



HP Unified Functional Testing

Software Version: 12.50

Windows® operating systems

Add-ins Guide

Document Release Date: July 2015
Software Release Date: June 2015

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 Notice

© Copyright 1992 - 2015 Hewlett-Packard Development Company, L.P.

Trademark Notices

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

Google™ and Google Maps™ are trademarks of Google Inc

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

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

Oracle and Java are registered trademarks of Oracle and/or its affiliates.

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:

<https://softwaresupport.hp.com>.

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

<https://softwaresupport.hp.com> and click **Register**.

Support

Visit the HP Software Support Online web site at: <https://softwaresupport.hp.com>

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: <https://softwaresupport.hp.com> and click **Register**.

To find more information about access levels, go to: <https://softwaresupport.hp.com/web/softwaresupport/access-levels>.

HP Software Solutions & Integrations and Best Practices

Visit **HP Software Solutions Now** at <https://h20230.www2.hp.com/sc/solutions/index.jsp> to explore how the products in the HP Software catalog work together, exchange information, and solve business needs.

Visit the **Cross Portfolio Best Practices Library** at <https://hpln.hp.com/group/best-practices-hpsw> to access a wide variety of best practice documents and materials.

Contents

Welcome to the Add-ins Guide	14
UFT Guides and References	15
Additional Online Resources	17
Part 1: Working with UFT Add-ins	19
UFT Add-ins Overview	20
UFT Add-in Support - Overview	21
Loading UFT Add-ins	22
Add-in Licenses	23
Considerations for Working with UFT Add-ins	23
Record and Run Settings for Add-ins - Overview	24
Considerations for Defining Record and Run Settings	25
Environment Variables in Record and Run Settings	26
UFT Add-in Extensibility	27
How to Manage UFT Add-ins	28
How to Define Record and Run Settings for UFT Add-ins	29
Add-in Manager Dialog Box	31
Chapter 2: Web-Based Application Support	34
Web-Based Application Support - Overview	35
Considerations - Web-Based Application Support	35
Registering Browser Controls	37
Accessing Password-Protected Resources in the Active Screen	37
Checkpoints for Web Pages	38
Event Recording Configuration for Web-Based Applications	39
Web Event Recording Configuration XML File Structure	40
Advanced Operations on Web-Based Applications	41
Activating methods associated with a Web-based object using the Object property	41
Using programmatic descriptions for the WebElement object	42
Web Object Identifiers	42
Web Object Identifier Types	43
CSS Web Object Identification	43
User-Defined XPath Web Object Identification	44
Automatic X-Path Web Object Identification	44
Attribute/* Notation Web Object Identification	44
Style/* Notation Web Object Identification	45
Considerations - Web Object Identifiers	48
How to Use Web Object Identifiers - Exercise	49

How to Modify Event Recording Configuration for Web-Based Applications	51
How to Configure UFT to Record Mouse Clicks	52
Active Screen Dialog Box	54
Environment Variables for a Web-Based Environment	55
Register Browser Control Utility	56
Web > General Pane (Options Dialog Box)	58
Browser Details Dialog Box	60
Web > Page/Frame Options Pane (Options Dialog Box)	61
Web > Advanced Pane (Options Dialog Box)	63
Web Pane (Test/Business Component Settings Dialog Box / Application Area - Additional Settings Pane)	68
Advanced Authentication Dialog Box	70
Web Tab (Record and Run Settings Dialog Box)	71
Troubleshooting and Limitations - Web-Based Application Support	77
Chapter 2: Windows-Based Application Support	81
Windows-Based Application Support - Overview	82
UFT Configuration for Windows-Based Applications	82
Record and Run Settings for Windows-Based Applications	83
Environment Variables for Windows-based Applications	84
Windows Applications > General Pane (Options Dialog Box > GUI Testing Tab)	85
Windows Applications > Advanced Pane (Options Dialog Box > GUI Testing Tab)	87
Considerations for Advanced Windows-based Application Testing	91
Windows Applications Tab (Record and Run Settings Dialog Box)	93
Application Details Dialog Box	96
Record and Run Setting Guidelines for Windows-Based Add-ins	98
Part 2: .NET Add-in	100
.NET Add-in - Overview	101
.NET Silverlight Add-in	102
.NET Silverlight Add-in - Quick Reference	103
Silverlight Add-in Extensibility	105
Troubleshooting and Limitations - Silverlight Add-in	106
.NET Web Forms Add-in	108
.NET Web Forms Add-in - Quick Reference	109
Considerations for Testing .NET Web Forms	110
.NET Windows Forms Objects - Checkpoints and Output Values	110
Accessing Internal Properties and Methods of Run-Time .NET Web Forms Objects	112
Troubleshooting and Limitations - .NET Web Forms	112
.NET Windows Forms Add-in	115
.NET Windows Forms Support - Quick Reference	116
.NET Windows Forms Support - Testing and Configuration	118

Considerations for Testing .NET Windows Forms Applications	119
.NET Windows Forms Objects - Checkpoints and Output Values	119
.NET Add-in Extensibility	120
.NET Windows Forms Spy	121
How to Use the .NET Windows Forms Spy	122
.NET Windows Forms Spy Dialog Box	126
Troubleshooting and Limitations - .NET Windows Forms	129
.NET Windows Presentation Foundation (WPF) Add-in	131
.NET Windows Presentation Foundation (WPF) Add-in - Quick Reference	132
.NET Windows Presentation Foundation (WPF) Add-in - Testing and Configuration	134
Considerations for Working with the WPF Add-in	135
WPF Objects, Methods, and Properties to Enhance Your Test or Component	135
About WPF User Interface Automation	136
Automation Elements	136
Control Patterns	136
WPF Add-in Extensibility	137
Troubleshooting and Limitations - .NET WPF	137
Part 3: ActiveX Add-in	139
ActiveX Add-in - Quick Reference	140
Considerations for Working with the ActiveX Add-in	141
Troubleshooting and Limitations - ActiveX Add-in	142
Part 4: Delphi Add-in	145
Delphi Add-in - Quick Reference	146
Delphi Add-in Extensibility	147
How to Enable Communications Between UFT and Your Delphi Application	148
Troubleshooting and Limitations - Delphi Add-in	149
Part 5: Flex Add-in	150
Flex Add-in - Quick Reference	151
Flex Add-in - Testing and Configuration	152
Enabling UFT to Identify Objects in your Flex Application	152
Considerations for Working with the Flex Add-in	155
How to Set Up the Adobe Flash Player Debugger to Enable UFT GUI Testing	156
How to Open Flex Applications Using the Runtime Loader	158
How to Embed a Flex Application in a Web Page with the Runtime Loader	160
How to Compile Flex Applications for UFT Testing	162
How to Work With Embedded Objects in Flex Lists, Tables, or Tree-Views	164
Flex Tab (Record and Run Settings Dialog Box)	166

Add/Edit Flex Application Dialog Box	168
Troubleshooting and Limitations - Flex Add-in	171
Part 6: Java Add-in	173
Java Add-in - Quick Reference	174
Java Add-in - Testing and Configuration	176
Java Add-in - Overview	177
Java Add-in Environments	177
Considerations - Java Add-in	178
Java Add-in Extensibility	179
How to Disable Dynamic Transformation Support (Advanced)	180
Java Pane (Options Dialog Box > GUI Testing Tab)	182
Advanced Java Options Dialog Box	184
Java Tab (Record and Run Settings Dialog Box)	187
Java Pane (Test/Business Component Settings Dialog Box / Application Area - Additional Settings Pane)	189
Java Environment Variables Settings	192
Record and Run Environment Variables for Java Objects	193
Troubleshooting and Limitations - Java Add-in	193
Running Another Java Application or Applet with the Same Settings	193
Identifying and Solving Common Problems	194
General Notes and Limitations	195
Java Add-in - Test Objects	201
Recording Steps on Java Objects	202
Recording Steps on Jtable Cell Editors	203
Text Checkpoint and Text Output Value Steps for Java Objects	203
Full Object Hierarchy Views	204
Advanced Java Test Object Methods	204
CreateObject Method	205
GetStatics Method	205
FireEvent / FireEventEx Methods	206
How to Modify Options for Recording on Java Tables	207
Part 7: Mobile Add-in	210
Mobile Add-in - Quick Reference	211
Mobile Tab (Record and Run Settings)	212
Mobile Pane (Options Dialog Box > GUI Testing Tab)	212
Open Application in HP Mobile Center	212
Part 8: Oracle Add-in	213

Oracle Add-in - Quick Reference	214
Oracle Add-in - Testing and Configuration	217
Considerations for Working with the Oracle Add-in	218
Recording Tests on Oracle Applications	219
Dynamic Transformation Support	220
How to Verify or Enable the Oracle Server Unique Name Attributes	221
How to Enable the Oracle Name Attribute	221
How to Set Oracle Environment Variables	222
How to Disable Dynamic Transformation Support	223
Oracle Tab (Record and Run Settings Dialog Box)	225
Oracle Record and Run Environment Variables	227
Troubleshooting and Limitations - Oracle Add-in	228
Part 9: PeopleSoft Add-in	230
PeopleSoft Add-in - Quick Reference	231
Considerations for Working with the PeopleSoft Add-in	232
Troubleshooting and Limitations - PeopleSoft Add-in	233
Part 10: PowerBuilder Add-in	234
PowerBuilder Add-in - Quick Reference	235
Considerations for Working with the PowerBuilder Add-in	236
Troubleshooting and Limitations - PowerBuilder Add-in	237
Part 11: Qt Add-in	238
Qt Add-in - Quick Reference	239
Considerations - Qt Add-in	241
Part 12: Add-in for SAP Solutions	242
Add-in for SAP Solutions - Overview	243
Web-based SAP Support	244
Web-Based SAP Support - Quick Reference	245
Considerations for Working with SAP GUI for HTML	247
Web > Page/Frame Options Pane (Options Dialog Box > GUI Testing Tab)	249
Web > Advanced Pane (Options Dialog Box > GUI Testing Tab)	251
Troubleshooting and Limitations - Web-based SAP	253
Windows-based SAP Support	257
Windows-based SAP Support - Quick Reference	259
Windows-based SAP Support - Testing and Configuration	261
SAP GUI Scripting API and UFT	262
How to Enable Support for SAP GUI for Windows	264

How to Enable Scripting on the SAP Application (Server-Side)	266
Considerations - Windows-based SAP Add-in for SAP Solutions	269
Environment Variables for Windows-based SAP Applications	270
Package and Patch Versions Requirements - SAP Application Server and SAP GUI for Windows	270
SAP Tab (Record and Run Settings Dialog Box)	271
SAP > General Pane (Options Dialog Box > GUI Testing Tab)	274
Troubleshooting and Limitations - Windows-based SAP	276
Windows-Based SAP Support - Test Objects	281
Checkpoints and Output Values in SAP GUI for Windows	282
Using the Auto-Parameterize Option to Parameterize Table and Grid Cell Values	282
How UFT Records in Auto-Parameterize Mode	283
Parameterized Cell Values in the Input Data Sheet	285
Considerations for Auto-Parameterization	286
Data in Rows that Require Scrolling	287
Low-Level or Analog Mode Recording on SAP GUI for Windows	288
Spooling Data from a Table	288
How to Record on Standard Windows Controls During an SAP GUI for Windows Recording Session	289
UFT-SAP Solution Manager Integration	290
UFT-SAP Solution Manager Integration - Overview	291
Test Management in SAP Solution Manager	291
Resource Files in Solution Manager	292
Solution Manager Testing Modes: Standalone or Integrated	292
Standalone Mode	293
Integrated Mode	294
SAP Structured Parameters	294
How to Configure Solution Manager to Work with UFT	295
How to Work with Tests in Solution Manager in Standalone Mode	297
How to Run a Test Stored in Solution Manager	301
How to Display or Edit a GUI Test from Solution Manager in Integrated Mode	303
How to Transfer Data To and From GUI Tests in Integrated Mode Using Test Parameters	304
How to Work with SAP Structured Parameters	305
Solution Manager Connection Dialog Box	309
Save GUI Test to Solution Manager Dialog Box	310
Open GUI Test from Solution Manager Dialog Box	311
Upload File to Solution Manager Dialog Box	312
Save External File to Solution Manager Dialog Box	313
Download File from Solution Manager	314
SAP > SAP Solution Manager Pane (Options Dialog Box > GUI Testing Tab)	315
Solution Manager Trace Options Dialog Box	316

Value Map Dialog Box	317
Part 13: Siebel Add-in	319
Siebel Add-in - Quick Reference	320
Siebel Add-in - Testing and Configuration	323
Siebel Add-in - Overview	324
Considerations - Siebel Add-in	324
Siebel Test Object Model - Overview	325
Recording Steps on Siebel Objects	326
Native Operations and Properties in Siebel 7.0.x and 7.5.x Applications	327
Siebel Add-in - Checkpoints and Output Values	327
Spooling Data from a Siebel Table	329
Siebel 7.7.x or Later - Test Automation Module Configuration	330
How to Define Environment Variables for Siebel Applications	331
Siebel Tab (Record and Run Settings Dialog Box)	333
Troubleshooting and Limitations - Siebel Add-in	335
Siebel 7.7.x or Later	336
Siebel 7.0.x and 7.5.x	337
Siebel Test Express	339
Using Siebel Test Express to Generate or Update Shared Object Repositories	340
How to Use Siebel Test Express to Generate or Update a Shared Object Repository	340
Create Object Repository Wizard	342
Connection Information Page (Create Object Repository Wizard)	342
Screen Selection Page (Create / Update Object Repository Wizard)	343
Importing Test Objects Page (Create Object Repository Wizard)	344
Object Repository Created / Updated Page (Create / Update Object Repository Wizard)	345
Part 14: Standard Windows Testing Support	346
Standard Windows Support -Quick Reference	347
Troubleshooting and Limitations - Standard Windows	349
Part 15: Stingray Add-in	350
Stingray Add-in - Quick Reference	351
Stingray Add-in - Testing and Configuration	353
Setting Up Stingray Object Support	354
Considerations for Working with the Stingray Add-in	354
Stingray Run-time Agent (Agent DLL)	354
Stingray Precompiled Agent Mode	355
How to Set Up Your Stingray Project Using the Precompiled Agent Mode	356
Stingray Support Configuration Wizard	358

Support Mode Selection Page (Stingray Support Configuration Wizard)	359
Add Support Code Page (Stingray Support Configuration Wizard)	360
Select Configuration Mode Page (Stingray Support Configuration Wizard)	362
Manual Configuration Mode Page (Stingray Support Configuration Wizard)	364
Automatic Configuration Mode Page (Stingray Support Configuration Wizard)	365
Detected Stingray Components Page (Stingray Support Configuration Wizard)	367
Finish Page (Stingray Support Configuration Wizard)	369
Detection Failure Page (Stingray Support Configuration Wizard)	370
Stingray Pane (Options Dialog Box > GUI Testing Tab)	371
Record Cell Editing Options - Example	374
Troubleshooting and Limitations - Stingray Add-in	374
Part 16: Terminal Emulator Add-in	377
Terminal Emulator Add-in - Quick Reference	378
Terminal Emulator Add-in - Testing and Configuration	380
Terminal Emulator Add-in - Overview	381
Recording Tests and Components on Terminal Emulator Applications	382
Considerations for Recording and Running Tests and Components on Terminal Emulators	383
Run Session Synchronization	384
Terminal Emulator Recovery Scenarios	384
How to Configure an Emulator to Work with the Terminal Emulator Add-in	385
How to Set Your HLLAPI Terminal Emulator to Work with UFT	389
How to Manage Terminal Emulator Configuration Settings	389
How to Copy Existing Terminal Emulator Configurations	390
How to Check the Validity of a Terminal Emulator Configuration	392
Validating a Terminal Emulator - Possible Error Responses	392
Invalid HLLAPI DLL	392
Cannot detect an open session	393
Cannot locate the main window class	393
Cannot detect the emulator screen	393
Cannot connect to the open session	393
Cannot retrieve session text	394
Cannot detect open session, or Cannot locate the main window class	394
HLLAPI DLL not found	394
More than one session open	394
Unknown error	394
How to Synchronize Steps on Terminal Emulators	395
Terminal Emulator Pane (Options Dialog Box > GUI Testing Tab)	397
Terminal Emulator Configuration Adjustments Dialog Box	399
Checkpoints and Output Values - Terminal Emulators	404

Terminal Emulator Configuration Wizard Overview	405
Terminal Emulator Configuration Wizard Welcome Page	406
Specify Wizard Operation Page	407
Emulator Setup Page	408
Configure HLLAPI Properties Page	409
HLLAPI Configuration Test Page	410
Configure Emulator Classes Page	413
Configure Emulator Screen Settings Page	414
Completing the Terminal Emulator Configuration Wizard Page	417
Troubleshooting and Limitations - Terminal Emulator	419
Part 17: VisualAge Smalltalk Add-in	424
VisualAge Smalltalk Add-in - Quick Reference	425
How to Configure the VisualAge Smalltalk Add-in	427
Part 18: Visual Basic Add-in	428
Visual Basic Add-in - Quick Reference	429
Troubleshooting and Limitations - Visual Basic Add-in	431
Part 19: Web Add-in	432
Web Add-in - Quick Reference	433
Web Add-in - Testing and Configuration	436
Event Recording Configuration for Web Objects - Overview	437
Event Listening and Recording for Web Objects	438
Considerations - Event Listening and Recording	439
Event Listening and Recording - Use-case Scenario	440
Web Add-in Extensibility	440
Extensibility Accelerator for HP Functional Testing	441
How to Manage Custom Web Event Recording Configurations	442
How to Manage Listening and Recording Events for Web Objects	443
Web Event Recording Configuration Dialog Box	445
Custom Web Event Recording Configuration Dialog Box	447
Troubleshooting and Limitations - Web Add-in	449
Web Add-in - Multiple Browser Support	451
Testing Applications on Multiple Browsers	452
Working with Multiple Browsers - Object Identification Issues	452
Testing Applications on Multiple Browsers - Creating a Single Test for All Browser Testing	457
Testing Applications on Multiple Browsers - Running the Test on Multiple Browsers	458
Using Descriptive Programming for Multiple Browser Testing - Use-case Scenario	459

Working With Mozilla Firefox	467
Working With Apple Safari on a Remote Mac Computer	469
The UFT Connection Agent for Mac Computers	471
Securing the Communication With the Remote Mac Computer	473
How to Set Up Multiple Browser Testing	474
How to Enable the HP Functional Testing Agent Chrome Extension	479
How to Enable UFT to Test Local HTML Pages in Google Chrome	480
How to Connect to a Remote Mac Computer	480
How to Install and Configure UFT Connection Agent on Your Mac	483
Troubleshooting and Limitations - Internet Explorer	487
Troubleshooting and Limitations - Mozilla Firefox	488
Troubleshooting and Limitations - Google Chrome and Apple Safari	491
Part 19: Web 2.0 Add-ins	495
Web 2.0 Add-ins - Quick Reference	496
Web 2.0 Toolkit Support	498
Troubleshooting and Limitations - Web 2.0 Add-ins	501
Part 19: Windows Runtime Add-in	504
Chapter 35: Windows Runtime Add-in - Quick Reference	505
Using the Windows Runtime Add-in in UFT - Overview	507
How to Use UFT in a Windows Runtime Environment	507
Troubleshooting and Limitations - Windows Runtime	509
Part 20: Appendix	511
Appendix A: GUI Checkpoints and Output Values Per Add-in	512
Supported Checkpoints	513
Supported Output Values	515
Send Us Feedback	518

Welcome to the Add-ins Guide

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

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

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

For users that work with UFT add-in extensibility, UFT also provides developer guides that describe how to extend UFT 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 Guide, available from the UFT Extensibility Documentation program group (**Start > All Programs > HP Software > HP Unified Functional Testing > Extensibility > Documentation** or the <UFT installation folder>\help\Extensibility folder.

