org and must be installed locally.</p>
Show Welcome screen in Web Service Testing Wizard	<p>Specifies whether to display the Welcome screen when you open the Web Service Testing Wizard. You can also control this option by clearing the Show Welcome screen when running the wizard check box in the Welcome screen.</p>
Display information message when Web Service Testing Wizard finishes	<p>When you select an operation with input arguments while using the Web Service Testing Wizard, the wizard generates default values for the arguments. If this option is selected, then the wizard displays a message when you click Finish to remind you that you must replace the automatically generated argument values with valid values.</p> <p>If you select the Do not show this message again check box inside the warning message, then the Display information message when Web Service Testing Wizard finishes check box in the Options dialog box is automatically cleared. If you want to reactivate the reminder message, select this check box again.</p>

Defining Web Service Test or Component Settings

The Web services settings enable you to define the way in which QuickTest runs Web service steps. (You open the Settings dialog box by selecting **File > Settings** in your test or in your component's application area.)



Note: The example above shows the Web Services pane of the Test Settings dialog box.

The Web Services pane of the Business Component Settings dialog box displays these settings in read-only format. To modify these settings for components, use the Web Services pane of the Application Area Settings dialog box, which provides the same set of options as shown above.

The Web Services pane includes the following options:

Option	Description
Run Web service operations using the following toolkit	<p>Enables you to select the toolkit you want QuickTest to use when running Web service operations. For new tests and components, the default toolkit is the same as the toolkit set in the Web Services pane of the Options dialog box for learning WebService objects.</p> <p>Tip: Make sure you run Web service tests using the same toolkit with which the test was created. You can verify the toolkit by viewing the Comment field in the object repository. (The Comment field is located in the Object Properties pane.)</p> <p>Note: The run toolkit you select is displayed in the Result Details tab of the Run Results window when the top branch is selected in the run results tree.</p>
Web service response timeout	<p>Defines the length of time in seconds that QuickTest attempts to communicate with the Web service during a Web service step. If the time is exceeded, the step fails.</p>

Working with Web Service Operations

A test consists of statements (steps) coded in Microsoft VBScript. These steps are composed of objects, methods, and/or properties that instruct QuickTest to perform operations on, or retrieve information from, your Web service. You can manually insert and edit steps in the Keyword View when working with tests or components, or in the Expert View when working with tests. You can also generate steps automatically using the Web Service Testing Wizard (described on page 706).

You can add steps using the WebService test object, methods, and properties, or the operations of the Web service you are testing.

When you use IntelliSense in the Expert View, or when you select the required operation from the **Operation** column in the Keyword View or the **Operation** box in the Step Generator, the operations are grouped according to the source from which they were derived. The top part of the list contains the operations that are defined in the WSDL. The bottom part of the list contains the WebService message management objects and the operations that are specific to the QuickTest WebService test object.

Using QuickTest WebService Test Object Methods and Properties

The QuickTest Professional Web Services Add-in supplies methods and properties to test the behavior of your Web service including working with message headers and attachments, setting client configurations and proxy information, protecting messages sent from QuickTest by applying security tokens, and more. These QuickTest Web service operations are available in addition to the Web service-specific operations defined in your Web service.

The Web service object model comprises several types of objects: the **WebService** test object, the **WSUtil** object (a utility object), and several **message management objects** (tests only).

Message Management Objects (Expert View)

Message management objects provide methods that enable you to control the way in which QuickTest sends messages to your Web service. A unique message management object exists for each of the following categories: **Attachments**, **Configuration**, **headers**, and **Security**. You return message management objects using the corresponding WebService message management property. For example, the **WebService.Attachments** property returns the **Attachments** object. WebService message management objects are not test objects, and they are not stored in an object repository. Message management objects are available only for tests.

Each message management object supports related message management operations. For example, the **Configuration** object supports configuration-related message management operations, such as **SetClientConfiguration** and **SetProxy**. The **Security** message management object supports security-related operations that enable you to add various types of tokens to a step, and so on.

To fully take advantage of the QuickTest Web service object model, you need to work in the Expert View, as the operations for message management objects are not available in the Keyword View or using the Step Generator.

When inserting steps manually in the Expert View, you can use IntelliSense to add a WebService message management property to a step, and then apply clearly defined operations to that step.

For example, if you add the **Security** WebService message management object, you can select a method for a specific token type, such as SetNetworkCredential, as shown in the following example:

```
WebService("FlightNetWebService").Security.SetNetworkCredential
    "MyUsername", "MyPwd", "MyDomain"
```

Although the WebService message management objects—**Attachments**, **Configuration**, **Headers**, and **Security**—are available in the Keyword View and the Step Generator, their methods and properties must be initially defined in the Expert View. Therefore, it is recommended not to use these WebService message management objects if you prefer to work in the Keyword View, or if you prefer to insert steps using the Step Generator. Instead, you can associate the **Web_Services.txt** function library with your test and use the alternative operations available from that function library. For information on associating function libraries with your test, see the *HP QuickTest Professional User Guide*.

WebService Objects (Keyword View)

If you prefer to create or modify Web service-based test steps in the Keyword View, or if you are working with component steps, you can use the functions defined in the **Web_Services.txt** function library provided with the Web Services Add-in to perform similar types of operations. The operations and arguments available in this function library are more generic than the operations and arguments that can be used in the Expert View. For example, instead of using a method, such as AddX509Token, that indicates a specific token type and whose arguments are relevant for this type of token (such as TokenDirection and X509Data), you would use a general method, such as AddSecurityToken. The values you need to enter for this method's arguments depend on the type of security token you want to use.

To use the operations defined in these function libraries, you must first associate the **Web_Services.txt** function library with your test or with your component's application area. For more information on function libraries, see the *HP QuickTest Professional for Business Process Testing User Guide*.

For more information, see the **Web Services** section of the *HP QuickTest Professional Object Model Reference*.

Accessing Operations Supported by the Web Service You Are Testing

In addition to the test object methods supported for every WebService test object, you can use standard statement completion options for operations supported by your Web service. The Web service operations are automatically added to IntelliSense when the test object is created. You can access these operations in the top part of the IntelliSense list displayed for your WebService test object.

You can also select operations supported by your Web service from the list of operations displayed in the **Operation** column in the Keyword View and in the **Operation** box in the Step Generator. The Web service-specific operations are displayed in the top part of the list. The test object methods supported for every WebService test object are displayed in the bottom part.

Working with the .NET Framework WSE Toolkit

If you are creating a Web service test using a .NET Framework WSE toolkit, and you want to perform one or more Web service operations that require parameter data to be sent in a message header, you need to add this information to your test using a **SetHeaderField_<FieldName>Value** method (where **<FieldName>** represents the name of the .NET field in which the parameter data is stored and **Value** is appended to the field name).

When QuickTest learns a WebService test object using the .NET Framework WSE toolkit, it analyzes the Web service's supported operations and automatically creates a **SetHeaderField_<FieldName>Value** method for each operation that requires parameter data (according to the header elements defined in the corresponding WSDL). For example, for the LicenseInfo header element, QuickTest creates the **SetHeaderField_LicenseInfoValue** method. Note that the *<FieldName>* value is not always identical to the actual header element as it appears in the WSDL.

When you run the WebService Testing Wizard, the Select Service and Operations screen automatically displays these special QuickTest methods in the **Available Operations** list. If you select operations in the Select Service and Operations screen that require parameter data in the header, you should also select the appropriate **SetHeaderField_<FieldName>Value** methods, making sure that the **SetHeaderField_<FieldName>Value** methods are located higher in the **Selected Operations** list than the Web service operations they support.

After a **SetHeaderField_<FieldName>Value** method is called, QuickTest saves the header field value you specify (as if you are working directly with a .NET client). Thereafter, QuickTest inserts this header field value every time it sends a message requiring this header. If a response message updates this header field value, the updated header field value is saved and is used for all subsequent method calls (unless you use the **SetHeaderField_<FieldName>Value** method to modify the header field value again).

You must insert a step with the appropriate **SetHeaderField_<FieldName>Value** method prior to any step containing a Web service operation supported by this method. When you insert a step on a WebService test object that was learned using the .NET Framework WSE toolkit, IntelliSense displays the automatically generated methods in the top part of the operations list, together with the other Web service-specific operations. If you insert the **SetHeaderField_<FieldName>Value** method using the Web Service Testing Wizard, an XML structure parameter is automatically created and populated with default values.

You should edit the XML structure parameter to provide the appropriate data. If you are working in the Keyword View or the Expert View, you must define and populate the XML structure parameter manually. For more information on XML structures, see "Working with XML Structures" on page 753. Note that the **SetHeaderField_<FieldName>Value** method parameter value is always a complex value.

Working with Business Process Testing

When working with Business Process Testing, it is recommended to associate the **Web_Services.txt** function library with your application areas. This function library provides keywords that wrap additional functionality for Web service-based steps, enabling you to perform commonly used Web service operations on any associated component. You can use these operations to perform checks and output values on XML elements, and to perform operations that are similar to the operations available for tests in the Expert View. For more information, see the *HP QuickTest Professional for Business Process Testing User Guide* and the *HP QuickTest Professional Object Model Reference*.

Guidelines When Working with Business Process Testing

You should consider the following points when working with business components in Business Process Testing.

- ▶ If you want to parameterize output values for a component step, use XML structures, or local or component parameters. For more information, see "Parameterizing XML Values" on page 763.
- ▶ During a run session, it is recommended to use the same toolkit that QuickTest used to learn your Webservice test objects. If you want to work with more than one toolkit, you can create separate components for each toolkit. For more information, see "Setting Web Services Test Options" on page 733 and "Defining Web Service Test or Component Settings" on page 736.
- ▶ External resources for components are stored in the associated application area. These include function libraries, object repositories, and recovery scenarios. If you need additional functionality, you or an Automation Engineer can wrap the required functionality in other function libraries and associate them with your application area. It is recommended not to overwrite the existing functionality in any function library supplied with QuickTest, as QuickTest function libraries may be overwritten during an upgrade.

For more information on working with Business Process Testing in QuickTest, see the *HP QuickTest Professional for Business Process Testing User Guide*. For more information on working with Business Process Testing in Quality Center, see the *HP Business Process Testing User Guide*.