Note: For details on accessing UFT and UFT tools and files in Windows 8 and Windows Server 2012, see ["Accessing UFT in Windows 8.X or Higher Operating Systems" on page 517](#).

Prerequisite Background

This guide is intended for UFT 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 UFT add-in takes advantage of commonly used UFT 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 UFT add-in.

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

This guide should be used in conjunction with the *HP Unified Functional Testing User Guide* and the *HP UFT Object Model Reference for GUI Testing*.

UFT Guides and References

The following tables provide a list of the UFT guides, online help and references:

Note: To check for recent updates of any of the guides below, visit the HP Software Product Manuals Web site (<https://softwaresupport.hp.com/group/softwaresupport/search-result?keyword=>).

Getting started

Reference	Description
What's New?	Describes the newest features in the latest version of Unified Functional Testing. You can also access the What's New from the Unified Functional Testing Help menu.
Product Movies	Click the link or select Help > Product Feature Movies to view short movies that demonstrate the main product features.
Readme	Provides last-minute news and information about Unified Functional Testing. For the latest readme file, go to the HP Software Manuals Web site (requires an HP Passport) at https://softwaresupport.hp.com/group/softwaresupport/search-result?keyword= .
UFT PAM	The Product Availability Matrix (PAM) provides current information about technologies and integrations supported for this version of UFT.
UFT Tutorial	The UFT Tutorial is a self-paced printable guide, designed to lead you through the process of creating GUI, API, and full UFT tests of your composite applications.
GUI Testing Tutorial for Web Applications	The GUI Testing Tutorial for Web Applications is a self-paced printable guide, designed to lead you through the process of creating GUI tests of your Web applications and familiarize you with the testing environment.

PDF guides

Guide	Description
UFT User Guide	The HP Unified Functional Testing User Guide describes how to use UFT to test your applications. It provides step-by-step instructions to help you create, debug, and run tests, and report defects detected during the testing process.
Run Results	The HP Run Results Viewer User Guide explains how to use the Run Results Viewer to interpret and use the test results from your GUI or API tests.

Guide	Description
Viewer	
UFT Installation Guide	The HP Unified Functional Testing Installation Guide provides complete, step-by-step instructions on how to install and set up UFT on a standalone computer.
UFT QuickStart	The UFT Installation QuickStart Sheet explains the steps to perform a basic installation of UFT.
UFT Add-ins Guide	The HP Unified Functional Testing Add-ins Guide explains how to set up support for UFT add-ins and standard Windows testing support. Add-ins enable you to test any supported environment using GUI tests and business components.
What's New	Describes the newest features in the latest version of Unified Functional Testing.
Security Reference	Contains security relevant information for using UFT and references for further information.
Runtime Engine Guide	Contains information on how to run tests using the Runtime Engine installation (without the UFT IDE).

References

Links to the references are available from the UFT online help home page.

Reference	Description
Object Model Reference	The Object Model Reference for GUI Testing includes a description, a list of methods and properties, syntax, examples, and identification properties for each UFT test object.
VBScript Reference	Microsoft's Visual Basic Scripting documentation that describes objects, methods, properties, functions, and other elements that can be used when writing VBScript scripts.
Automation Object Model Reference	List the objects, methods, and properties that enable you to control UFT from within another application.
Object Repository Automation Reference	Describes the objects that enable you to manipulate UFT shared object repositories and their contents from outside of UFT.
Run Results Schema Reference	Provides details about the structure of the Run Results XML schema, and describes the elements and attributes used in the its XML reports.
Test Object Schema Reference	A reference describing the elements and attributes available for creating test object configuration XML content, for use when creating UFT extensibility projects.
Object Repository Schema Reference	Describes the elements and complex types defined for the object repository schema.

For additional online references, see the Welcome section of a PDF guide.

Additional Online Resources

The following additional online resources are available:

Resource	Description
HP Software Support Online	<p>The HP Software Support Web site (www.hp.com/go/hpsupport). To access, choose Help > HP Software Support.</p> <p>Note: 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 user ID, go to: http://h20229.www2.hp.com/passport-registration.html</p>
Testing Forums	<ul style="list-style-type: none"> • GUI Testing: http://h30499.www3.hp.com/t5/Unified-Functional-Testing/bd-p/sws-Fun_TEST_SF • API Testing: http://h30499.www3.hp.com/t5/Service-Test-Support-and-News/bd-p/sws-Serv_TEST_SF • BPT: http://h30499.www3.hp.com/t5/Business-Process-Validation/bd-p/sws-BPT_SF
UFT Product Page	<p>The HP Unified Functional Testing product page (http://www8.hp.com/us/en/software-solutions/unified-functional-testing-automated-testing/index.html), with information and related links about UFT. To access, select Help > Useful Links > Product Page.</p>
Troubleshooting & Knowledge Base	<p>The Troubleshooting page (http://h20230.www2.hp.com/troubleshooting.jsp) on the HP Software Support Web site where you can search the HP Software Self-solve knowledge base. To access, select Help > Knowledge Base or Help > Troubleshooting.</p>
HP Software Community	<p>The HP IT Experts Community site (http://h10124.www1.hp.com/campaigns/IT_Experts/pages/home.html), where you can interact with other HP software users, read articles and blogs on HP software and access downloads of other software products.</p>
HP Manuals Site	<p>The HP Software Product Manuals Web site (http://support.openview.hp.com/selfsolve/manuals), to search for the most up-to-date documentation for a selected HP Software product. To access, select Help > Useful Links > HP Manuals Site.</p>
What's New	<p>The UFT What's New Help, describing the new features and enhancements in this version of UFT.</p>
Product Movies	<p>The UFT HPLN (HP Live Networks) page (https://hpln.hp.com/page/uft-120-videos) with a list of all product movies.</p>
HP Software Web site	<p>The HP Software Web site (www.hp.com/go/software). 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.</p>

You can access the following sample applications from the **Start** menu. These applications are the basis for many examples in this guide:

- Mercury Tours sample Web site. The URL for this Web site is <http://newtours.demoaut.com>.
- Flight reservation application. To access from the Start menu, select **All Programs > HP Software > HP Unified Functional Testing > Sample Applications > Flight API / Flight GUI**.

These applications are also available from the file system at: <UFT installation folder>\samples\Flights Application\FlightsGUI.exe (for the Flight GUI application) or <UFT installation folder>\samples\Flight_Application\FlightsAPI.exe (for the Flight API application)

Part 1: Working with UFT Add-ins

This section includes:

["UFT Add-ins Overview" on page 20](#)

["Web-Based Application Support" on page 34](#)

["Windows-Based Application Support" on page 81](#)

UFT Add-ins Overview

This chapter includes:

- UFT Add-in Support - Overview 21
 - Loading UFT Add-ins 22
 - Add-in Licenses 23
 - Considerations for Working with UFT Add-ins 23
- Record and Run Settings for Add-ins - Overview 24
 - Considerations for Defining Record and Run Settings 25
 - Environment Variables in Record and Run Settings 26
- UFT Add-in Extensibility 27
- How to Manage UFT Add-ins 28
- How to Define Record and Run Settings for UFT Add-ins 29
- Add-in Manager Dialog Box 31

UFT Add-in Support - Overview

UFT add-ins help you to create and run tests and business components on applications in a variety of development environments. After you load an add-in, you can record and run tests or business components on applications in the corresponding development environment, similar to the way you do with any other application. When you work with UFT add-ins, you can use special methods, properties, and various special options to create the best possible test or business component for your application.

You can install UFT add-ins when you install UFT, or you can install the add-ins at a later time by running the installation again in **Modify** mode. For details about installing and loading add-ins, see ["How to Manage UFT Add-ins" on page 28](#).

When UFT opens, you can choose which of the installed add-ins you want to load using the Unified Functional Testing ["Add-in Manager Dialog Box"](#) (described on page 31), but to maximize performance, you should load only the add-ins you need for that testing session.

UFT includes built-in support for testing standard Windows applications. Standard Windows testing support is automatically loaded when UFT opens.

Your UFT license enables all UFT features, including the use of all UFT add-ins. You can use the latest released version of all add-ins with UFT. If you upgrade from a version earlier than 9.5, only licensed add-ins are available. Additional non-licensed add-ins that you install are disabled in the ["Add-in Manager Dialog Box"](#). For details on installing add-ins and licenses, see the *HP Unified Functional Testing Installation Guide*. For details, see ["Add-in Licenses" on page 23](#).

Using Add-ins in Your Test or Component

- You can use the Keyword View, the Step Generator, and the Editor 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 enhance your tests and business components using environment-specific checkpoints and output values. See the sections describing checkpoints and output values in the *HP Unified Functional Testing User Guide*.
- You can customize the Active Screen capture settings for some of the UFT 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 details, see the section describing Active Screen capture setting options in the *HP Unified Functional Testing User Guide*.

Available Add-ins Environments

- Several UFT 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 ["Web-Based Application Support" on page 34](#)

- UFT 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 Smalltalk, 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 ["Windows-Based Application Support" on page 81](#)

This section also includes:

- [Loading UFT Add-ins](#) 22
- [Add-in Licenses](#) 23
- [Considerations for Working with UFT Add-ins](#) 23

Loading UFT Add-ins

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

When you start UFT, the ["Add-in Manager Dialog Box"](#) (described on page 31) opens. It displays a list of all installed add-ins.

If you have UFT add-ins installed, you can specify which add-ins to load at the beginning of each UFT session. You can also load UFT without add-in support if you want to test only standard Windows-based objects.

While UFT is open, you can check whether a specific add-in is installed by choosing **Help > About Unified Functional Testing**. Loaded add-ins are indicated by a check mark in the add-ins list.

Tip: The Web Services Add-in is supported for backwards compatibility only and is not enabled by default. New tests and components can use UFT's API testing solution for web service testing purposes. To enable the Web Services Add-in for previously created tests, contact HP Software support.

When you load an add-in, UFT 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 UFT, 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 UFT Object Model Reference for GUI Testing* (select **Help > HP Unified Functional Testing Help**).

Add-in Licenses

When you open UFT, 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" (described on page 31). 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 details on connecting to concurrent license servers, see the *HP Unified Functional Testing Installation Guide*.

You can also view license details for all currently loaded licensed add-ins in the About Unified Functional Testing Dialog Box (**Help > About Unified Functional Testing**).

- 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** UFT 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 UFT and all loaded add-ins—either seat or concurrent. For more information on license types, installing licenses, and modifying licenses, see the *HP Unified Functional Testing Installation Guide*.

Considerations for Working with UFT Add-ins

Consider the following when loading and using UFT add-ins:

Installing and Loading Add-ins

- You must install and load an add-in to enable UFT to recognize objects from the corresponding environment. To load an add-in, select the add-in from the "Add-in Manager Dialog Box" (described on page 31) that opens when you start UFT.
- 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 UFT add-ins require additional configuration after the installation is complete. Similarly, some environments may require configuration to enable UFT 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 UFT, while some must be opened after UFT is opened. These requirements are described in the introductory section for each relevant environment.
- When testing applications that do not contain .NET objects, it is strongly recommended that you do not load the .NET Add-in.
- When you open a test or business component, UFT compares the add-ins that are currently loaded with the add-ins associated with your test or with your business component's application area. If they do not match, UFT issues a warning message. For details on matching loaded add-ins with installed add-ins, see "[Match loaded add-ins with associated add-ins](#)" on page 28.

Running UFT from ALM with Add-ins

- When you run a UFT test from ALM, ALM instructs UFT to load the add-ins that are associated with the test.

If you created the test in ALM (and not in UFT), 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 UFT. For details, see the section on template tests in the *HP Unified Functional Testing User Guide*.
- Before you run a UFT test from ALM, make sure that the required UFT add-ins are installed on the computer on which you want to run the UFT test.

Record and Run Settings for Add-ins – Overview

Before you record or run a test on an application, you can use the Record and Run Settings Dialog Box to instruct UFT 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 UFT to recognize during record, run, and Object Spy sessions. For example, you can choose to have UFT 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.

The Record and Run Settings dialog box opens automatically each time you begin recording a new test and saves your settings with that test. Subsequently, when you perform additional record or run sessions on existing tests, the Record and Run Settings dialog box does not open. This is because UFT automatically applies the saved record and run settings.

You can modify the record or run settings prior to any session using the Record and Run Settings dialog box (**Record > Record and Run Settings**).

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 details on which tab of the Record and Run Settings dialog box you should use with an add-in, see the relevant add-in chapter.

For details on defining record and run settings, see the relevant add-in environment section, for example, "[Windows Applications Tab \(Record and Run Settings Dialog Box\)](#)" on page 93 if you are testing a standard Windows-based application.

This section also includes:

- [Considerations for Defining Record and Run Settings](#)25
- [Environment Variables in Record and Run Settings](#)26

Considerations for Defining Record and Run Settings

- The setting of the Active Screen capture level (**Tools > Options > GUI Testing tab > 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 details, see the section on setting active screen options in the *HP Unified Functional Testing User Guide*.
- You can set the record and run settings for some add-in environments using the corresponding tab in the Record and Run Settings dialog box (displayed only when the add-in is installed and loaded). For details on record and run settings for:
 - Windows-based applications, see "[Record and Run Settings for Windows-Based Applications](#)" on page 83.
 - Web-based environments, see "[Web Tab \(Record and Run Settings Dialog Box\)](#)" on page 71.
 - other environments, see the relevant add-in chapter.

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

The following settings are recommended:

- **Windows Applications tab.** When not running Windows-based applications, select **Record and run only on:** and confirm that all three 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, they prevent you from inadvertently recording operations performed on Windows applications (such as e-mail) during your recording session. These settings also prevent UFT from opening unnecessary applications when you record or run tests on Windows-based applications.

- 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 UFT to ensure that UFT recognizes the application. For details, see the relevant add-in chapter.
- 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 **Record > Record and Run Settings**.

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.

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 define environment variables to specify the record and run details, those values override the values in the Record and Run Settings dialog box. For details, see "[Environment Variables in Record and Run Settings](#)" below.
- After defining the connection information for HP Mobile Center in the Mobile pane of the Options dialog box (**Tools > Options > GUI Testing** tab > **Mobile** node), a remote access window opens every time a record or run session begins. To prevent this window from opening when you are not testing mobile applications, select **Do not record and run tests on mobile** in the **Mobile** pane of the Record and Run settings dialog box.

Environment Variables in Record and Run Settings

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), UFT 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, UFT 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.)

For details on setting and modifying environment variables, see ["How to Define Record and Run Settings for UFT Add-ins" on page 29](#).

UFT Add-in Extensibility

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

When UFT 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 UFT 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 UFT 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 UFT 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 UFT is able to recognize. This enables you to create tests that fully support the specific behavior of your controls.

UFT 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 Testing Extensibility for UFT program. Testing Extensibility is intended for customers who want to extend UFT testing capabilities for technologies or applications not supported by existing UFT add-ins. Participation in the program requires a separate license agreement with HP.

For details on Testing Extensibility, contact HP Software support.

For details on UFT Add-in Extensibility, see:

- ["Delphi Add-in Extensibility" on page 147](#)
- ["Java Add-in Extensibility" on page 179](#)
- [".NET Add-in Extensibility" on page 120](#)
- ["Silverlight Add-in Extensibility" on page 105](#)
- ["Web Add-in Extensibility" on page 440](#)
- ["WPF Add-in Extensibility" on page 137](#)

How to Manage UFT Add-ins

This task contains the following steps:

- "Load or remove add-ins from UFT" below
- "Match loaded add-ins with associated add-ins" below

Load or remove add-ins from UFT

1. Start UFT.

The Unified Functional Testing "Add-in Manager Dialog Box" (described on page 31) opens.

2. In the add-in list, select or clear the check box for the relevant add-in and click **OK**.

Match loaded add-ins with associated add-ins

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

- Close and reopen UFT, and select the required add-ins in the "Add-in Manager Dialog Box" (described on page 31).
- Remove the add-ins from the list of associated add-ins for your test or business component. To change the list of add-ins associated with your test or business component, select **File > Settings** and click **Modify** in the Properties pane.

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

- Close and reopen UFT, 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 business component's application area.
 - To change the list of add-ins associated with your test, select **File > Settings** and click **Modify** in the Properties pane.
 - To change the list of add-ins associated with your business component, open the application area associated with your business component, and modify the list in the Properties pane.

For details on associating add-ins with your test or business component, see the *HP Unified Functional Testing User Guide*.

How to Define Record and Run Settings for UFT Add-ins

This task contains the following steps:

- ["Define record and run settings for specific add-ins" below](#)
- ["Set record and run environment variables for add-ins" below](#)

Define record and run settings for specific add-ins

1. Review ["Considerations for Defining Record and Run Settings" on page 25](#).
2. Use one of the following to open the Record and Run Settings dialog box:
 - Select **Record > Record and Run Settings**.
 - Click the **Record** button or select **Record > 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.

The Record and Run Settings dialog box is divided by environment into several tabbed pages.

3. Select the relevant environment by clicking a tab.
4. Set the required options, as described in the relevant add-in chapter.
5. To apply your changes and keep the Record and Run Settings dialog box open, click **Apply**.
6. Close the Record and Run Settings dialog box to begin your record or run session, click **OK**.

Set record and run environment variables for add-ins

1. Review ["Environment Variables in Record and Run Settings" on page 26](#).
2. Use one of the following to open the Record and Run Settings dialog box:
 - Select **Record > Record and Run Settings**.
 - Click the **Record** button or select **Record > 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.

The Record and Run Settings dialog box is divided by environment into several tabbed pages.

3. Set your record and run preferences normally before recording 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), UFT ignores the record settings you enter in the dialog box.

4. Record and edit your test normally.

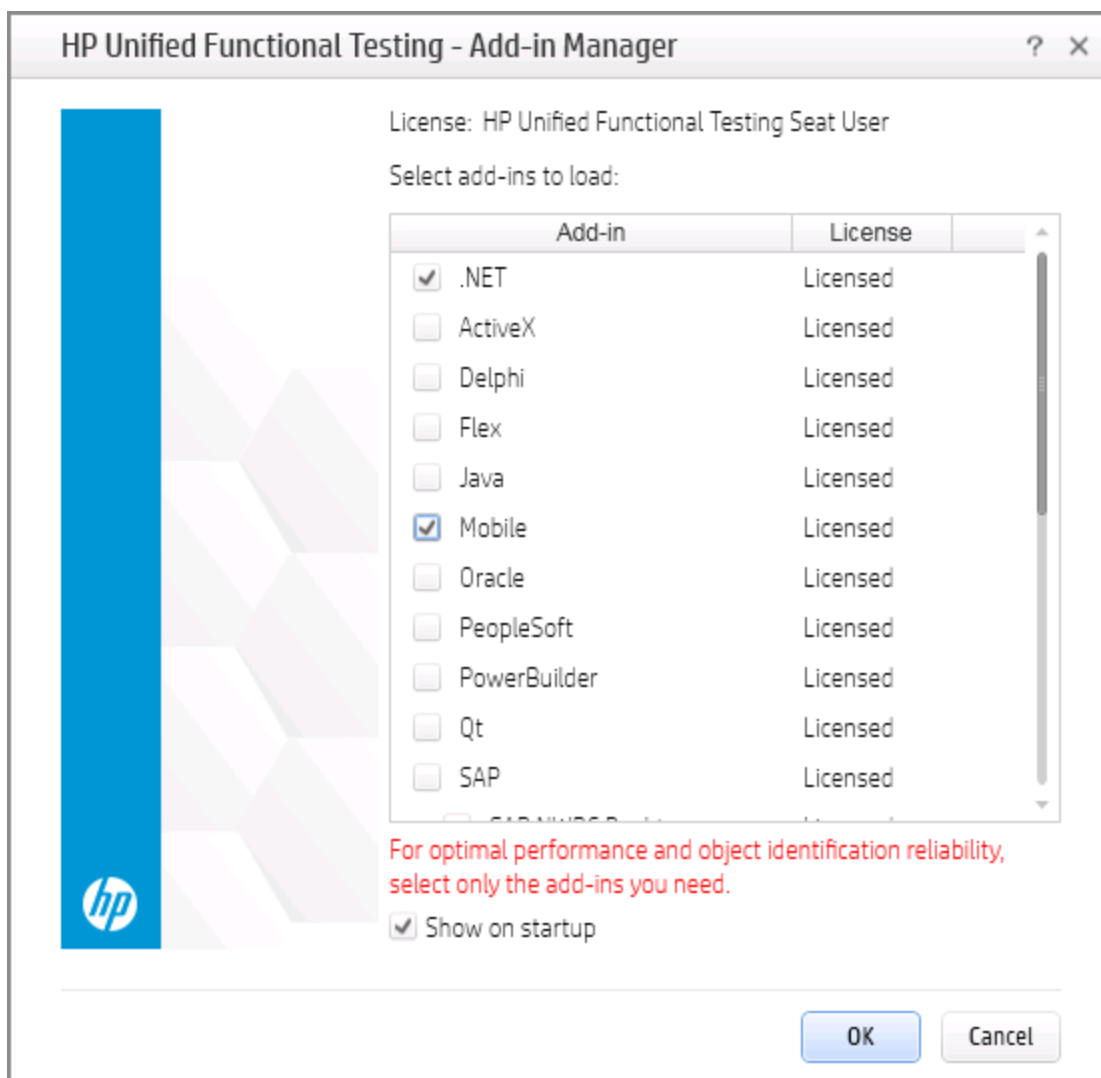
5. If you did not define environment variables prior to recording your test, define an environment variable for each application detail you want to set using the appropriate variable name. For details on the variable names required, see:
 - For Web browsers and URLs to open, see ["Environment Variables for a Web-Based Environment" on page 55.](#)
 - For Windows applications on which you want to record and run tests, see ["Record and Run Settings for Windows-Based Applications" on page 83.](#)
 - For other tabs in the Record and Run Settings dialog box, see the relevant add-in chapter in this guide.

For details 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 Unified Functional Testing User Guide*.

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

Add-in Manager Dialog Box

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



<p>To access</p>	<p>By default, this dialog box opens when you start UFT.</p> <p>To display the Add-in Manager if it does not open when you start UFT, select Tools > Options > General tab > Startup Options node and select Display Add-in Manager on startup.</p>
<p>Important information</p>	<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. • UFT remembers which add-ins you selected so that the next time you open UFT, the same add-ins are selected in the Add-in Manager dialog box.

	<ul style="list-style-type: none"> 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.
Relevant tasks	<ul style="list-style-type: none"> How to Start UFT in the <i>HP Unified Functional Testing User Guide</i> "How to Manage UFT Add-ins" on page 28
See also	<ul style="list-style-type: none"> About Unified Functional Testing Detailed Information Window in the <i>HP Unified Functional Testing User Guide</i> "UFT Add-in Support - Overview" on page 21 "Considerations for Working with UFT Add-ins" on page 23

User interface elements are described below:

UI Element	Description
License	The currently active license used by UFT.
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 details, see the relevant Add-in Extensibility Developer Guide, available from the UFT Extensibility Documentation program group (Start > All Programs > HP Software > HP Unified Functional Testing > Extensibility > Documentation or the <UFT installation folder>\help\Extensibility folder).</p> <div style="background-color: #f0f0f0; padding: 10px;"> <p>Note:</p> <ul style="list-style-type: none"> 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. For details on accessing UFT and UFT tools and files in Windows 8 and Windows Server 2012, see "Accessing UFT in Windows 8.X or Higher Operating Systems" on page 517. </div>
License column (for add-ins only)	<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 UFT. Add-ins use the same license as UFT. Therefore, if UFT uses a Permanent license, the add-ins use the same Permanent license; if UFT 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 UFT seat license—not a concurrent license.) <p>For more details, see the <i>HP Unified Functional Testing Installation Guide</i>.</p>
Show on startup	<p>Instructs UFT to display the Add-in Manager dialog box each time you open UFT. When this check box is cleared, UFT opens and loads the same add-ins it loaded in the previous session, without displaying the Add-in Manager.</p> <div style="background-color: #f0f0f0; padding: 10px;"> <p>Note: (for concurrent license users) If this check box was cleared in the previous session, and the</p> </div>

UI Element	Description
	<p data-bbox="407 296 1364 359">license type selected from the concurrent license server in that session is not currently available, UFT tries to load an available license that matches the selected add-ins.</p> <p data-bbox="407 394 781 422">To display the Add-in Manager again:</p> <p data-bbox="407 443 1364 499">Select Tools > Options > General tab > Startup Options node and select Display Add-in Manager on startup.</p>

Chapter 2: Web-Based Application Support

This chapter includes:

- Web-Based Application Support - Overview 35
 - Considerations - Web-Based Application Support 35
- Registering Browser Controls 37
- Accessing Password-Protected Resources in the Active Screen 37
- Checkpoints for Web Pages 38
- Event Recording Configuration for Web-Based Applications 39
 - Web Event Recording Configuration XML File Structure 40
- Advanced Operations on Web-Based Applications 41
 - Activating methods associated with a Web-based object using the Object property 41
 - Using programmatic descriptions for the WebElement object 42
- Web Object Identifiers 42
 - Web Object Identifier Types 43
 - Considerations - Web Object Identifiers 48
 - How to Use Web Object Identifiers - Exercise 49
- How to Modify Event Recording Configuration for Web-Based Applications 51
- How to Configure UFT to Record Mouse Clicks 52
- Active Screen Dialog Box 54
- Environment Variables for a Web-Based Environment 55
- Register Browser Control Utility 56
- Web > General Pane (Options Dialog Box) 58
 - Browser Details Dialog Box 60
 - Web > Page/Frame Options Pane (Options Dialog Box) 61
 - Web > Advanced Pane (Options Dialog Box) 63
- Web Pane (Test/Business Component Settings Dialog Box / Application Area - Additional Settings Pane) 68
 - Advanced Authentication Dialog Box 70
- Web Tab (Record and Run Settings Dialog Box) 71
- Troubleshooting and Limitations - Web-Based Application Support 77

Web-Based Application Support - Overview

UFT provides a number of add-ins for testing Web-based applications. The way you configure many of your UFT settings is the same or similar for most UFT 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 following sections:

- [".NET Web Forms Add-in - Quick Reference" on page 109](#)
- ["PeopleSoft Add-in - Quick Reference" on page 231](#)
- ["Siebel Add-in - Quick Reference" on page 320](#)
- [".NET Silverlight Add-in - Quick Reference" on page 103](#)
- ["Web Add-in - Quick Reference" on page 433](#)
- ["Web 2.0 Add-ins" on page 495](#)
- ["Web-Based SAP Support - Quick Reference" on page 245](#)

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 UFT Web-based add-ins. For details, see ["Extensibility Accelerator for HP Functional Testing" on page 441](#).

Considerations - Web-Based Application Support

This section contains the following items to consider when testing Web-based applications:

- ["Recording and Running Steps on Web Controls" below](#)
- ["Working with Web Browsers" on the next page](#)
- ["Testing Applications with Embedded Web Browser Controls" on the next page](#)

Recording and Running Steps on Web Controls

- If UFT 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 details, see ["Event Recording Configuration for Web-Based Applications" on page 39](#).
- 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 UFT records a step only if the value in the list changes.
- 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 UFT

can perform operations on those objects even though a human user of the application could not.

Working with Web Browsers

- You select your browser in the "[Web Tab \(Record and Run Settings Dialog Box\)](#)" (described on page 71).
- UFT does not support the option to zoom in and out of a Web page. If you use this option, some UFT 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.
- UFT 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.
- For UFT to run JavaScript methods, the security settings in your browser must be set to allow active scripting. (In Internet Explorer, for example, you can find these security settings under: **Tools > Internet Options > Security > Custom Level > Scripting > Active scripting**.)

This is relevant if your test steps include **RunScript** or **EmbedScript** methods, or if you are working with test objects supported using Web Add-in Extensibility, such as Web 2.0 test objects.

- 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.

Testing 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 UFT is open.
- (For tests) In the "[Web Tab \(Record and Run Settings Dialog Box\)](#)" (described on page 71), the **Record and run test on any open browser** option is selected. (This option is not relevant for business components.)

After these conditions are met, you can start adding steps or running your test or business component.

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.

UFT cannot automatically recognize the objects that provide browser functionality in your non-Web application as Web objects. For UFT 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.

For user interface details, see "[Register Browser Control Utility](#)" on page 56.

Accessing Password-Protected Resources in the Active Screen

When UFT 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 keyword components (although it is available for scripted 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 (see "[Active Screen Dialog Box](#)" on page 54), 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 \(Test/Business Component Settings Dialog Box / Application Area - Additional Settings Pane\)](#)" (described on page 68). UFT 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.

If you do not enter this information in the Web pane of the Test Settings dialog box and attempt to access the password-protected resources, the "[Active Screen Dialog Box](#)" (described on page 54) opens.

- **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](#)" (described on page 70). This gives the Active Screen access to password-protected resources in your Active Screen pages for the duration of your UFT 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 UFT session.

In most cases, the automatic login is sufficient. In some cases, you must use the manual login method. In rare cases, you may need to use both login mechanisms to enable access to all resources in your Active Screen pages.

Note: If your Web site is not password-protected, but you are still unable to view images or other resources on your Active Screen, you may not be connected to the Internet, the Web server may be down, or the source path that was captured with the Active Screen page may no longer be accurate.

Checkpoints for Web Pages

This section describes the checkpoint types that are supported only for Web-based add-ins. For a list of all supported checkpoints per add-in, see "[Supported Checkpoints](#)" on page 513

Accessibility Checkpoints

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 keyword components.

For details, see the section on accessibility checkpoints in the *HP Unified Functional Testing User Guide*.

Page Checkpoints

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 keyword components.

For details, see the section on page checkpoints in the *HP Unified Functional Testing User Guide*.

Tip: You can instruct UFT 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 of the Options dialog box (**Tools > Options > GUI Testing** tab > **Web > Advanced** node). For details, see "[Web > Advanced Pane \(Options Dialog Box\)](#)" on page 63.

Event Recording Configuration for Web-Based Applications

When you record on a Web application, UFT 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.

UFT includes event recording configurations that have been optimized for each Web-based add-in, so that in most cases UFT 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, UFT 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), UFT 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 Event Recording Configurations for Web-Based Applications

You can view and customize the configuration settings for the Web Add-in in the "[Web Event Recording Configuration Dialog Box](#)" on page 445. The settings in that dialog box affect the recording behavior only for objects that UFT recognizes as Web test objects.

Note: For the purposes of Web event recording, UFT 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.

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.

Web Event Recording Configuration XML File Structure

The Web event recording configuration XML file is structured in a specific format when you export it from the "Custom Web Event Recording Configuration Dialog Box" (described on page 447). 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.

For task details, see "How to Modify Event Recording Configuration for Web-Based Applications" on page 51.

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>
```

The following attributes enable you to define the listening criteria and recording status options in the XML file:

Attribute	Possible Values
Listen	1. Always 2. If Handler 4. If Behavior

Attribute	Possible Values
	6. If Handler or Behavior 0. Never
Record	1. Disabled 2. Enabled 6. Enabled on Next Event

Advanced Operations on Web-Based Applications

This section describes various advanced operations you can perform on Web-based objects, and includes the following:

- [Activating methods associated with a Web-based object using the Object property](#)41
- [Using programmatic descriptions for the WebElement object](#)42

Activating methods associated with a Web-based object using the Object property

In the Editor, 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 details on the **Object** property (.Object), see the section on retrieving and setting identification property values in the *HP Unified Functional Testing 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 UFT recognizes an object as a Web-based object that does not fit into any other UFT 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, UFT 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 details on the **WebElement** object, see the *HP UFT Object Model Reference for GUI Testing*. For details on programmatic descriptions, see the section on programmatic descriptions in the *HP Unified Functional Testing User Guide*.

Web Object Identifiers

During a run session, UFT 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 details on the standard methods UFT uses to identify objects, see the section on how UFT identifies objects in the *HP Unified Functional Testing User Guide*.

You can instruct UFT to use Web object identifiers before the regular object identification process to help limit the number of candidate objects to identify. UFT accesses the application's DOM and returns

objects that match the object identifier property values. UFT 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.

To follow an exercise describing the identification process using Web object identifiers, see "[How to Use Web Object Identifiers - Exercise](#)" on page 49.

For details about the general workflow of the object identification process, see the section on object identification in the *HP Unified Functional Testing User Guide*.

This section includes:

- [Web Object Identifier Types](#) 43
 - [CSS Web Object Identification](#) 43
 - [User-Defined XPath Web Object Identification](#) 44
 - [Automatic X-Path Web Object Identification](#) 44
 - [Attribute/* Notation Web Object Identification](#) 44
 - [Style/* Notation Web Object Identification](#) 45
- [Considerations - Web Object Identifiers](#) 48
- [How to Use Web Object Identifiers - Exercise](#) 49

Web Object Identifier Types

The following Web object identifiers are available:

- [CSS Web Object Identification](#) 43
- [User-Defined XPath Web Object Identification](#) 44
- [Automatic X-Path Web Object Identification](#) 44
- [Attribute/* Notation Web Object Identification](#) 44
- [Style/* Notation Web Object Identification](#) 45

For general considerations on working with Web object identifiers, see "[Considerations - Web Object Identifiers](#)" on page 48.

CSS Web Object Identification

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.

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

For usage examples, see "[How to Use Web Object Identifiers - Exercise](#)" on page 49.

For more details on the Object Identification dialog box and on programmatic descriptions, see the sections on the Object Identification dialog box and Programmatic Descriptions in the *HP Unified Functional Testing User Guide*.

User-Defined XPath Web Object Identification

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.

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

For usage examples, see "[How to Use Web Object Identifiers - Exercise](#)" on page 49.

Automatic X-Path Web Object Identification

You can instruct UFT 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 UFT-generated property, and therefore it is not available from the Object Spy dialog box, the Add/Remove Properties dialog box, or the Object Identification dialog box.

You enable this option in the **Web > Advanced** pane of the Options dialog box. For details, see "[Web > Advanced Pane \(Options Dialog Box\)](#)" on page 63.

Attribute/* Notation Web Object Identification

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.

The following examples describe how you can use the `attribute/*` notation to identify Web objects:

- "[Example of using attribute/<property> to identify a Web object](#)" on the next page
- "[Example of using attribute/<event> to identify a Web object](#)" on the next page

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"
```

Style/* Notation Web Object Identification

You can use the style/* notation to access the values of CSS properties for a Web-based object. You can then use these property values to identify such objects by adding the notation to the object's description properties using programmatic descriptions.