Analyzing the Results of a Web Service Test

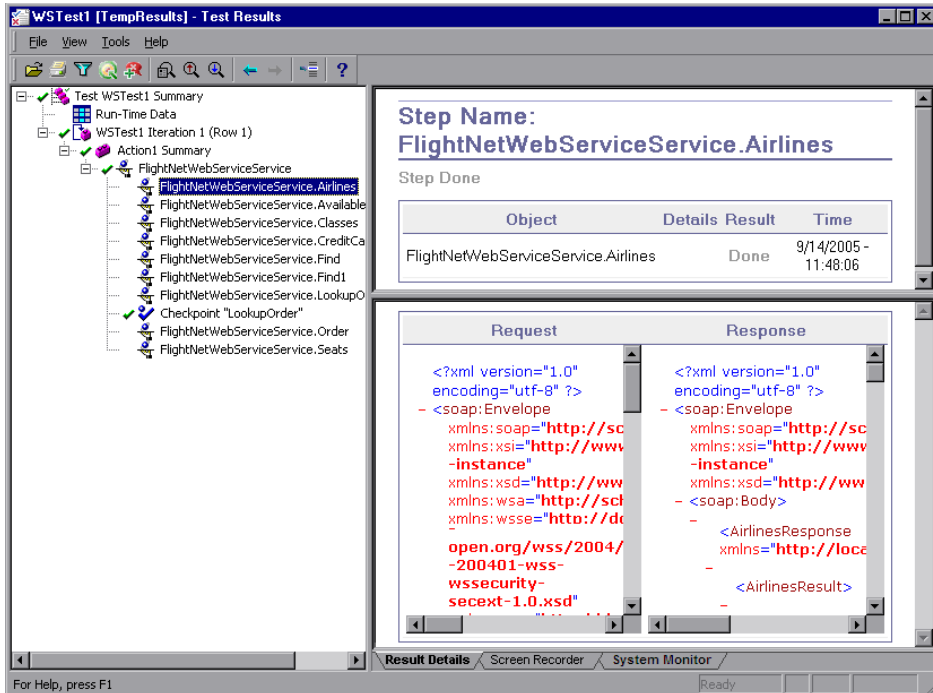
You can view the results of a Web service run session in the Run Results window. The window contains a description of the steps performed during the run, as well as a summary of the run results. By default, the Run Results window opens automatically at the end of a run.



Note: You can choose not to open the Run Results window automatically after each run. Select **Tools > Options** and click the **Run** node. Then clear **View results when run session ends**. You can click the **Results** button or select **Automation > Results** to open the Run Results window when you want to view run results.

If the Web Services Add-in is installed and loaded during a run session, the run toolkit you specified in the Settings dialog box is displayed in the Result Details tab in the Run Results window when the top branch of the tree is selected. The run toolkit is displayed in this tab even if the test or component does not include any Web Services test object.

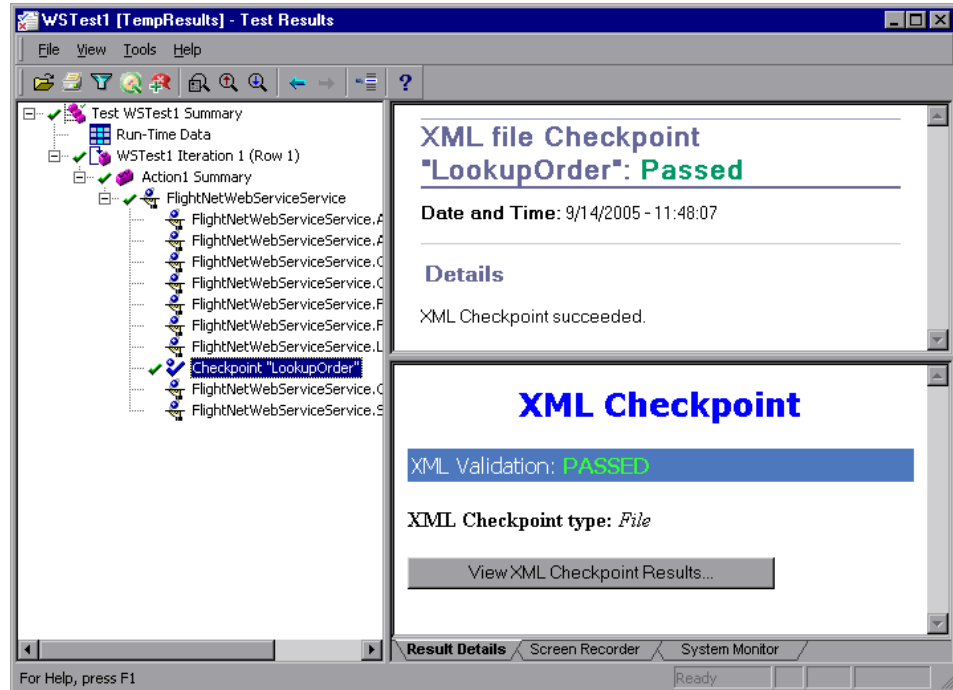
Expand the run results tree in the left pane of the Run Results window.



Note: By default, QuickTest captures the detailed requests and responses for all steps.

The Web Services Add-in uses the **Save step screen capture to results** option in the Options dialog box to determine whether to capture requests and responses for your steps. If you do not want QuickTest to capture requests and responses for your steps, select **Tools > Options** and click the **Run** node. Then select **Never** from the **Save step screen capture to results** list.

Select a step or checkpoint in the run results tree to view the results for that step in the right-hand pane.



For checkpoints, the top right pane displays the checkpoint step results. The bottom right pane shows the details of the schema validation (if applicable) and a summary of the step or checkpoint results. If the schema validation failed, the reasons for the failure are also shown.

The bottom right pane may include the **View XML Checkpoint Results** button, enabling you to view details of the checkpoint's failure. The Web Services Add-in uses the **Save step screen capture to results** option in the Run pane of the Options dialog box to determine when to display this button.