The following examples describe how you can use the style/* notation to identify Web objects:

- ["Example of using style/<property> to identify a Web object using the background-color property" on the next page](#)
- ["Example of using style/<property> to identify a Web object using the background-position property" on the next page](#)
- ["Example of using style/<property> to identify a Web object and update invalid input controls" on page 47](#)

For more details, see ["Considerations for using the Style/* notation" on page 48](#)

Example of using style/<property> to identify a Web object using the background-color property

Suppose a web page has different colored button objects on the same page:

```
<input type="button" style="background-color:rgb(255, 255, 0)">  
<input type="button" style="background-color:rgb(255, 0, 0)">
```

You can identify the button that you want to click by adding the style/background-color notation to the object's description properties or using a programmatic description to identify the object:

```
Browser("Simple controls").Page("Simple controls").WebButton("style/background-  
color:=rgb\ (255, 255, 0)").Click
```

Example of using style/<property> to identify a Web object using the background-position property

Suppose a web page has different image objects on the same page:

```
<head>  
<style>  
img.home {  
width:46px;  
height:44px;  
background:url(img_navsprites.gif) 0px 1px;  
}  
img.next {  
width:43px;  
height:44px;  
background:url(img_navsprites.gif) -91px 1px;  
}  
</style>  
</head>  
<body>  
  
<br /><br />  
  
<br /><br /><br /><br /><br />  
<input type="text" id="info" />  
</body>
```

You can identify the image that you want to click by adding the `style/background-position` notation to the object's description properties and using a programmatic description to identify the object:

```
Browser("Browser").Page("Page").WebElement("html tag:=img", "style/background-
position:=-91px 1px").Click
```

Example of using `style/<property>` to identify a Web object and update invalid input controls

Suppose a web page has different edit objects on the same page, each defined to use a red background color if the input is invalid:

```
<head>
<STYLE TYPE="text/css">
input:required:invalid, input:focus:invalid {
background-color: rgb(255, 0, 0);
}
input:required:valid {
background-color: rgb(0, 255, 0);
}
</STYLE>
</head>
<body>
<form>
<p>email:</p>
<input id="emailTxt" type="email" placeholder="test@.com" required><br />
<p>url:</p>
<input id="urlTxt" type="url" pattern="https?://.+" placeholder="https?://.+"
required><br />
</form>
```

You can identify the invalid values that need to be corrected by adding the `style/background-color` notation to the object's description properties and using a programmatic description to identify the object. You can then enter valid input into the edit object based on the object's `type` property:

```
Set oDesc = Description.Create()
oDesc("micclass").Value = "WebEdit"
oDesc("style/background-color").Value = "rgb\ (255, 0, 0\)"

Set invalidEdit = Browser("Browser").Page("Page").ChildObjects(oDesc)
numberOfEdits = invalidEdit.Count
For i = 0 To numberOfEdits - 1
If invalidEdit(i).GetROProperty("type") = "email" Then
invalidEdit(i).Set "test@.com"
End If
```

```
invalidEdit(i).GetROProperty("type") = "url" Then  
invalidEdit(i).Set "http://www.test."  
End If
```

Considerations for using the Style/* notation

- The CSS property values are returned using the browser's functions, and values may differ depending on the browser you are using.
- UFT retrieves the CSS property values from the browser. When designing tests or components that will run on different browsers, keep in mind that different browsers may have different CSS functionality and return different property values for the same object.
- CSS shorthand properties, such as **animation**, **font**, **background**, and **outline** are not supported. Instead, use a concrete CSS property in your descriptions, such as **background-image**, **font-family**, **border-width**, and so on.

Considerations - 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, UFT 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	UFT continues to identify the matching objects.	UFT ignores the learned XPath and continues with the regular object identification process.
No objects match the XPath value	Object identification fails, and UFT continues to identify the object using Smart Identification	UFT 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:

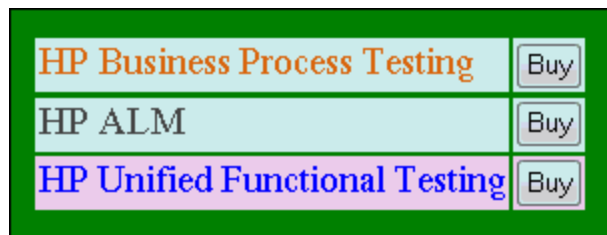
1. ["Prerequisites" below](#)
2. ["Create a sample Web application" below](#)
3. ["Learn the button objects in the Web application" below](#)
4. ["Remove the ordinal identifiers from the button objects" on the next page](#)
5. ["Add a CSS identifier based on the object's parent container" on the next page](#)
6. ["Add an XPath identifier based on the object's parent container" on the next page](#)
7. ["Add an XPath identifier based on the object's sibling element" on page 51](#)
8. ["Results" on page 51](#)

1. Prerequisites

- a. Open UFT 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 in the **Web > Advanced** node (**Tools > Options > GUI Testing tab > Web > Advanced** node) by clearing the **Learn and run using automatic XPath identifiers** check box.

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 UFT, open the Object Repository Manager, and select **Object > Navigate and Learn**. UFT is hidden, and the cursor changes to a pointing hand.

- b. To verify that UFT learned the objects correctly, in the object repository, select each **Button** object and select **View > Highlight in Application**. UFT 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_ALM**.
 - Rename **Buy_3** to **Buy_UFT**.

4. Remove the ordinal identifiers from the button objects

Because all of the Button objects have identical property values, when UFT 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 UFT to identify the objects incorrectly if the sorting order of the buttons in the application changes.

- a. In the Object Repository Window, 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 the previous steps 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**. UFT 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_UFT** 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='UFT']/*/INPUT
```

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

- a. Select the **Buy_ALM** 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(text(),'ALM')]/../*/INPUT
```

8. Results

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

How to Modify Event Recording Configuration for Web-Based Applications

This task includes the following steps:

- ["Modify the event recording configuration XML file manually" below](#)
- ["Modify the event recording configuration in the Web Event Recording Configuration dialog box" on the next page](#)

Modify the event recording configuration XML file manually

- 1. In a text or XML editor, open the appropriate **MyEnvEventConfiguration.xml** file from the **<UFT 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.

Modify the event recording configuration in the Web Event Recording Configuration dialog box

1. Back up the event recording configuration for the Web environment:
 - a. Select **Record > 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 **<UFT 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 **<UFT 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 "[Event Recording Configuration for Web Objects - Overview](#)" on page 437.
 - c. Select **File > Save Configuration As** and overwrite the previous **<UFT 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 XML file that you saved earlier.

Caution: UFT always applies the configuration that is loaded in the "[Web Event Recording Configuration Dialog Box](#)" on page 445 to all Web objects. If you do not restore the Web configuration file, then UFT will apply the configuration for the **<MyEnv>EventConfiguration.xml** file you last loaded, and as a result, UFT may not record Web events properly.

How to Configure UFT to Record Mouse Clicks

This task describes how to instruct UFT to record right mouse clicks by modifying the configuration file manually.

1. **(Web Add-in only) Prerequisite - Export the configuration file from the Custom Web Event Recording Configuration dialog box**
 - a. Select **Record > Web Event Recording Configuration**, and then click **Custom Settings**. For user

interface details, see the section on the [Custom Web Event Recording Configuration dialog box](#).

- b. Export your custom configuration to an `.xml` file by selecting **File > Save Configuration As**. Then, 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`.

2. Open the XML file in a text editor

Open the configuration file for editing in any text editor. The configuration file uses a predefined XML structure. For details, see "[Web Event Recording Configuration Dialog Box](#)" on page 445.

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

3. Modify the XML file to enable mouse click recording

- a. Edit the `.xml` 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:

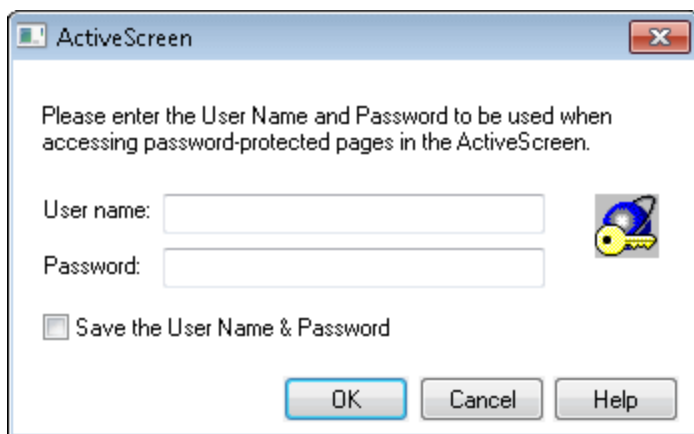
```
<Event Name="onmousedown" Listen="2" Record="1">
  <Property Name="button" Value="2" Listen="2" Record="2"/>
  <Property Name="button" Value="1" Listen="2" Record="2"/>
</Event>
```

Note: Only one event, either **onmouseup** or **onmousedown**, should be used to handle mouse clicks. If both events are used, UFT records two clicks instead of one. By default, UFT listens for the **onmouseup** event.

- b. Save the .xml file.
- 4. **(Web Add-in only) Load the XML file into the Custom Web Even Recording Configuration dialog box**
 - a. In the "Custom Web Event Recording Configuration Dialog Box" (described on page 447), select **File > Load Configuration**. The Open dialog box opens.
 - b. 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.
 - c. 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.

Active Screen Dialog Box

This dialog box enables you to enter login information for any images or other resources in the Active Screen that are password-protected using the standard authentication mechanism.



To access	This dialog box opens when you select a step in your test or results in which one or more images or other resources in the Active Screen may be password-protected.
Important information	<ul style="list-style-type: none">• After you enter a user name and password, refresh the Active Screen by selecting a new step in the Keyword View or select View > Active Screen to re-display 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 details, see "Advanced Authentication Dialog Box" on page 70.
See also	<ul style="list-style-type: none">• "Advanced Authentication Dialog Box" on page 70• "Web Pane (Test/Business Component Settings Dialog Box / Application Area - Additional Settings Pane)" on page 68

User interface elements are described below:

UI Element	Description
User name	The user name used to log into your application.
Password	The password used to log into your application.
Save the User Name and Password	<p>Enables the automatic Active Screen login mechanism, by adding the login information to the Active Screen access area in the Web pane of the Test Settings dialog box. This prevents the Active Screen dialog box from opening, and ensures 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.</p> <p>Note: You can also add the login information manually in the Web pane of the Test Settings dialog box. For details, see "Web Pane (Test/Business Component Settings Dialog Box / Application Area - Additional Settings Pane)" on page 68.</p>

Environment Variables for a Web-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 details on environment variables and how to use them in tests, see ["Environment Variables in Record and Run Settings" on page 26.](#)

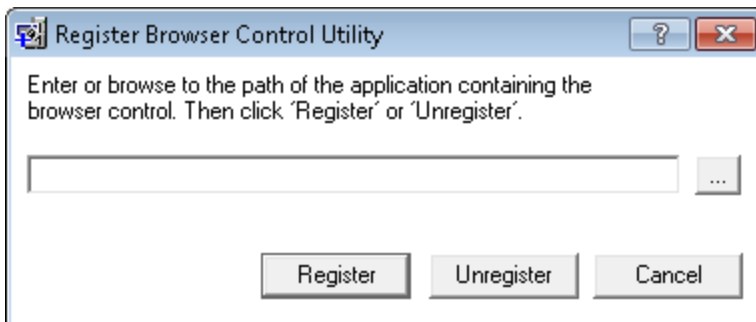
To use environment variables to define the Web browser and URL to open, set the appropriate variable names as specified below:

Option	Variable Name	Description
Type	BROWSER_ENV	<p>The browser program to open. For example, Microsoft Internet Explorer, Google Chrome, or Mozilla Firefox.</p> <p>Possible values:</p> <p>IE. Opens Internet Explorer.</p> <p>IE64. Opens a 64-bit version of Internet Explorer.</p> <p>CHROME. Opens Google Chrome.</p> <p>FIREFOX. Opens the latest version of Mozilla Firefox that is both installed on the computer and supported by UFT.</p> <p>FIREFOX64. Opens the latest version of 64-bit Mozilla Firefox that is both installed on the computer and supported by UFT.</p> <p>FF<VersionNumber>. Opens the specified version of Mozilla Firefox. For example: FF36 (version 3.6), FF40 (version 4.0), FF140 (version 14.0).</p> <p>SAFARI. Opens Safari on the remote Mac computer connected to UFT (defined in the Web tab of the</p>

Option	Variable Name	Description
		Record and Run Settings dialog box or in the REMOTE_HOST environment variable). Note: <ul style="list-style-type: none"> • If the specified browser program is not installed, the default browser is used. • Mozilla Firefox is supported only for the Web Add-in. • When specifying a Firefox version number, make sure to specify a supported version. For a list of all supported browser versions, see the <i>HP Unified Functional Testing Product Availability Matrix</i>, available from the HP Support Matrix page.
Address	URL_ENV	The Web address to display in the browser.
Remote Host	REMOTE_HOST	The host name or IP address of the Mac computer to which UFT connects. By default, UFT connects to the Mac using port 8822 . To use a different port, append the port number to the host name: <i><hostname>:<port number></i> . Make sure to configure the same port number on the Mac, in the UFT port option in the UFT Connection Agent preferences. For more details, see " How to Connect to a Remote Mac Computer " on page 480. Note: Relevant only for running tests and components on the Apple Safari browser.
use SSL flag	USE_SSL	Specifies whether to secure the connection to the Mac computer by using SSL for the connection. Possible values: <ul style="list-style-type: none"> • TRUE • FALSE (Default) Note: Relevant only for running tests and components on the Apple Safari browser.

Register Browser Control Utility

This utility enables you to define the path of your Web application hosting the browser control. After registration, UFT will recognize Web objects in your application when recording or running tests.



To access	Select one of the following:
------------------	------------------------------

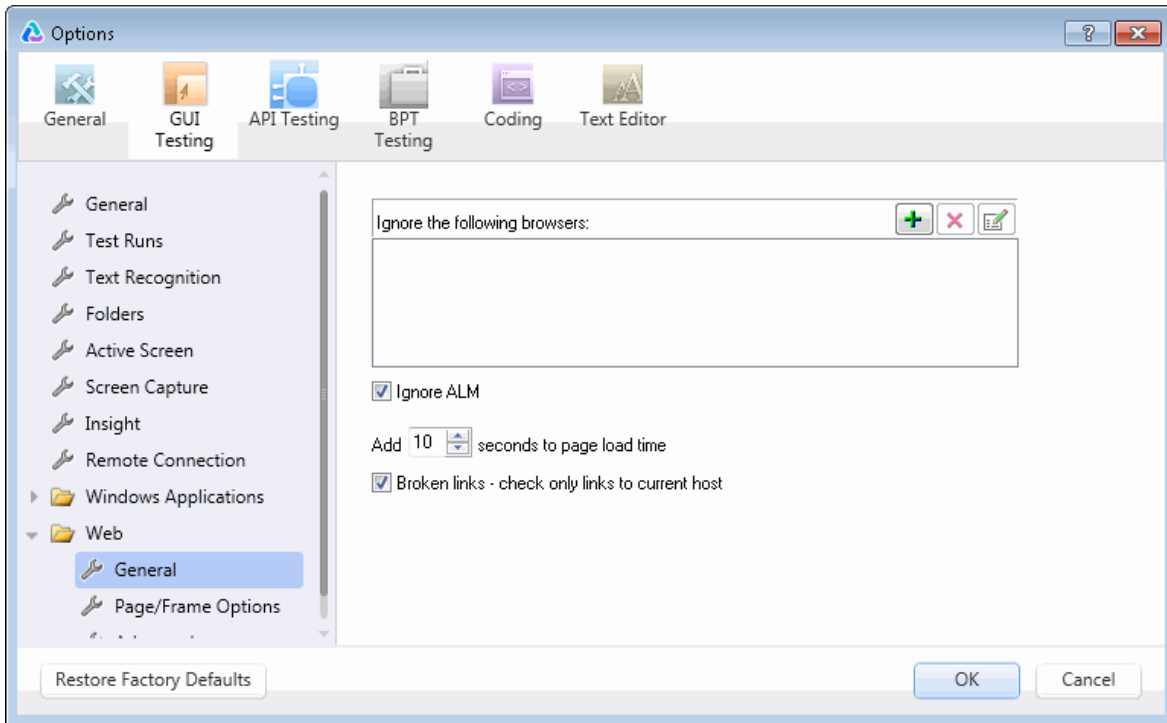
	<ul style="list-style-type: none"> • Start > All Programs > HP Software > HP Unified Functional Testing > Tools > Register New Browser Control • <UFT installation folder>\bin\SettingNewBrowserControlApplication.exe <p>Note: For details on accessing UFT and UFT tools and files in Windows 8 and Windows Server 2012, see "Accessing UFT in Windows 8.X or Higher Operating Systems" on page 517.</p>
Important information	<ul style="list-style-type: none"> • After you register an application hosting a browser control using this utility, you must restart UFT before you test your application. • This tool must be run with UAC disabled. For details about disabling UAC, see the <i>HP Unified Functional Testing User Guide</i>.
See also	"Registering Browser Controls" on page 37

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

UI Element	Description
<host application path>	The absolute path to the .exe file of the application hosting the browser control.
Register / Unregister	Registers or removes a registered application.




Web > General Pane (Options Dialog Box)

This pane enables you to determine how UFT behaves when recording and running tests or business components on Web sites.