For more information on the Run Results window, analyzing run results, and analyzing XML checkpoint results, see the *HP QuickTest Professional User Guide*.

Introduction to HP Service Test and HP Service Test Management

In addition to the functionality available in the QuickTest Web Services Add-in, as described in this chapter, HP provides full SOA testing capabilities with HP Service Test and HP Service Test Management.

HP Service Test is HP's tool for the construction and execution of functional tests for headless systems. You can create tests for Web Services, REST services, and other types of GUI-less applications.

With Service Test, you create tests by dragging and dropping Web Service activities from a toolbox into a canvas. Service Test also provides built-in steps for functional testing in standard areas such as string manipulations and file management.

Service Test also allows you to enable the tests for load testing. You can incorporate the test into **LoadRunner**, HP's tool for load testing, and check your service's behavior under load.

Service Test's integration with QuickTest allows Service Test tests to call QuickTest tests and vice versa.

HP Service Test Management is a tool for managing the testing process of application components and their changes in service-oriented architecture (SOA) and other GUI-less systems. Service Test Management integrates with HP Quality Center to provide a Web-based solution for testing the quality and performance of all types of application components throughout the entire application development life cycle.

HP Service Test Management adds an Application Components module in Quality Center that enables you to centrally manage your application component assets, such as Web Services. After you define or import the application components in Quality Center, you can generate a set of requirements and tests to validate their functionality, interoperability, security, boundaries, standards compliance, and performance in your environment.

For more information, contact your HP Quality Management supplier.

Troubleshooting and Limitations - Web Services

- ▶ If you are working with Microsoft .NET Framework 1.1 WSE 2.0 and you try to learn a WSDL that defines an RPC/literal service, QuickTest displays an error message because .NET Framework 1.1 WSE 2.0 does not support RPC/literal messages.

Workaround: Use .NET Framework 2.0 WSE 3.0 or Apache Axis 1.x.

- ▶ If you create a checkpoint or output value step for an operation that returns a multi-dimensional array (or perform an update run on such a step), the XML tree is generated for only one dimension of the array.
- ▶ When running existing tests, checkpoints checking complex values may fail if the following conditions are met:
 - ▶ The checkpoint was created using the QuickTest Professional Web Services Add-in 9.1.
 - ▶ The WSDL on which the WebService test object is based uses RPC-literal encoding.
 - ▶ The test object was learned using the .NET 2.0 WSE toolkit.

This is due to the fact that the old mechanism for creating checkpoints on this type of WSDL created an unnecessary element with an empty value. The new mechanism does not create this element for checkpoints and does not capture this element during the run session.

Workaround: Do one of the following:

- ▶ Perform an update run (**Automation> Update Run Mode**) of tests containing checkpoints matching the above description.
- ▶ Recreate the relevant checkpoints in QuickTest.
- ▶ Open the Checkpoint Properties dialog box and clear the check box next to the problematic elements.

41

Working with XML Data

This chapter describes the various ways in which you can use QuickTest Professional to manipulate XML data when testing Web services.

This chapter includes:

- About Working with XML Data on page 749
- Checking XML on page 750
- Outputting XML Values on page 752
- Working with XML Structures on page 753
- Parameterizing XML Values on page 763
- Working with XML Data Operations on page 765

About Working with XML Data

Web service operations often return XML data when the operation is performed.

When working with tests, you can create checkpoints and output values on the XML data that is returned from an operation performed on a Web service. You can perform **XMLData** methods on the returned XML data. You can also create XML structures and use them as templates for data in your test. This enables you to parameterize specific values in an XML document and use the parameterized XML hierarchy, for example, as the value for an XML operation argument or as the expected XML element value in a checkpoint.

When working with components, checkpoints and output values are not available. Instead, you can use the functions (keywords) defined in the **Web_Services.txt** function library to perform similar types of steps. To use these functions, the **Web_Services.txt** function library must be associated with the component's application area. If needed, an Automation Engineer can also wrap additional types of functionality in other function libraries and associate them with the component's application area. For more information on working with function libraries and application areas, see the *HP QuickTest Professional for Business Process Testing User Guide*.

QuickTest provides a set of XMLData methods that you can use to manipulate the XML values returned from the Web service operations in your test or component. For more information, see the *HP QuickTest Professional Object Model Reference*.

Checking XML

When working with tests, you can add XML checkpoints on the returned values from Web service operations in your test. An XML checkpoint is a verification point that compares actual values for specified XML elements, attributes and/or values with their expected values. If the results do not match, the checkpoint fails.

For example, suppose a flight reservation Web service has an operation that sets a flight itinerary and returns XML containing the specified itinerary details. A step in the test of the Web service could select a specific airline and return XML containing flight details. An XML checkpoint could compare the date and time information contained in the flight details XML data with the expected values for those XML nodes and check that the itinerary details match the expected result.

You insert a checkpoint using **Insert > Checkpoint > XML Checkpoint (From Resource)** or by using the **Insert Checkpoint or Output Value** button. You then select a test object and the operation whose return values you want to check.

Note: You can also insert checkpoints for every step automatically when using the Web Service Testing Wizard. For more information, see "Understanding the Web Service Testing Wizard" on page 706.

After you insert an XML checkpoint, QuickTest creates an expected XML hierarchy based on the return type defined in the WSDL. If the operation return type is not clearly defined, then QuickTest generates a generic XML hierarchy containing two nodes. In either case, QuickTest cannot generate the expected values of the nodes at the time that you insert the checkpoint. To check the relevant information, you must first populate the XML checkpoint hierarchy with the elements, attributes, and values expected in the returned XML. You can populate the XML hierarchy and values using the **Update Run Mode** option, by updating the hierarchy manually, or by importing a hierarchy from an existing file.

For more information on working with XML checkpoints and updating the XML hierarchy for XML checkpoints, see the *HP QuickTest Professional User Guide*.

Checking Business Components

When working with components, you can check that the returned values from Web service operations in your test match the expected values using a function library operation, such as **VerifyXMLValue**.

This and other verification operations are available only if the **Web_Services.txt** function library file is associated with your component's application area. For more information on using operations from a function library, see the *HP QuickTest Professional for Business Process Testing User Guide*.

Outputting XML Values

When working with tests, you can insert XML output value steps to retrieve data from the XML returned from Web service operations. An XML output value is a value captured from an XML return value during the run session and stored in a parameter for use at another point in the run session. When you create an output value step, the test retrieves a value at a specified point during the run session and stores it in the specified parameter. When you want to use the value later in the run session as input, you instruct QuickTest to retrieve it from the parameter.

For example, a step in a test of a flight reservation system could require the retrieval of an order number. This order number could be a parameterized output value from an XML structure, and could be stored in an action parameter in a test. You could then retrieve the order number value from the parameter for use as an operation argument in a later step.

You insert an output value step using **Insert > Output Value > XML Output Value (From Resource)** or by using the **Insert Checkpoint or Output Value** button.

QuickTest creates an expected XML hierarchy based on the return type defined in the WSDL. If the operation return type is not clearly defined, then QuickTest generates a generic XML hierarchy containing two nodes. To select the nodes you want to output, you must first ensure that the XML output value has the hierarchy you want. You can populate the XML hierarchy by updating the structure manually, importing it from an existing file, or updating it using the Update Run mode. For more information on updating the XML hierarchy for Web service XML output values, see the *HP QuickTest Professional User Guide*.

When you run your test, you can view summary results of the XML output value step in the Run Results window. You can also view detailed results by opening the XML Output Value Results window. For more information, see the *HP QuickTest Professional User Guide*.

Outputting XML Values for Business Components

When working with components, you can insert a step to retrieve a value from an XML object using the **OutputXMLValue** operation.

This operation is available only if the **Web_Services.txt** function library file is associated with your component's application area. For more information on using operations from a function library, see the *HP QuickTest Professional for Business Process Testing User Guide*.

Working with XML Structures

When you need to supply a complex value of type XML, you can either store the value in an XMLData object or you can use a QuickTest **XML structure**, which provides a visual, editable interface for the hierarchy of the XMLData object. XML structures are available for both tests and components.

XML structures also enable you to parameterize individual node values within the XML hierarchy so that those values can be retrieved from another parameter during the run session.

For example, you may have a Web service that manages an address book. The Web service has an **AddAddr** operation that adds a new entry to the address book. The operation receives an XML data argument containing the name, phone numbers, and address of the new entry.

Suppose you want to run a test that performs the **AddAddr** operation five times to add five different people to the address book, and then retrieves the updated address book. Instead of entering five **AddAddr** statements, each with a different XML string for the argument, you can create one XML structure and parameterize its name, phone number, and address values to take values from the data table (for tests) or from a local parameter (for components). You can then insert one **AddAddr** statement and set its argument to use the XML structure you created. Then, you can run multiple iterations of the action or component containing that step.

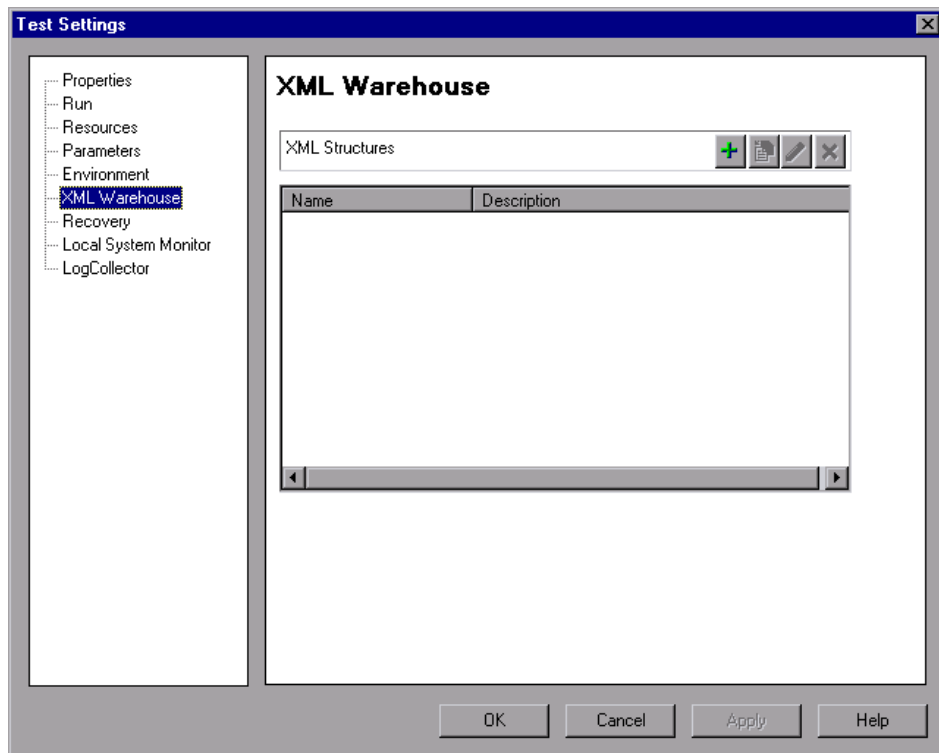
You can create several different XML structures. You can use an XML file or an existing XML structure as the base for a new XML structure, or you can edit nodes manually.