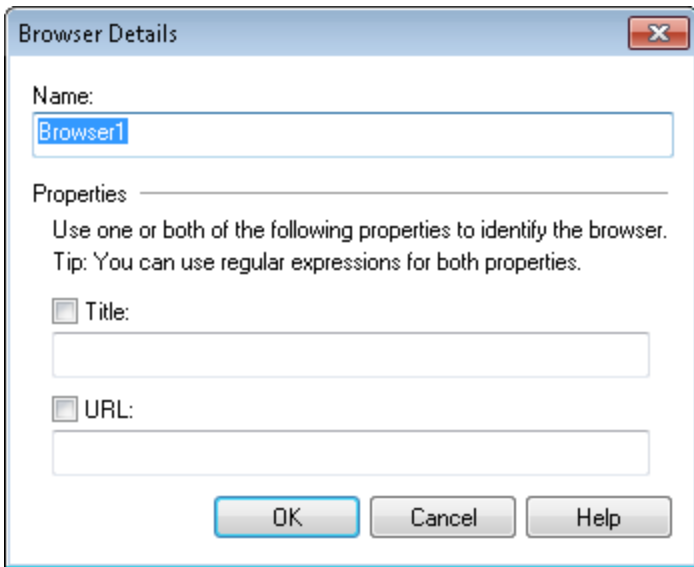
To access	<ol style="list-style-type: none"> Do one of the following: <ul style="list-style-type: none"> Ensure that a GUI test, action, or component is in focus in the document pane. In the Solution Explorer, select a GUI test or component node or one of its child nodes. Select Tools > Options > GUI Testing tab > Web > General node.
Important information	<ul style="list-style-type: none"> You can also modify how UFT displays captured Web pages in the Active Screen. You do this in the Active Screen pane of the Options dialog box (Tools > Options > GUI Testing tab > Active Screen node). For details, see the section describing Active Screen options in the <i>HP Unified Functional Testing User Guide</i>. The Restore Factory Defaults button resets all Options dialog box options to their defaults.
Relevant tasks	<p>"How to Modify Event Recording Configuration for Web-Based Applications" on page 51</p>
See also	<ul style="list-style-type: none"> "Browser Details Dialog Box" on page 60 "Web > Page/Frame Options Pane (Options Dialog Box)" on page 61 "Web > Advanced Pane (Options Dialog Box)" on page 63



User interface elements are described below:

UI Element	Description
<p>Ignore the following browsers</p>	<p>Instructs UFT to ignore any specified browsers that may be open while UFT is recording or running a test or business 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.</p> <p>For example, you may want to check your company's share price or the news headlines during a record and run session. If you instruct UFT to ignore these specific browsers, they do not affect the session.</p> <div style="background-color: #f0f0f0; padding: 10px; margin: 10px 0;"> <p>Note:</p> <ul style="list-style-type: none"> • UFT 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 business components and new steps in existing tests or business components only, but not to any other existing steps. • When working with tests, UFT ignores 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 details, see "Web Tab (Record and Run Settings Dialog Box)" on page 71. </div> <p>For details, see "Browser Details Dialog Box" on the next page.</p>
	<p>Add Browser. Opens the "Browser Details Dialog Box" (described on page 60, which enables you to add browsers to the Ignore the following browsers list.</p>
	<p>Modify Browser Details. If you selected a browser from the Ignore the following browsers list, this option opens the "Browser Details Dialog Box" (described on page 60, which enables you to edit the details of the selected browser.</p>
	<p>Remove Browser. Removes the selected browser from the list if you no longer want UFT to ignore it during a record or run session.</p> <div style="background-color: #f0f0f0; padding: 10px; margin: 10px 0;"> <p>Note: 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.</p> </div>
<p>Ignore ALM</p>	<p>Instructs UFT to ignore all instances of ALM that are opened while recording or running a test or business component. By default, this option is selected.</p>
<p>Add __ seconds to page load time</p>	<p>Instructs UFT to add a specified number of seconds to the page load time property specified in each Page checkpoint. (Page checkpoints are not relevant for keyword components.)</p> <div style="background-color: #f0f0f0; padding: 10px; margin: 10px 0;"> <p>Note: This option is a safeguard that prevents page checkpoints from failing in the event that the amount of time it takes for a page to load during the run is longer than the amount of time it took during the record session.</p> </div>
<p>Broken links - check only links to current host</p>	<p>Instructs UFT to check only for broken links that are targeted to your current host.</p>

Browser Details Dialog Box

This dialog box enables you to specify the browsers that you want UFT to ignore during a record or run session.



To access	In the " Web > General Pane (Options Dialog Box) ", click the Add Browser button  or the Modify Browser Details button  . For details on the Web pane, see " Web > General Pane (Options Dialog Box) " on page 58.
Important information	<ul style="list-style-type: none"> You can specify any one of the Title or URL properties, or both. However, the Title and URL properties have an AND relationship, meaning that a browser must match both property values (if defined) to be ignored by UFT. You can use regular expressions when specifying the values of these properties. For example, you can use <code>*finance.mybank.com</code> to specify all finance.mybank.com domains and Web sites starting with <code>www.</code>, <code>http://</code>, or <code>https://</code>. You do not need to use a regular expression to include child pages of a site, as UFT automatically ignores the entire domain or site. For details on supported regular expressions, see the <i>HP Unified Functional Testing User Guide</i>.
See also	<ul style="list-style-type: none"> "Web > General Pane (Options Dialog Box)" on page 58 "Web > Page/Frame Options Pane (Options Dialog Box)" on the next page "Web > Advanced Pane (Options Dialog Box)" on page 63

User interface elements are described below:

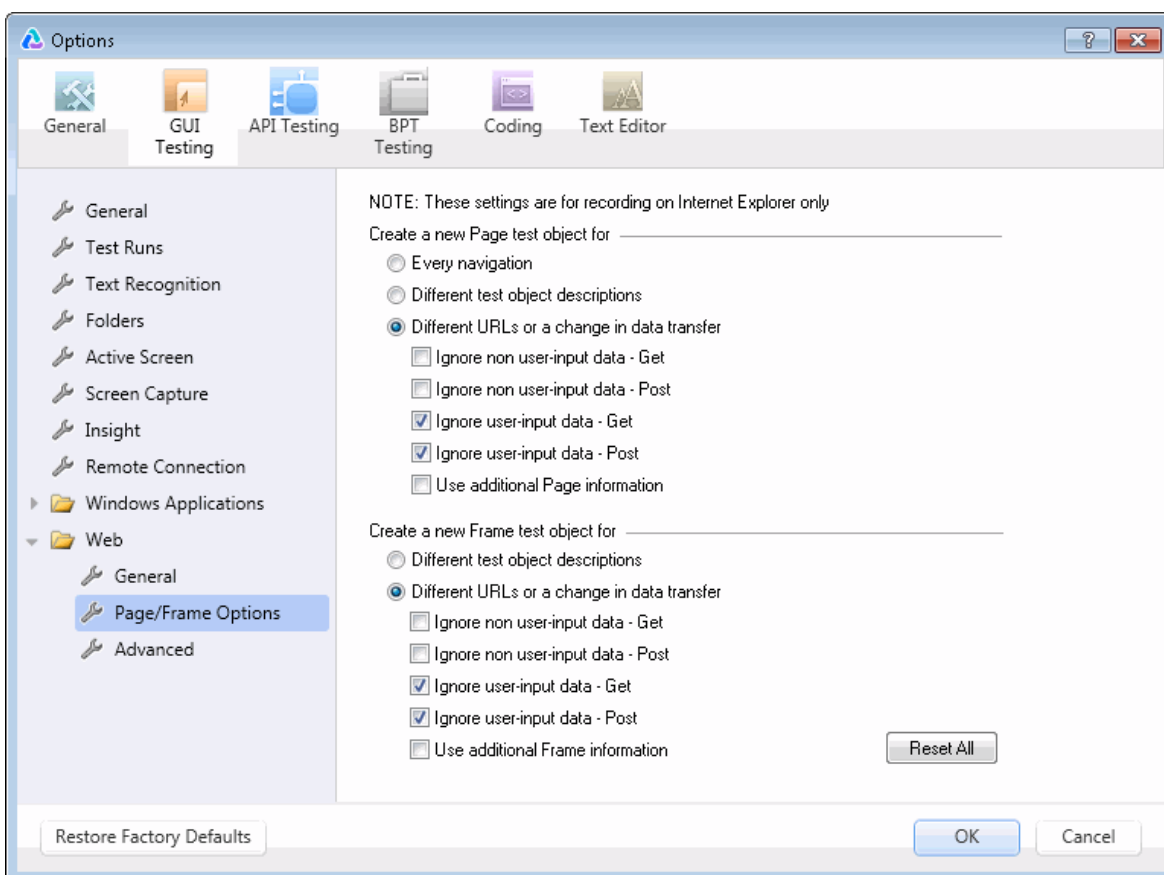
UI Element	Description
Name	The name of the browser definition. The name you specify is used only to identify the browser in the list, and is not used by UFT.

UI Element	Description
	Default format: Browser<number of browser in list>
Title	The name of the Web page as it appears in the title bar of the browser. Example: MyBank - Finance.*
URL	The URL of the Web page. Any descendants of the specified Web page are automatically included in the list of browsers to ignore. Example: http://www.finance.mybank.com

Web > Page/Frame Options Pane (Options Dialog Box)

This pane enables you to modify how UFT records Page and Frame objects.

Note: The settings in this pane are relevant for recording in Internet Explorer only.



To access	<ol style="list-style-type: none"> 1. Make sure that a GUI test is open. 2. Select Tools > Options > GUI Testing tab > Web > Page/Frame Options node.
See also	<ul style="list-style-type: none"> • "Web > General Pane (Options Dialog Box)" on page 58

- ["Web > Advanced Pane \(Options Dialog Box\)" on the next page](#)

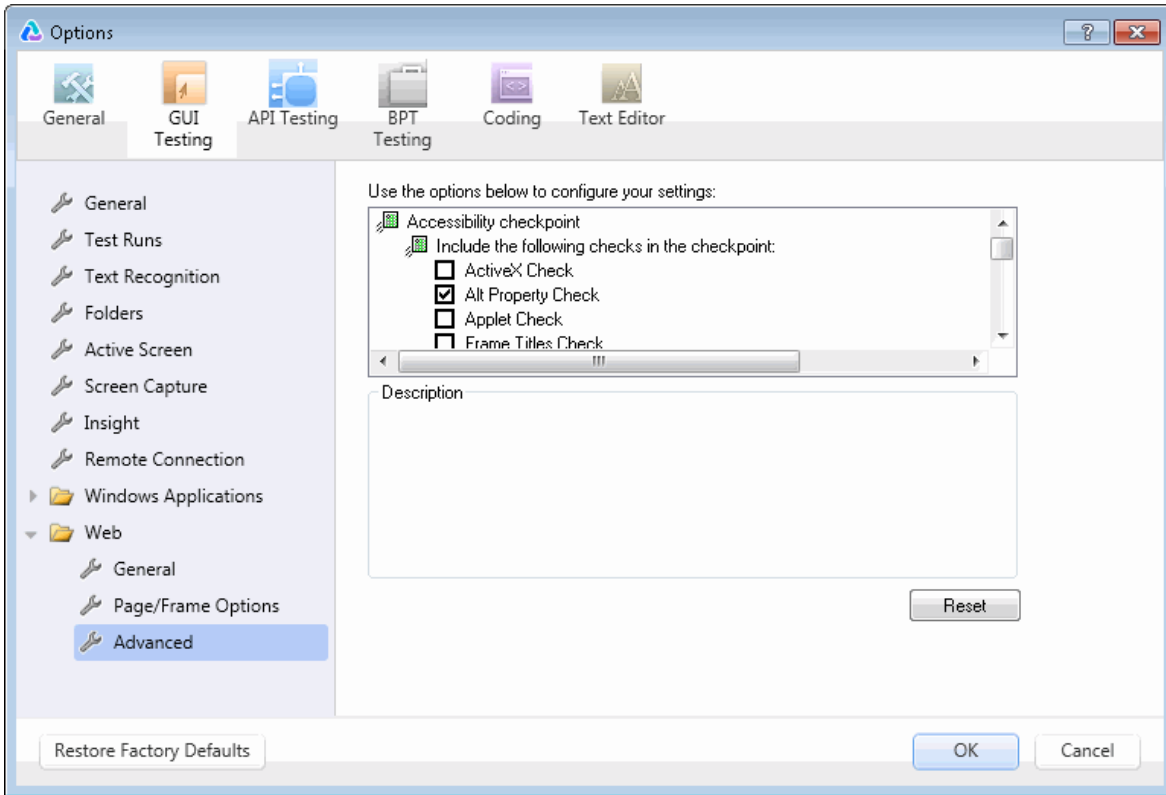
User interface elements are described below:

UI Element	Description
<p>Create a new Page test object for</p>	<p>These options instruct UFT when to create a new Page object in the object repository while recording.</p>
<p>Create a new Frame test object for</p>	<p>These options instruct UFT 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).</p>
<p>Every navigation</p>	<p>Instructs UFT to create a new Page object every time a navigation is performed in a Web page. (Relevant only for Create a new Page test object for.)</p>
<p>Different test object descriptions</p>	<p>Instructs UFT to create a new Page test object for pages with different test object descriptions, according to the properties defined for the Page test object.</p> <div style="background-color: #f0f0f0; padding: 5px; margin-top: 10px;"> <p>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.</p> </div>
<p>Different URLs or a change in data transfer</p>	<p>Instructs UFT 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> <ul style="list-style-type: none"> • Ignore non user-input data - Get. Instructs UFT 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, UFT does not create a new Page test object. • Ignore non user-input data - Post. Instructs UFT 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, UFT does not create a new Page test object. • Ignore user-input data - Get. Instructs UFT 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, UFT does not create a new Page test object. • Ignore user-input data - Post. Instructs UFT 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, UFT does not create a new Page test object.

UI Element	Description
Use additional Page information	Instructs UFT to use additional properties of the test object to identify an existing Page test object. <div style="border: 1px solid gray; padding: 5px; background-color: #f0f0f0;"> Tip: Select this option to instruct UFT to recognize existing pages when the Back and Forward navigation buttons are used. </div>
Reset All	Resets 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.

Web > Advanced Pane (Options Dialog Box)

This pane enables you to modify how UFT records and runs tests and business components on Web sites.



To access	<ol style="list-style-type: none"> 1. Make sure that a GUI test is open. 2. Select Tools > Options > GUI Testing tab > Web > Advanced node.
Important information	The Accessibility checkpoint and Automatic Page checkpoint options are not relevant for keyword components.
See also	<ul style="list-style-type: none"> • "Web > General Pane (Options Dialog Box)" on page 58 • "Web > Page/Frame Options Pane (Options Dialog Box)" on page 61

User interface elements are described below:

Accessibility Checkpoint (Tests Only)

Accessibility checkpoints enable you 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.

For general information about accessibility checkpoints, see the section on Accessibility Checkpoints in the *HP Unified Functional Testing User Guide*.

UI Element	Description
ActiveX Check	Checks whether the page or frame contains ActiveX objects. If so, UFT 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 UFT 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, UFT 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 UFT 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, UFT 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, UFT 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, UFT sends a warning and displays the table format and the tags used in each cell in the run results.
Add an automatic accessibility checkpoint to each Web page while recording	Instructs UFT to automatically add an accessibility checkpoint to each Web page while recording, using the checks selected in the option above. Note: This option is not supported for Firefox or Chrome browsers.

Automatic Page Checkpoint (Tests Only)

Note: Automatic page checkpoints are not supported for Firefox or Chrome browsers.

These options enable you to check that expected and actual page properties are identical, by instructing UFT 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.

UI Element	Description
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 in the Web pane of the Options dialog box (see " Web > General Pane (Options Dialog Box) " on page 58, 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 " Web > General Pane (Options Dialog Box) " on page 58).
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.
Create a checkpoint for each Web page while recording	Instructs UFT to automatically add a Page checkpoint for each Web page navigated during the recording process.
Ignore automatic checkpoints while running tests	Instructs UFT to ignore the automatically added Page checkpoints while running your test.

Record Settings

These options enable you to set preferences for recording Web objects.

Note: If UFT 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 details, see "[Event Recording Configuration for Web-](#)

Based Applications" on page 39.

UI Element	Description
Enable Web support for Microsoft Windows Explorer	<p>When selected, UFT treats relevant objects in Microsoft Windows Explorer as Web objects. When cleared, UFT does not record events on Web pages displayed in Microsoft Windows Explorer.</p> <p>Note: After modifying this setting, for the change to take effect, you must close all instances of Microsoft Windows Explorer (confirm that all <code>explorer.exe</code> processes are closed in the Windows Task Manager or restart the computer) and then restart UFT.</p>
Record coordinates	Records the actual coordinates relative to the object for each operation.
Record MouseDown and MouseUp as Click	<p>Records a Click method for mouseup and mousedown events.</p> <p>Note: For Web, UFT records RightClick and MiddleClick methods for most Web objects. Therefore, this option is relevant only for clicks made using the left mouse button.</p>
Record Navigate for all navigation operations	<p>Records a Navigate statement each time a Frame URL changes.</p> <p>Note: This option is not supported for Firefox or Chrome browsers.</p>
Use standard Windows mouse events	<p>Instructs UFT to use standard Windows mouse events instead of browser events for the following events:</p> <ul style="list-style-type: none"> • OnClick • OnMouseDown • OnMouseUp <p>Note:</p> <ul style="list-style-type: none"> • Use this option only if the events are not properly recorded using browser events. • For Web, UFT 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.

Run Settings

These options enable you to set preferences for working with Web objects during a run session.

UI Element	Description
Browser cleanup	<p>Closes all open browsers after the current run or iteration ends.</p> <p>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 UFT was opened.</p> <p>Note:</p> <p>This option is only applicable when a test iteration finishes naturally by running all test steps, and not by using the ExitTestIteration method.</p>

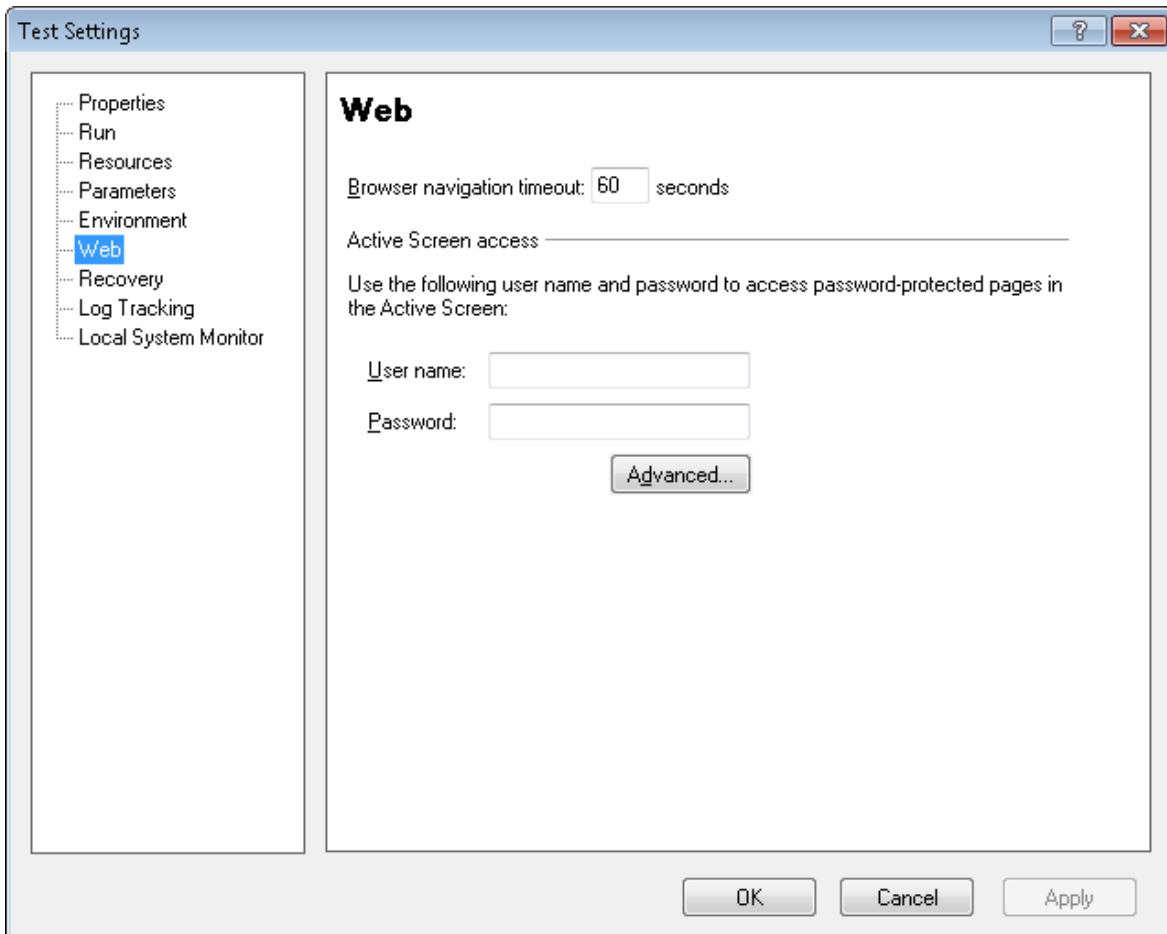
UI Element	Description
	<p>If you use the ExitTestIteration method, add a Wait statement just before the ExitTestIteration step.</p>
Run only click	<p>Determines whether a Click operation is run on the application by sending MouseDown, MouseUp, and Click events, or by sending only a Click event. This option is relevant only for Click operations when Event is selected for the 'Replay type' option. It is relevant only for left-button mouse clicks.</p>
Replay type	<p>Configures how to run mouse operations according to the selected option:</p> <ul style="list-style-type: none"> • Event. Runs mouse operations using browser events. • Mouse. Runs mouse operations using the mouse, and keyboard operations using the keyboard.
Run using source index	<p>Instructs UFT to learn and store the source index value when learning Web test objects, and to use that value during a run session to improve performance. During run sessions, UFT uses the learned source index value to return the relevant DOM element from the application and then verifies that this object matches the test object description. This operation is faster than searching the entire DOM for objects that match the test object description.</p> <p>If the returned object does not match the description, the source index is ignored and UFT attempts to identify the object using the standard object identification process.</p> <p>The source index is also ignored during a run session in the following scenarios:</p> <ul style="list-style-type: none"> • The description for a test object includes the XPath or css identifier. For details, see "Web Object Identifiers" on page 42. • If you manually define an Index ordinal identifier for the test object. For details, see the section on the Index Ordinal Identifier in the <i>HP Unified Functional Testing User Guide</i>. • The run session is performed using a browser other than Internet Explorer. <p>Note: This option is not supported for Firefox or Chrome browsers.</p>
Resize browser on run if resized during a recording session	<p>If you select this option and then resize the browser during a recording session, UFT resizes the browser to this size when the first time you run the recorded steps. 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.</p> <p>Note:</p> <ul style="list-style-type: none"> • 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, UFT does not change the browser size when a run session begins. If you run your steps with this option cleared, and then select this option again, UFT opens the browser to the same size as the previous run session, regardless of any resizing information you provided when recording the steps.
Learn and run using automatic XPath identifiers	<p>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, UFT 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.</p> <p>Additionally, if the description for a test object includes the XPath or CSS identifier, or if the Run using</p>

UI Element	Description
(Web Add-in only)	source index option is selected, then this option is ignored for that object. For details, see " Web Object Identifiers " on page 42.

Web Pane (Test/Business Component Settings Dialog Box / Application Area - Additional Settings Pane)

This pane provides options for recording and running tests on Web sites.

The following image displays the Web pane of the Test Settings dialog box. The content of this pane is identical to the content of the Web pane of the Business Component Settings dialog box and the Application Area's Additional Settings pane.



To access	Do one of the following: <ul style="list-style-type: none">• For a test or component:
------------------	---

	<p>Select File > Settings > Web node.</p> <ul style="list-style-type: none"> For an application area: Open the application area and select Additional Settings > Web in the sidebar.
Important information	<ul style="list-style-type: none"> The Web pane is available only if the Web Add-in is installed and loaded. Recording events. 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 details, see "Event Recording Configuration for Web-Based Applications" on page 39. Business components. For business components, the settings displayed in this pane are read-only. To change the Web pane settings for a business component, open its associated application area and use the application area's Additional Settings > Web pane.

User interface elements are described below:

UI Element	Description
Browser navigation timeout	Sets the maximum time (in seconds) that UFT waits for a Web page to load before running a step in the test.
User name (tests only)	The user name for password-protected resources that use a standard authentication mechanism. For details, see " Advanced Authentication Dialog Box " on the next page.
Password (tests only)	The password for password-protected resources that use a standard authentication mechanism. For details, see " Advanced Authentication Dialog Box " on the next page.
Advanced (tests only)	Opens the " Advanced Authentication Dialog Box " (described on page 70, which enables you to manually log in to your Web site to enable access to password-protected resources that use an advanced authentication mechanism.

Advanced Authentication Dialog Box

This dialog box enables you to define login information, so that the Active Screen can access the resources on a site, where the automatic Active Screen login mechanism may not be sufficient.



<p>To access</p>	<p>In the "Web Pane (Test/Business Component Settings Dialog Box / Application Area - Additional Settings Pane)", click Advanced.</p> <p>For details on the Web pane, see "Web Pane (Test/Business Component Settings Dialog Box / Application Area - Additional Settings Pane)" on page 68.</p>
<p>Important information</p>	<ul style="list-style-type: none"> • After closing this dialog box, refresh the Active Screen by selecting a new step in the Keyword View or select View > Active Screen to re-display 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. • When you log in using the Advanced Authentication mechanism, you remain logged in to the site for the duration of the UFT session. If you close and reopen UFT and then reopen your test, you must log in again. • 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.
See also	<ul style="list-style-type: none"> • "Web Pane (Test/Business Component Settings Dialog Box / Application Area - Additional Settings Pane)" on page 68 • "Active Screen Dialog Box" on page 54 • "Accessing Password-Protected Resources in the Active Screen" on page 37

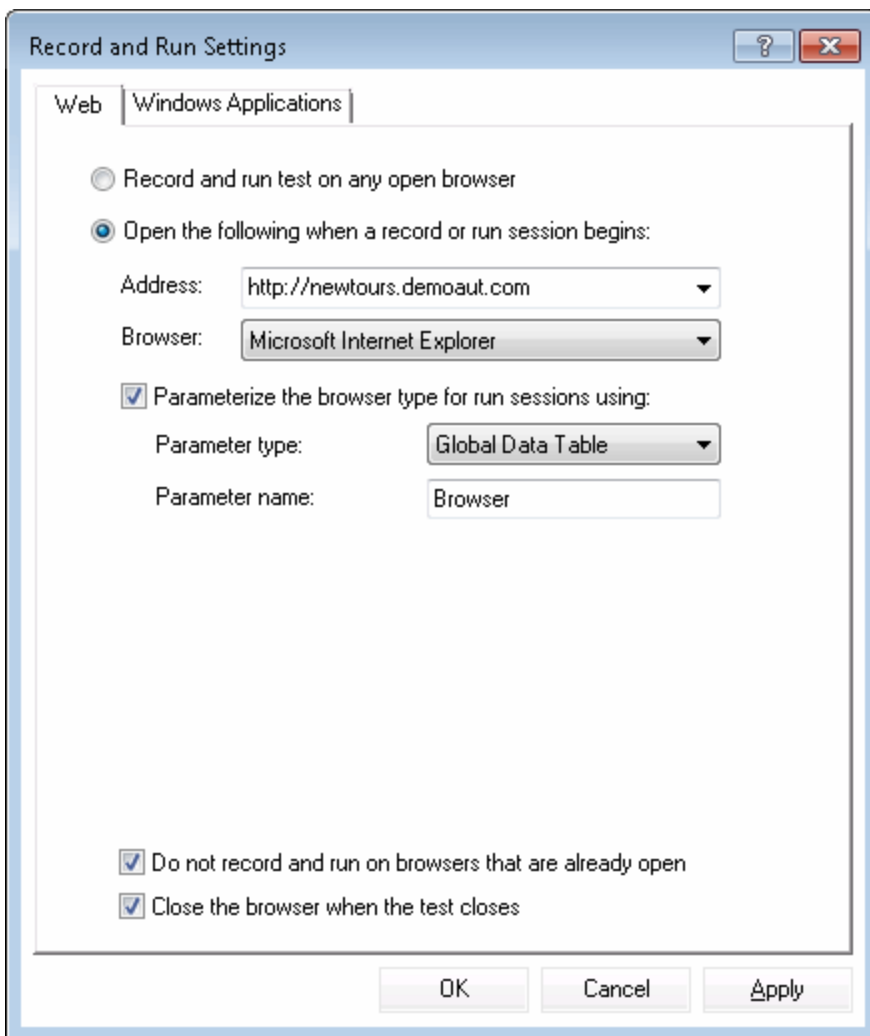
User interface elements are described below (unlabeled elements are shown in angle brackets):

UI Element	Description
<browser window>	<p>The default Web page for the test, displayed according to the following guidelines:</p> <ul style="list-style-type: none"> • 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. <p>Use this window to enter your login information, and after logging into your application, click Close. The login session remains open for the remainder of your UFT session (or until the Web site's inactivity timeout is exceeded).</p>
Address	The URL address of the Web page. 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 .

Web Tab (Record and Run Settings Dialog Box)

This tab enables you to set options that instruct UFT which applications to open when you begin to record or run your test.

You can instruct UFT to open and record on applications from more than one environment. You can also create steps on more than one browser tab, if your browser supports tabbed browsing.





To access	Select Record > Record and Run Settings and select the Web tab.
Important information	<ul style="list-style-type: none"> • The Web tab is available only when the Web add-in is installed and loaded. UFT uses the settings in this tab when recording and running tests or business components on Web, .NET Web Forms, PeopleSoft, and Web-based SAP objects. • Siebel. For Siebel objects, UFT uses the settings in the Siebel tab of the Record and Run Settings dialog box, available when the Siebel Add-in is installed and loaded. For details, see "Siebel Tab (Record and Run Settings Dialog Box)" on page 333. • Environment variables. You can also use special, predefined environment variables to specify the applications or browsers you want to use for your test. For details, see "Environment Variables in Record and Run Settings" on page 26. • If you have the Add-ons Manager enabled, if you set Firefox to open to a specific URL, Firefox does not open to the specified URL at the beginning of a recording or run session. Workaround: Disable the Add-ons Manager before using Firefox.

Relevant tasks	<ul style="list-style-type: none"> • "How to Modify Event Recording Configuration for Web-Based Applications" on page 51 • "Environment Variables for a Web-Based Environment" on page 55 • "How to Define Record and Run Settings for UFT Add-ins" on page 29
See also	<ul style="list-style-type: none"> • "Considerations for Testing Applications with Embedded Web Browser Controls" on page 75

User interface elements are described below:

UI Element	Description
Record and run test on any open browser	<p>Instructs UFT to record and run on any open (supported) Web browser installed locally on your computer. (For details on supported browsers, see the <i>HP Unified Functional Testing Product Availability Matrix</i>, available from the HP Support Matrix page (requires an HP passport)..</p> <p>Note: You must open the Web browser after you open UFT.</p> <p>Tip: You can instruct UFT to ignore ALM browsers or other browsers that are open to a specified URL or have a specific title. For details, see "Web > General Pane (Options Dialog Box)" on page 58.</p>
Open the following address when a record or run session begins	<p>Instructs UFT 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 details, see "Environment Variables in Record and Run Settings" on page 26.</p>
Open the following browser when a record or run session begins	<p>Instructs UFT to open the specified browser type when recording or running a test:</p> <p>Note:</p> <ul style="list-style-type: none"> • This list shows only browsers currently installed on your computer and the Apple Safari browser, which UFT can access on a remote Mac computer. • If you define a value for the BROWSER_ENV environment variable, that value overrides the value specified here during a run session. For details, see "Environment Variables for a Web-Based Environment" on page 55 and "Environment Variables in Record and Run Settings" on page 26. • If you select Mozilla Firefox, UFT opens the latest version of Mozilla Firefox that is both installed on your computer and supported by UFT. <ul style="list-style-type: none"> • If you want to open a specific version of Firefox that is not available in this list, you can use the BROWSER_ENV environment variable as described in "Environment Variables for a Web-Based Environment" on page 55. • For a list of all supported browser versions, see the <i>HP Unified Functional Testing Product Availability Matrix</i>, available from the HP Support Matrix page (requires an HP passport).
Parameterize the browser type for run sessions using:	<p>Enables you to set a parameter for the browser on which to run the test. You must specify the following:</p> <ul style="list-style-type: none"> • Parameter type: Global Data Table parameter or Test Parameter <p>Note:</p>

UI Element	Description
	<p>When you select the type of parameter, UFT automatically adds a column to the Global tab of the Data pane or a test parameter.</p> <p>The value you enter for this parameter is the same as the environmental variables for the browser types. For details, see "Environment Variables for a Web-Based Environment" on page 55.</p> <ul style="list-style-type: none"> • Parameter Name: The name of the column in the Global tab of the Data pane or the name of the test parameter
<p>Host Name or IP (Remote Connection Details)</p>	<p>The host name or IP address of the Mac computer to which UFT connects.</p> <p>By default, UFT connects to the Mac using port 8822. To use a different port, append the port number to the host name: <i><hostname>:<port number></i>.</p> <p>Make sure to configure the same port number on the Mac, in the UFT port option in the UFT Connection Agent preferences.</p> <p>For more details, see "How to Connect to a Remote Mac Computer" on page 480.</p> <p>Note:</p> <ul style="list-style-type: none"> • UFT uses this address whenever the test runs, whether you run it manually in UFT or as part of an ALM test set, or any other way. • If you want to test the connection to the host you specify, click the Remote Connection button  in the UFT toolbar, enter the host name or IP in the dialog box that opens, and click Connect. • If you define a value for the REMOTE_HOST environment variable, that value overrides the value specified here during a run session. <p>Available only if you select Apple Safari (on remote Mac computer) from the list of browsers.</p>
<p>Use SSL (Remote Connection Details)</p>	<p>Instructs UFT to use SSL when initiating the connection to the remote Mac computer.</p> <p>If the UFT Connection Agent on the Mac is configured to expect an SSL connection, you must select the Use SSL option.</p> <p>To use an SSL connection to secure the communication between UFT and the Mac, make sure that the relevant certificates and key are defined in the Options dialog box in UFT (Tools > Options > GUI Testing > Remote Connection) and in the UFT Connection Agent preferences on the Mac. For details, see the "Securing the Communication With the Remote Mac Computer" on page 473.</p> <p>Note: If you define a value for the USE_SSL environment variable, that value overrides the value specified here during a run session. For details, see "Environment Variables in Record and Run Settings" on page 26.</p> <p>Available only if you select Apple Safari (on remote Mac computer) from the list of browsers.</p>
<p>When run session ends(Remote Connection Details)</p>	<p>Specifies the procedures for UFT to perform at the end of a run session.</p> <ul style="list-style-type: none"> • Disconnect. Specifies whether to close UFT's connection to the remote Mac. <p>For example:</p>

UI Element	Description
<p>Available only if you select the Apple Safari browser</p>	<ul style="list-style-type: none"> Select this option if a subsequent run session needs to connect to a different Mac, or if you want to release the port so another UFT instance can use it. Clear this option if you are running multiple tests but only the first one sets up the remote connection. <p>While editing tests in UFT, you can see the connection status in the Remote Connection dialog box (opened from the Remote Connection button  in the UFT toolbar). You can also use this dialog box to manually close the connection to the Mac.</p> <ul style="list-style-type: none"> Close the browser. Specifies whether to close the Apple Safari browser. <p>For example:</p> <ul style="list-style-type: none"> Select this option if the test is independent from all other tests. Clear this option during the design stages of your test. The browser remains open after you run the test, and you can continue working with the browser while you edit the test. Clear this option if you are running multiple tests but only the first test contains a step that runs the Safari browser. <p>These options are available only if you select Apple Safari (on remote Mac computer) from the list of browsers.</p>
<p>Do not record and run on browsers that are already open</p>	<p>Instructs UFT 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 UFT).</p> <p>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 UFT to close the browser window specified in the Address box when the test closes.</p> <p>Examples of when to select or clear this option:</p> <ul style="list-style-type: none"> Select this option if you are finished designing your test, and it runs independently from all other tests. Clear this option during the design stages of your test. The browser remains open after you run the test, and you can continue working with the browser while you edit the test. Clear this option if you plan to run multiple tests on the same browser, but only the first test includes a step to open the browser. <p>Not available if you select Apple Safari (on remote Mac computer) from the list of browsers.</p>

Considerations for Testing 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.
- Register your browser control application (using the Register Browser Control Utility) so that UFT recognizes your Web object when recording or running tests. For details, see ["Register Browser](#)

[Control Utility" on page 56.](#)

- Make sure that the application is opened after UFT, and start recording.

Troubleshooting and Limitations - Web-Based Application Support

This section contains general troubleshooting and limitation information about the Web add-in, and includes the following sections:

- ["Test Objects, Methods, and Properties" below](#)
- ["Creating and Running Testing Documents" below](#)
- ["Recognition of WebTable Test Objects" on the next page](#)
- ["Registering Browsers" on page 79](#)
- ["Checkpoints, Output Values, and the Active Screen" on page 79](#)
- ["Working with Multiple Web Browsers" on page 79](#)

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.

- UFT 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`.

Creating and Running Testing Documents

- If you use the `Tab` key when recording password fields in the `AutoComplete` dialog box, UFT may record incorrectly.

Workaround: Press **ENTER** after entering the user name or click the button for logging in.

- When UFT 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 business component.

Recognition of WebTable Test Objects

By default, when using the UFT Web Add-in, UFT 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
<UFT 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 UFT treats as an abstract table:

Edit the **IsHPAbstractTable** JavaScript located in:

<UFT installation folder>\dat\Extensibility\Web\Toolkits\HPInternal\HPAbstractTable.js.