You save XML structures in the XML Warehouse for your test (**File > Settings > XML Warehouse**). XML structures are stored with the test or component—they are not external resources.





After you create an XML structure, you can parameterize XML data arguments using your XML structures. You can do this in the Keyword View or by manually inserting statements using the Expert View (tests only). For more information, see "Parameterizing XML Values" on page 763, or "Working with XML Data Operations" on page 765.

Managing XML Structures

The XML Warehouse pane displays information about the XML structures that you defined for your test or component. It also enables you to create new XML structures, modify or duplicate existing structures, and remove structures that you no longer require.



The XML Warehouse pane includes the following buttons:

Button	Description
	Enables you to create an XML structure. For more information, see "Creating XML Structures" on page 755.
	Enables you to duplicate an XML structure. For more information, see "Duplicating XML Structures" on page 757.
	Enables you to edit an XML structure. For more information, see "Modifying XML Structures" on page 757.
	Enables you to delete an XML structure. For more information, see "Deleting XML Structures" on page 758.

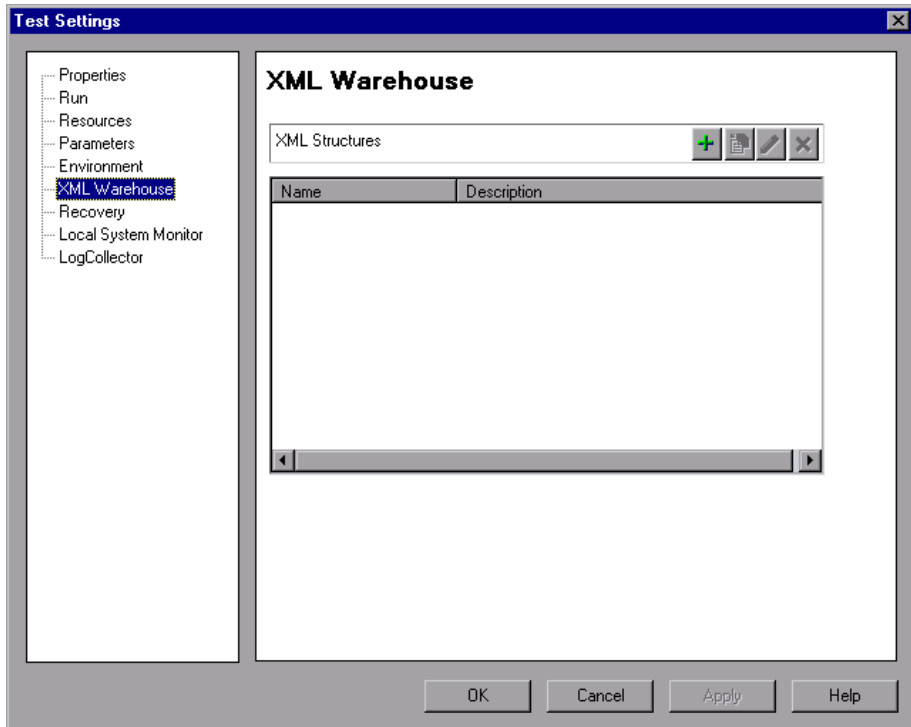
Creating XML Structures

You can create XML structures for use as an XML value in your test. You first set up the XML hierarchy for the XML structure by importing it and/or manually adding and editing its nodes. You can then edit or parameterize the attributes and values of the XML structure.

To create an XML structure:



- 1 Click the **Settings** toolbar button or select **File > Settings**. The Test Settings or Business Component Settings dialog box opens. Click the **XML Warehouse** node.



Note: The XML Warehouse pane of the Business Component Settings dialog box is identical to the example shown above.



- 2 Click the **Create new XML structure** button. The Create New XML Structure dialog box opens.
- 3 Enter a name and description for your XML structure.

- 4 Create the new XML structure, as described in "Understanding the Create New/Edit/Duplicate XML Structure Dialog Box" on page 758.
- 5 Click **OK** to create the XML structure and save it in the XML Warehouse for your test.

Duplicating XML Structures

If you already created an XML structure, you can create a new XML structure based on an existing one. This is useful if you need a new XML structure with the same hierarchy as an existing one, but you want to use different values or parameterization settings.

To duplicate an XML structure:

- 1 In the XML Warehouse pane of the Test Settings or Business Component Settings dialog box, select the XML structure you want to duplicate.
- 2 Click the **Duplicate XML Structure** button. The Duplicate XML Structure dialog box opens.
- 3 Specify a name for the new XML structure.
- 4 Modify the XML name, description, hierarchy elements and/or parameterization options, as described in "Understanding the Create New/Edit/Duplicate XML Structure Dialog Box" on page 758.
- 5 Click **OK**. The new XML structure is added to the list in the XML Warehouse pane.



Modifying XML Structures

If you already created an XML structure, you can modify its settings. For example, you can define or modify parameterization of values or attributes, or add or delete specific elements in the XML structure.

To modify an XML structure:

- 1 In the XML Warehouse pane of the Test Settings or Business Component Settings dialog box, select the XML structure you want to modify.
- 2 Click the **Edit XML Structure** button. The Edit XML Structure dialog box opens.