The sample **IsHPAbstractTable** JavaScript function below causes UFT 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 UFT 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 UFT is currently handling.)

Caution: HPAbstractTable.js affects the way UFT 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.

Registering Browsers

UFT supports applications with embedded browsers only for applications that embed Internet Explorer.

Checkpoints, Output Values, and the Active Screen

- 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 (**Run > Update Run Mode**) of your test or business component before you run a test or business 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 instead of a Web browser, your checkpoints may fail.
Workaround: Insert checkpoints while recording.

Working with Multiple Web Browsers

UFT retrieves the CSS property values from the browser. When designing tests or components that will run on different browsers, keep in mind that different browsers may have different CSS functionality and return different property values for the same object.

Problem

When running steps that are intended to be performed on different browsers, and UFT tries to perform the step intended for the second browser before the second browser has finished loading, UFT 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, UFT 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 UFT tries to perform a step intended for that browser, UFT 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.

Chapter 2: Windows-Based Application Support

This chapter includes:

- [Windows-Based Application Support - Overview](#) 82
- [UFT Configuration for Windows-Based Applications](#) 82
- [Record and Run Settings for Windows-Based Applications](#)83
- [Environment Variables for Windows-based Applications](#) 84
- [Windows Applications > General Pane \(Options Dialog Box > GUI Testing Tab\)](#) 85
- [Windows Applications > Advanced Pane \(Options Dialog Box > GUI Testing Tab\)](#) 87
 - [Considerations for Advanced Windows-based Application Testing](#)91
- [Windows Applications Tab \(Record and Run Settings Dialog Box\)](#) 93
 - [Application Details Dialog Box](#) 96
 - [Record and Run Setting Guidelines for Windows-Based Add-ins](#) 98

Windows-Based Application Support - Overview

UFT provides a number of add-ins for testing Windows-based applications.

The way you configure many of your UFT options is the same or similar for most UFT Windows-based add-ins (as well as for the built-in standard Windows testing support). For details, see "[UFT Configuration for Windows-Based Applications](#)" below.

Many UFT add-ins rely on the settings in the "[Windows Applications Tab \(Record and Run Settings Dialog Box\)](#)" (described on page 93) to determine on which applications UFT records and runs. For some add-ins, these settings may also affect the applications that UFT recognizes for certain operations while in edit mode, such as using the Object Spy or other pointing hand operations.

You can also 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.

There may also be additional issues that you need to address to ensure that UFT recognizes your objects properly during record, run, and/or pointing hand operations. For details, see "[Record and Run Setting Guidelines for Windows-Based Add-ins](#)" on page 98.

For details about standard Windows testing support, see "[Standard Windows Support -Quick Reference](#)" on page 347

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

- "[ActiveX Add-in - Quick Reference](#)" on page 140
- "[Delphi Add-in - Quick Reference](#)" on page 146
- "[.NET Windows Forms Support - Quick Reference](#)" on page 116
- "[PowerBuilder Add-in](#)" on page 234
- "[Qt Add-in](#)" on page 238
- "[Windows-based SAP Support - Quick Reference](#)" on page 259
- "[Stingray Add-in](#)" on page 350
- "[Terminal Emulator Add-in](#)" on page 377
- "[VisualAge Smalltalk Add-in](#)" on page 424
- "[Visual Basic Add-in](#)" on page 428
- "[Windows Runtime Add-in](#)" on page 504

UFT Configuration for Windows-Based Applications

Use the **Windows Applications** pane of the Options dialog box (**Tools > Options > GUI Testing tab > Windows Applications** node) to configure general and advanced Windows-based application options, as

follows:

- **General options.** Configure how UFT records and runs tests and business components for Windows-based applications, such as Standard Windows, ActiveX, .NET Windows Forms, WPF, SAP GUI for Windows, Visual Basic applications, and Windows Runtime. For details, see "[Windows Applications > General Pane \(Options Dialog Box > GUI Testing Tab\)](#)" on page 85.

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 disable Active Screen captures entirely. You set this preference in the **Active Screen** pane of the Options dialog box (**Tools > Options > GUI Testing tab > Active Screen** node). The less information saved, the faster your recording times will be.

This option is not relevant for keyword components.

For details, see the section describing the Active Screen pane in the *HP Unified Functional Testing User Guide*.

- **Advanced options.** Modify how UFT records and runs tests or business components on Windows-based applications, such as ActiveX or Visual Basic. For details, see "[Windows Applications > Advanced Pane \(Options Dialog Box > GUI Testing Tab\)](#)" on page 87.

Record and Run Settings for Windows-Based Applications

You can set options that affect how you start creating and running tests for Windows-based applications in the "[Windows Applications Tab \(Record and Run Settings Dialog Box\)](#)" (described on page 93). These options instruct UFT which applications to open when you begin to record or run your test.

Additionally, you can specify the applications on which you want to record in the "[Application Details Dialog Box](#)" (described on page 96). You can use environment variables to define the details for the Windows-based applications on which you want to record and run tests. If you do this, use the appropriate variable names as specified in "[Environment Variables for Windows-based Applications](#)" on the next page.

Note: The Record and Run Settings dialog box applies only to tests. Record settings for business 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 business components. For details on the Applications pane in the Additional Settings pane of an application area, see the *HP Unified Functional Testing User Guide*.

Environment Variables for Windows-based Applications

You can use environment variables to define the details for the Windows-based applications on which you want to record and run tests. If you do this, use the appropriate variable names as specified below.

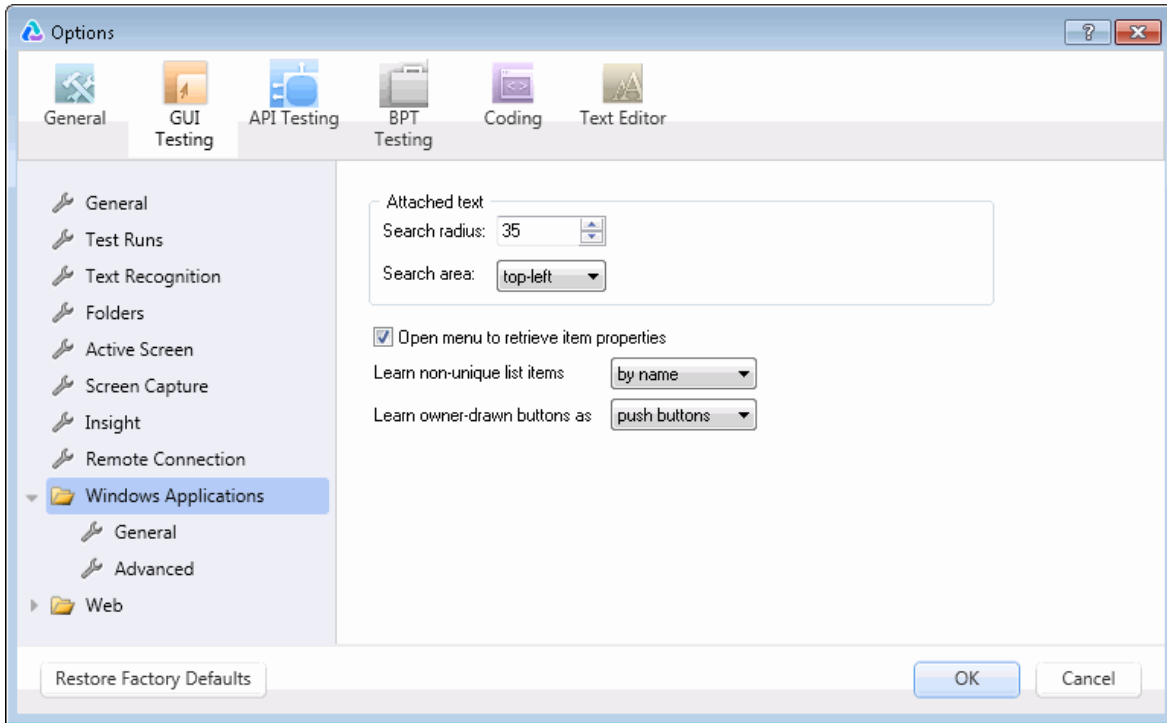
Note:

- For details on environment variables and how to use them in tests, see ["Environment Variables in Record and Run Settings"](#) on page 26.
- The environment variables described in this section correspond with the settings you define in the ["Application Details Dialog Box"](#) (described on page 96).

Option	Variable Names	Description
Application	EXE_ENV_1 EXE_ENV_10	The executable files on which UFT 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: <ul style="list-style-type: none"> • 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: <ul style="list-style-type: none"> • 0 (do not record on descendant processes) • 1 (record descendant processes)

Windows Applications > General Pane (Options Dialog Box > GUI Testing Tab)

This pane enables you to configure how UFT records and runs tests and business components for Windows-based applications.



To access	Select Tools > Options > GUI Testing tab > Windows Applications > General node.
Important information	The Restore Factory Defaults button resets all Options dialog box options to their defaults.
See also	<ul style="list-style-type: none"> • "Windows Applications > Advanced Pane (Options Dialog Box > GUI Testing Tab)" on page 87 • "Windows Applications Tab (Record and Run Settings Dialog Box)" on page 93 • "Windows-Based Application Support - Overview" on page 82 • "UFT Configuration for Windows-Based Applications" on page 82

User interface elements are described below:

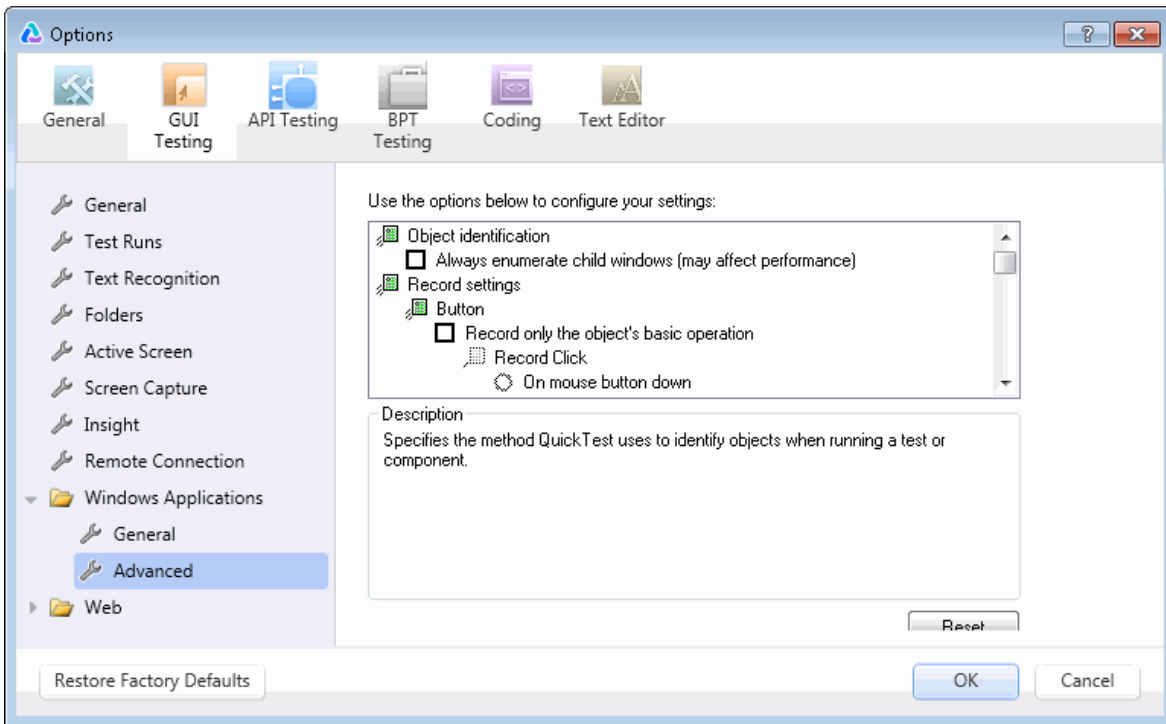
UI Elements	Description
Attached text	The search criteria that UFT uses to retrieve an object's attached text. An object's attached text is the closest static text within a specified radius from a specified point. The retrieved attached text is saved in the object's corresponding text or attached text identification property.

UI Elements	Description
	<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. The maximum distance, in pixels, that UFT searches for attached text.</p> <p>Search area. The point on an object from which UFT 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 UFT to open Standard Windows menu objects before retrieving menu item properties during a run session.</p> <p>Note:</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 ExpandMenu property in a test or function library to set this behavior for a specified menu object. For details, see the specific test object methods and properties in the <i>HP UFT Object Model Reference for GUI Testing</i>.
Learn non-unique list items	<p>Determines what UFT learns when more than one item (in a Standard Windows list or tree) has an identical name.</p> <ul style="list-style-type: none"> • by name. Learns the item's name. During a run session, UFT finds and selects the first instance of the name, regardless of the item chosen when learning. Select this option if the all items with the same name have the identical behavior. • by index. Learns the item's index number. Select this option if items with the same name do not necessarily have the identical behavior.
Learn owner-drawn buttons as	<p>Instructs UFT how to identify and learn Standard Windows 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

UI Elements	Description
	<p>Note: If you select objects, UFT learns each owner-drawn button as a WinObject. (When working with tests, UFT can also learn an owner-drawn button as a virtual object, if you define the virtual object. For details, see the section on learning virtual objects in the <i>HP Unified Functional Testing User Guide</i>.)</p>

Windows Applications > Advanced Pane (Options Dialog Box > GUI Testing Tab)

This pane enables you to modify how UFT records and runs tests or business components on Windows-based applications.



To access	Select Tools > Options > GUI Testing tab > Windows Applications > Advanced node.
Important Information	For advanced Windows-based application testing information, intended for users with expertise in the Win32 API and the Windows messages model, see " Considerations for Advanced Windows-based Application Testing " on page 91.
See also	" Windows-Based Application Support - Overview " on page 82

User interface elements are described below:

UI Elements	Description
Object Identification node	
Always enumerate child windows (may affect performance)	<p>Instructs UFT to enumerate all child windows when recording a running a test or business component.</p> <p>Caution: 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 details, see "Considerations for Advanced Windows-based Application Testing" on page 91.</p>
Record Settings node	
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. <p>Note: This option is cleared by default and should be used only when the default recording method does not meet your needs. For details, see "Considerations for Advanced Windows-based Application Testing" on page 91.</p> • 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). <p>Note: This option is enabled only when Record only the object's basic operation is selected.</p> <p>Default = On mouse button up.</p>
List	<p>Defines record settings for Windows-based list objects (for example, WinList, WinListView, and VbList):</p> <ul style="list-style-type: none"> • Record only the object's basic operation. Enables simplified recording on the list. Using this mode may improve recognition of user operations in non-standard cases. <p>Note: This option is cleared by default and should be used only when the default recording method does not meet your needs. For details, see "Considerations for Advanced Windows-based Application Testing" on page 91.</p> • 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). <p>Default = On mouse button up.</p>
Menu	<p>Defines record settings for menu objects:</p> <ul style="list-style-type: none"> • Enable recording. Specifies whether UFT records operations on menu controls. For example, you may want UFT to ignore the actual process of selecting a menu to open another window. This option is selected by default. • Menu recording mode. Specifies whether UFT verifies or ignores menu initialization events before recording operations on menu controls.

UI Elements	Description
	<p>Note: This option is enabled only when Enable recording is selected. For details, see "Considerations for Advanced Windows-based Application Testing" on page 91.</p> <p>Default = Verify menu initialization event.</p>
Object	<p>Defines record settings for objects recognized as WinObject test objects:</p> <ul style="list-style-type: none"> • Record only the object's basic operation. Enables simplified recording on the WinObject test object. Using this mode may improve recognition of user operations in non-standard cases. <p>Note: This option is cleared by default and should be used only when the default recording method does not meet your needs. For details, see "Considerations for Advanced Windows-based Application Testing" on page 91.</p> • 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). <p>Note: This option is enabled only when Record only the object's basic operation is selected.</p> <p>Default = On mouse button down.</p>
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. <p>Note: This option is cleared by default and should be used only when the default recording method does not meet your needs. For details, see "Considerations for Advanced Windows-based Application Testing" on page 91.</p> • 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). <p>Note: This option is enabled only when Record only the object's basic operation is selected.</p> <p>Default = On mouse button up.</p>
Toolbar	<p>Defines record settings for toolbar objects:</p> <ul style="list-style-type: none"> • Record only the object's basic operation. Enables simplified recording on the toolbar. Using this mode may improve recognition of user operations in non-standard cases. <p>Note: This option is cleared by default and should be used only when the default recording method does not meet your needs. For details, see "Considerations for Advanced Windows-based Application Testing" on page 91.</p> • 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

UI Elements	Description
	<p>button up).</p> <p>Note: This option is enabled only when Record only the object's basic operation is selected.</p> <p>Default = On mouse button up.</p>
Tree view	<p>Defines record settings for tree view objects:</p> <ul style="list-style-type: none"> • Record only the object's basic operation. Enables simplified recording on the tree view. Using this mode may improve recognition of user operations in non-standard cases. This option is cleared by default and should be used only when the default recording method does not meet your needs. For details, see "Considerations for Advanced Windows-based Application Testing" on the next page. • 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). Note: 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.
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. Note: This option is cleared by default and should be used only when the default recording method does not meet your needs. For details, see "Considerations for Advanced Windows-based Application Testing" on the next page. • 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). Note: 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 UFT should use to detect the keyboard state. Default = Standard. For details, see "Considerations for Advanced Windows-based Application Testing" on the next page.
Utility object	<p>Defines record settings for utility objects:</p>

UI Elements	Description
	<p>Record SystemUtil.Run commands. Specifies whether UFT records SystemUtil.Run commands when you open an application during a recording session.</p> <p>This option is selected by default. For details on the SystemUtil.Run method, see the <i>HP UFT Object Model Reference for GUI Testing</i>.</p>
Run Settings node	
Edit Box	<p>Defines run settings for Edit objects:</p> <ul style="list-style-type: none"> • Click Edit box before inserting text. Specifies whether UFT performs a Click operation to set the focus in an edit box before inserting text in it while running a test or business component. This option is cleared by default. • Use keyboard events to perform Set operations. When selected, instructs UFT to simulate keyboard events when performing Set operations on edit boxes during a run session. When cleared, instructs UFT to use API or Window messages for edit box Set operations. This option is cleared by default.
Bottom area	
Description	Describes the selected option.
Reset button	Resets all options in the pane to their default values.