- 3 Modify the hierarchy elements and/or parameterization options, as described in "Understanding the Create New/Edit/Duplicate XML Structure Dialog Box" on page 758.
- 4 Click **OK** to save your changes and close the Edit XML Structure dialog box.

Deleting XML Structures

You can delete XML structures from the XML Warehouse if you no longer need them for your test or component.

To delete an XML structure:

- 1 In the XML Warehouse pane of the Test Settings dialog box, select the XML structure you want to delete.
- 2 Click the **Delete XML Structure** button. A confirmation message opens.
- 3 Click **Yes** to delete the selected XML structure from the XML Warehouse.

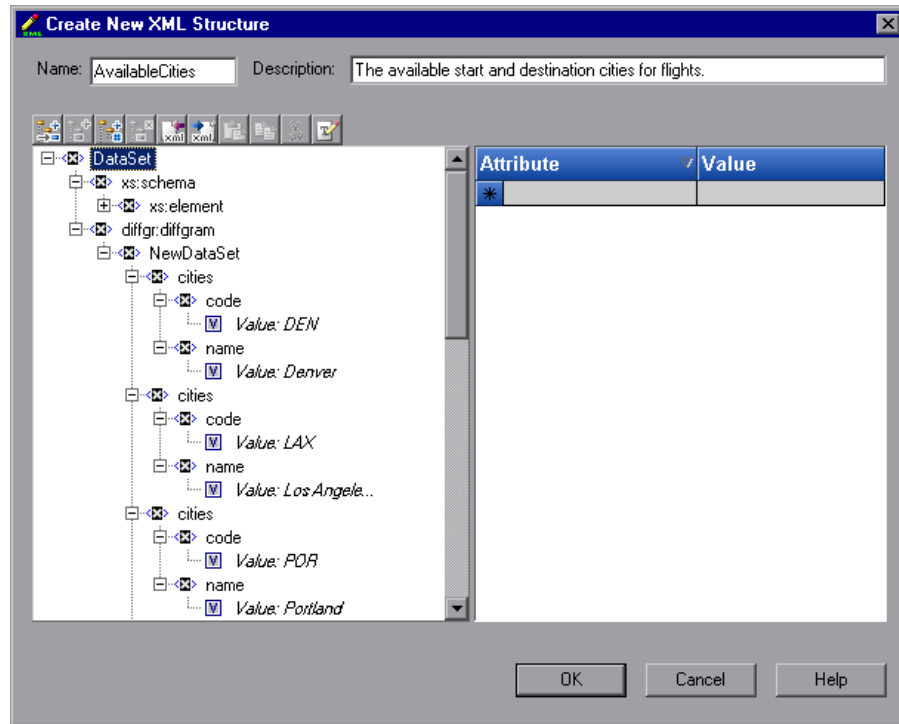


Understanding the Create New/Edit/Duplicate XML Structure Dialog Box

Note: Although this section refers specifically to the Create New XML Structure dialog box, the information provided also applies to the Edit XML Structure dialog box and the Duplicate XML Structure dialog box.

The Create New/Edit/Duplicate XML Structure dialog box enables you to create and modify an XML structure and save it in the XML Warehouse.

The dialog box displays the element hierarchy, attributes, and values (character data) of the XML hierarchy you create.













XML Structure Information

The top part of the Create New/Edit/Duplicate XML Structure dialog box displays information about the XML structure:

Item	Description
Name	The XML structure name. When you are editing an existing structure, this box is read-only.
Description	A textual description of the XML structure.




In addition, the following commands are available according to the node you select in the XML tree:

Command	Icon	Description
Add Child		Adds a child node below the selected node in the XML tree.
Insert Sibling		Adds a sibling node at the same level as the selected node in the XML tree.
Add Value		Enables you to assign a constant or parameterized value to the selected item.
Delete		Deletes the selected node.
Import XML		Enables you to browse to and select another XML file. The new file will override the selected node's current sub-tree.
Export XML		Enables you to save the current 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.

Command	Icon	Description
Edit XML as Text		<p>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.</p> <p>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 XML tree.</p> <p>For more information, see the <i>HP QuickTest Professional User Guide</i>.</p>
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>

Choosing the Element Values and/or Attributes to Edit or Parameterize

The Create New/Edit/Duplicate XML Structure dialog box contains an XML tree that displays the hierarchy of the XML structure, enabling you to select the element values and/or attributes that you want to edit or parameterize. You can edit or parameterize only the values; you cannot edit or parameterize element tags. This pane contains the following areas:

Area	Description
XML Tree	<p>The XML tree displays the hierarchal relationship between each element and value in the XML file. Each element is displayed with a  icon. Each value is displayed with a  icon.</p> <p>Note: Expanding a large XML tree might take some time.</p>
Data Area	<p>When an element is selected in the XML tree, the data area to the right of the tree displays the element's attributes and values.</p> <p>When an element value is selected in the XML tree, the data area displays the value node's data.</p> <p>When you click in a Value cell, the parameterization button  is displayed. Click the button to parameterize the value. For more information, see "Parameterizing XML Values" on page 763.</p>

Tips:


- ▶ The XML tree pane and the **Attribute** and **Value** columns in the right pane are resizable.
 - ▶ You can delete an attribute by selecting its row and pressing the DELETE key on your keyboard.
-


Parameterizing XML Values

When you add a step to your test, you can parameterize operation arguments of type XML so that they use dynamic data from an XML structure that you saved in the XML Warehouse. When you select to parameterize an XML argument, the Value Configuration Options dialog box opens. You can create a new XML structure from within this dialog box, and then use this XML structure as the parameterized value.

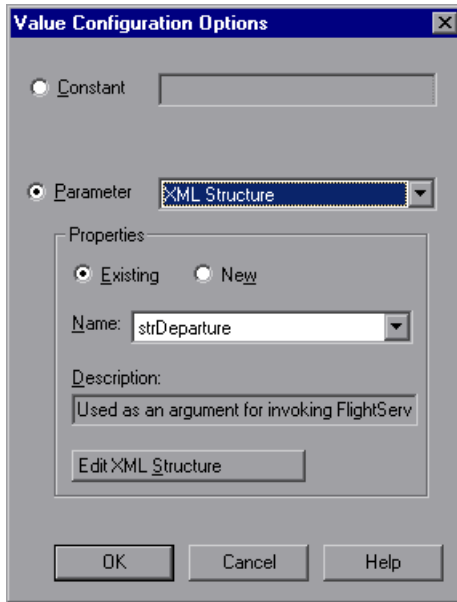
To parameterize an XML operation argument:

- 1 In the Keyword View, click in the **Value** column of the operation that you want to parameterize.

Note: The **Value** cell can comprise more than one partition, containing different argument values for the selected operation. Clicking a partition displays the parameterization button , as well as information for each argument in a tooltip. In the tooltip, the argument for the partition that is currently highlighted is displayed in bold.

- 2 Click the parameterization button . The Value Configuration Options dialog box opens, showing the currently defined value.
- 3 Select **Parameter**. If the value is already parameterized, the **Parameter** section displays the current parameter definition for the value. If the value is not yet parameterized, the **Parameter** section displays the default parameter definition for the value.

- 4 From the **Parameter** list, select **XML Structure**.



- 5 In the **Properties** area, select the type of XML structure you want to use:

- **Existing.** Enables you to specify an XML structure you have already created.

If you select **Existing**, select the name from the **Name** list. If you want to modify the existing structure, click the **Edit XML Structure** button and go to step 6. Otherwise, click **OK**. The test uses the values from the XML structure you specify. If you specify an existing XML structure, the description is displayed in read-only mode.

- **New.** Enables you to create a new XML structure.

If you select **New**, enter a name and textual description for it in the **Name** and **Description** boxes and click **Create New XML Structure**.

The Create New/Edit XML Structure dialog box opens.

- 6 Create or edit the XML structure as described in "Understanding the Create New/Edit/Duplicate XML Structure Dialog Box" on page 758.
- 7 When you finish, click **OK** to close the Create New/Edit XML Structure dialog box.
- 8 Click **OK** to close the Value Configuration Options dialog box.

Working with XML Data Operations

A test consists of statements (steps) coded in Microsoft VBScript. These statements are composed of objects, methods, and/or properties that instruct QuickTest to perform operations on, or retrieve information from, your Web service. You can insert and edit statements in your test manually in the Keyword View (tests and components) or in the Expert View (tests only).

You can add steps to perform XML data operations using the XMLWarehouse objects (tests and components) and XMLData objects (tests only).

Specifying XML Structures for XML Value Types

The XMLWarehouse object enables you to parameterize XML values such as method arguments of type XML. This enables you to use data from an XML structure that you saved in the XML Warehouse. The individual values of the elements and attributes in the XMLWarehouse object can be parameterized so that dynamic data can be used during a run session.

The following example uses the XMLWarehouse object in the Expert View to use the parameterized **Departure** XML structure as the argument for the **Find** method.

```
Set Find=
WebService("FlightNetWebService").Find(XMLWarehouse("Departure"))
```

When inserting steps manually in the Expert View, you can use IntelliSense to add an XML structure from the XML Warehouse. When you type `XMLWarehouse` followed by an open parenthesis (`(`, QuickTest displays a list of all XML structures in the XML Warehouse.

After inserting an XMLWarehouse object in a step in the Expert View, you can right-click on the object and select **View/Edit XML Structure** to open the Edit XML Structure dialog box.

Working with Retrieved XML Data

Many Web service operations return XML data objects. You can check and/or manipulate pieces of these objects using `XMLData` and other XML-related methods.

For example, you can use the `XMLData.ChildElementsByPath` method to return the child elements of a returned XML data object as an `XMLElementsColl` collection object. You can then use the `XMLElementsColl.Count` property to check whether the collection has the expected number of elements.

For information on the `XMLData` object, see the **Supplemental Objects** section of the *HP QuickTest Professional Object Model Reference*.

Part IX

Appendix

A

Supported Checkpoints and Output Values Per Add-in

The tables in this chapter show the categories of checkpoints and output values that are supported by QuickTest Professional for each add-in.

For more information about using checkpoints and output values in a specific add-in, see the relevant add-in section.

This chapter includes:

- Supported Checkpoints on page 770
- Supported Output Values on page 772

Supported Checkpoints

Table Legend

- ▶ S: Supported
- ▶ NS: Not Supported
- ▶ NA: Not Applicable

For additional information, see "Footnotes" on page 771.

	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
Web ²	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 773.

	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
Web ²	NA	NA	NA	S	S	S	S ³	NS	S	NA
Web Services	NA	NA	NA	NA	NA	NA	NA	NA	NA	S
WPF	NA	NA	NA	NA	S	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.