Considerations for Advanced Windows-based Application Testing

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 options described in "[Windows Applications > Advanced Pane \(Options Dialog Box > GUI Testing Tab\)](#)" on page 87.

Always enumerate child windows

If UFT does not correctly record an object in your application, you can select this option to force UFT to enumerate all windows in the system. This means that even when UFT 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, UFT records operations on Windows objects based on Windows messages sent by the application. UFT 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**, UFT records the operation performed when a `WM_LBUTTONDOWN` message is detected; if you select **On mouse button up**, UFT records the operation performed when a `WM_LBUTTONUP` message is detected.

Keyboard state detection

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

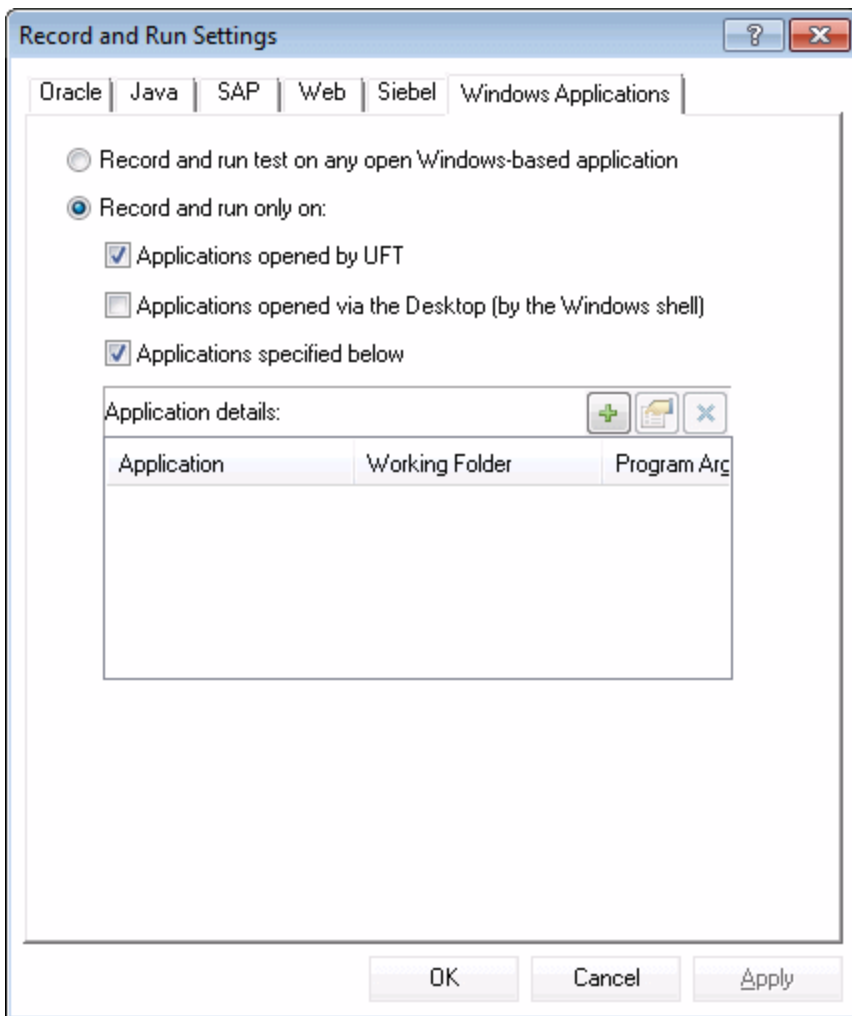
Menu recording mode

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 UFT to record menu operations only after detecting a menu initialization message. If UFT 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 UFT to always record menu operations.

Windows Applications Tab (Record and Run Settings Dialog Box)

This tab enables you to define preferences for recording and running tests on Windows-based applications.






To access	Select Record > Record and Run Settings and then select the Windows Applications tab. 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).
Important information	<ul style="list-style-type: none">• You can instruct UFT to open and record on applications from more than one environment.• For some Windows-based add-ins, the settings in this dialog box may also affect the applications that UFT recognizes for other UFT operations, such as learning objects or using the Object Spy. For details, see "Record and Run Setting Guidelines for Windows-Based Add-ins" on page 98.

	<ul style="list-style-type: none"> 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, UFT will not record or run on any Windows-based application. If you loaded other add-ins when you opened UFT, there may be additional tabs in the Record and Run Settings dialog box. If this is the case, confirm that in each additional tab, the option to record and run on any open application (upper radio button of each tab) is selected. <p>While this setting does not directly affect your record or run sessions, it prevents UFT from opening unnecessary applications when you begin record or run sessions.</p>
See also	<ul style="list-style-type: none"> "Record and Run Settings for Windows-Based Applications" on page 83 "Record and Run Setting Guidelines for Windows-Based Add-ins" on page 98 "Environment Variables for Windows-based Applications" on page 84 "Environment Variables in Record and Run Settings" on page 26 "Application Details Dialog Box" on page 96 "Record and Run Settings for Add-ins - Overview" on page 24 "Windows-Based Application Support - Overview" on page 82

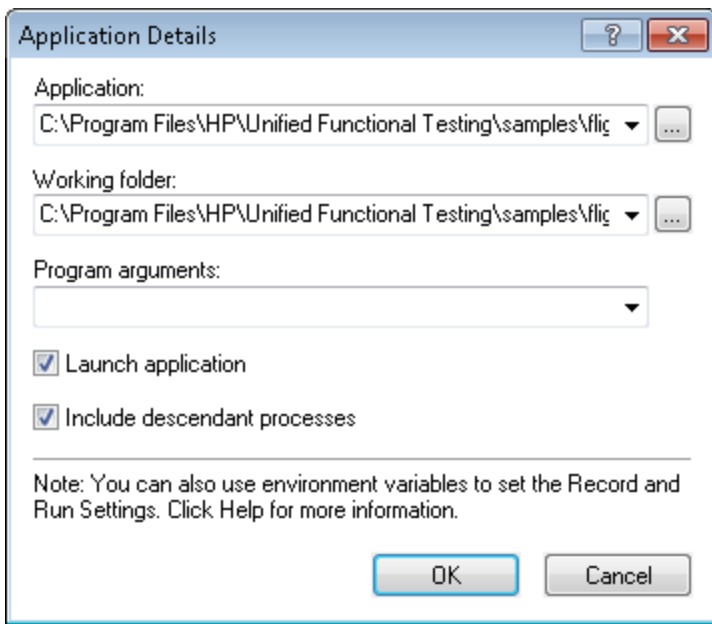
User interface elements are described below:



UI Elements	Description
Record and run test on any open Windows-based application	<p>Instructs UFT 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). UFT 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 UFT, 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, UFT 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>

UI Elements	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 UFT 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"> <p>Applications opened by UFT. This option records, recognizes, and runs only on applications invoked by UFT (as child processes of UFT). For example, applications opened during a record or run session using a <code>SystemUtil.Run</code> statement, or using a statement such as:</p> <pre>Set shell = createobject("wscript.shell") shell.run "notepad".</pre> <p>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.</p> <p>Applications specified below. This option records, recognizes, and runs only on applications listed in the Application details area.</p> <p>Tip:</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, UFT 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.
<p>Application details</p>	<p>The details of the applications on which to record and run the test. For details on the details displayed, see "Application Details Dialog Box" on the next page.</p> <p>Note: If you define values for one or more environment variables, those values override the values in the Windows Applications tab during a test run. For details, see "Environment Variables in Record and Run Settings" on page 26.</p>
	<p>Add. Opens the "Application Details Dialog Box" (described on page 96) to enable you to add an application to the application list. You can add up to ten applications.</p>
	<p>Edit. Opens the "Application Details Dialog Box" (described on page 96) to enable you to edit the application details for the selected application.</p>
	<p>Delete. Removes the selected application from the application list.</p>

Application Details Dialog Box

This dialog box enables you to add details for an application or to edit details for the application that you selected in the Windows Applications tab of the Record and Run Settings dialog box.



<p>To access</p>	<p>In the "Windows Applications Tab (Record and Run Settings Dialog Box)" (described on page 93), click the Add button  or Edit button .</p> <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>
<p>Important information</p>	<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.
<p>See also</p>	<ul style="list-style-type: none"> "Windows Applications Tab (Record and Run Settings Dialog Box)" on page 93 "Windows Applications > General Pane (Options Dialog Box > GUI Testing Tab)" on page 85 "Windows Applications > Advanced Pane (Options Dialog Box > GUI Testing Tab)" on page 87 "Record and Run Settings for Add-ins - Overview" on page 24 "Record and Run Settings for Windows-Based Applications" on page 83 "Windows-Based Application Support - Overview" on page 82

User interface elements are described below:

UI Elements	Description
Application	<p>The executable file on which to record.</p> <p>You can enter the executable file as a relative path. During the run session, UFT 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 (Tools > Options > GUI Testing tab > Folders node). For details, see the sections on setting folder testing options, and on using relative paths in the <i>HP Unified Functional Testing User Guide</i>.</p> <p>Note:</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. UFT records and runs on any application with the specified executable file name. For example, if you specify %SYSTEMROOT%\Notepad.exe, UFT 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\a.txt. In this case, UFT automatically opens the specified file in the associated application (Notepad in this example). If you use this option, UFT ignores any defined program arguments.</p>
Working folder	<p>(Optional) 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 UFT to open the application using the specified command line arguments.</p> <p>Note: This parameter is used only when Launch application is selected. If Launch application is not selected, its value has no effect.</p>
Launch application	<p>Instructs UFT whether to launch the selected application when the record and run session begins. By default, this option is selected.</p>
Include descendant processes	<p>Instructs UFT whether to record and run on processes created by the specified application during the record and run session. For example, a process that is used only as a launcher may create another process that actually provides the application functionality. This descendant process must therefore be included when recording or running tests on this application, otherwise the functionality will not be recorded, or the run session will fail.</p> <p>By default, this option is selected.</p>

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

Special considerations are detailed below for each UFT add-in that is affected by the settings in the ["Windows Applications Tab \(Record and Run Settings Dialog Box\)"](#) (described on page 93).

Add-in Environment	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. UFT recognizes ActiveX objects only in applications that are opened after changing the settings in the "Windows Applications Tab (Record and Run Settings Dialog Box)" (described on page 93).
Delphi	<ul style="list-style-type: none"> UFT recognizes only Delphi applications that have been precompiled with the Delphi agent module (<code>MicDelphiAgent.pas</code>). For details, see "How to Enable Communications Between UFT and Your Delphi Application" on page 148. 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.
.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.
.NET Windows Presentation Foundation Environment	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 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. UFT 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 (Record and Run Settings Dialog Box)" (described on page 93).
Stingray	<ul style="list-style-type: none"> In addition to the settings in the Record and Run Settings dialog box, you must also configure UFT to recognize your Stingray applications in the Stingray pane of the Options dialog box (Tools > Options > GUI Testing tab > Stingray node). For details, see "Stingray Pane (Options Dialog Box > GUI Testing Tab)" on page 371. 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"> UFT recognizes only the terminal emulator set in the Terminal Emulator pane of the Options dialog box (Tools > Options > GUI Testing tab > Terminal Emulator node). For details, see "Terminal Emulator Pane (Options Dialog Box > GUI Testing Tab)" on page 397. The Record and Run only on radio button does not affect the applications on which UFT records, recognizes, and runs.
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

Add-in Environment	Guidelines
	<p>applications that are recognized by the Object Spy and other pointing hand operations.</p> <ul style="list-style-type: none"> • UFT recognizes Visual Basic objects only in applications that are opened after changing the settings in the "Windows Applications Tab (Record and Run Settings Dialog Box)" (described on page 93).
VisualAge	<ul style="list-style-type: none"> • UFT can recognize only VisualAge Smalltalk applications that have been precompiled with the VisualAge Smalltalk agent (qt-adapter). For details, see "How to Configure the VisualAge Smalltalk Add-in" on page 427. • The Record and Run only on radio button applies only to record and run sessions. UFT recognizes all VisualAge Smalltalk objects for Object Spy and pointing hand operations, regardless of the settings in the Record and Run Settings dialog box.
Windows Runtime	<ul style="list-style-type: none"> • UFT cannot open Windows Runtime applications as part of recording or running. Therefore, the Applications opened by UFT and Applications opened via the Desktop are not supported. • If you specify a specific application for recording or running using the Applications specified below option, enter the following information: <ul style="list-style-type: none"> • For WPF or XAML-based applications: the name of the .exe process for the application • For HTML or JavaScript-based applications: WWAHOST.exe

Part 2: .NET Add-in

This section includes:

[".NET Silverlight Add-in" on page 102](#)

[".NET Web Forms Add-in" on page 108](#)

[".NET Windows Forms Add-in" on page 115](#)

[".NET Windows Presentation Foundation \(WPF\) Add-in" on page 131](#)

.NET Add-in - Overview

You can use the UFT .NET Add-in to test user interface objects (controls) in Silverlight, .NET Web Forms, .NET Windows Forms, and Windows Presentation Foundation applications. You can create and run tests and business components on these objects, and check their properties.

This section contains:

- [".NET Silverlight Add-in" on page 102](#)
- [".NET Web Forms Add-in" on page 108](#)
- [".NET Windows Forms Add-in" on page 115](#)
- [".NET Windows Presentation Foundation \(WPF\) Add-in" on page 131](#)

.NET Silverlight Add-in

This chapter includes:

- [.NET Silverlight Add-in - Quick Reference](#) 103
- [Silverlight Add-in Extensibility](#)105
- [Troubleshooting and Limitations - Silverlight Add-in](#) 106

.NET Silverlight Add-in - Quick Reference

You can use the UFT Silverlight Add-in to test user-interface objects (controls) in Silverlight applications.

The following tables summarize basic information about the Silverlight Add-in and how it relates to some commonly-used aspects of UFT.

General	
Add-in Type	<p>This is a Web-based add-in. Much of its functionality is the same as other Web-based add-ins.</p> <p>This add-in is installed as a sub add-in of the .NET Add-in.</p> <p>See "Web-Based Application Support" on page 34.</p>
Supported Environments	<p>For details on supported Silverlight environments, see the .NET Add-in section of the <i>HP Unified Functional Testing Product Availability Matrix</i>, available from the HP Support Matrix page (requires an HP passport).</p>
Important Information	<ul style="list-style-type: none"> 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.aspx Registering Microsoft sllauncher.exe. You can use the UFT Silverlight Add-in to test Silverlight out-of-browser applications. To do this you must register the Microsoft sllauncher . exe as a browser control. This executable is located in the Silverlight installation folder, for example, %ProgramFiles%\Microsoft Silverlight. You can do this using the UFT Register Browser Control Utility, which is available from Start > All Programs > HP Software > HP Unified Functional Testing > Tools > Register New Browser Control. or <UFT installation folder>\bin\SettingNewBrowserControlApplication.exe. For details, see "Registering Browser Controls" on page 37. <div style="background-color: #f0f0f0; padding: 5px; margin-top: 10px;"> <p>Note: For details on accessing UFT and UFT tools and files in Windows 8 and Windows Server 2012, see "Accessing UFT in Windows 8.X or Higher Operating Systems" on page 517.</p> </div>
Test Object Methods and Properties	<p>The Silverlight Add-in provides test objects, methods, and properties that can be used when testing objects in Silverlight applications. For details, see the Silverlight section of the <i>HP UFT Object Model Reference for GUI Testing</i>.</p>
Checkpoints and Output Values	<ul style="list-style-type: none"> See the sections describing checkpoints and output values in the <i>HP Unified Functional Testing User Guide</i>. See "GUI Checkpoints and Output Values Per Add-in" on page 512
Extending the Silverlight Add-in	<p>"Silverlight Add-in Extensibility " (described on page 105 enables you to develop support for testing third-party and custom Silverlight controls that are not supported out-of-the-box by the UFT Silverlight Add-in.</p>
Troubleshooting and Limitations	<p>"Troubleshooting and Limitations - Silverlight Add-in" on page 106.</p>
Prerequisites	

Opening Your Application	You must open UFT before opening your Silverlight application.
Add-in Dependencies	The Web Add-in must be loaded.
Other	<ul style="list-style-type: none"> • To work with the Silverlight Add-in, .NET Framework 3.0 or later must be installed on your computer. • To test your Web application in Mozilla Firefox version 33 and later, you must enable the Functional Testing Extension in one of the following ways: <ul style="list-style-type: none"> • If the Select Your Add-ons screen is displayed when you open Firefox, select to enable the Functional Testing Extension. • If the Install Add-on tab opens and displays Functional Testing Extension when opening Firefox, select the Allow this installation check box and click Continue. <p>Otherwise:</p> <ol style="list-style-type: none"> i. In Firefox, open the browser menu. ii. In the menu, click Add-ons. iii. In the Add-ons Manager tab, select the Extensions node. iv. In the Functional Testing Extension row, click the Enable button. <p>To test your Web application in Mozilla Firefox versions earlier than 33, you must enable the legacy Firefox agent:</p> <ol style="list-style-type: none"> a. Open the <UFT installation folder>\Installations\Firefox folder. b. From the Firefox folder, drag the Legacy.xpi file into Firefox. c. In Firefox, open the browser menu. d. In the menu, click Add-ons. e. In the Add-ons Manager tab, select the Extensions node. f. In the Functional Testing Extension row, disable the Functional Testing 12.50 extension and enable the extension you added to Firefox.

Configuration	
Options Dialog Box	<p>Use the Web pane. (Make sure that a GUI test is open and select Tools > Options > GUI Testing tab > Web > General node.)</p> <p>See "Web > General Pane (Options Dialog Box)" on page 58.</p>
Record and Run Settings Dialog Box (tests only)	<p>Use the Web tab. (Record > Record and Run Settings.)</p> <p>See "Web Tab (Record and Run Settings Dialog Box)" on page 71.</p>
Test Settings Dialog Box (tests only)	<p>Use the Web pane. (File > Settings > Web node)</p> <p>See "Web Pane (Test/Business Component Settings Dialog Box /</p>

	Application Area - Additional Settings Pane)" on page 68.
Custom Active Screen Capture Settings Dialog Box (tests only)	Use the Web section. (Tools > Options > GUI Testing tab > Active Screen node > Custom Level) See the section on the Custom Active Screen Capture Settings dialog box in the <i>HP Unified Functional Testing User Guide</i> .
Application Area Additional Settings pane (business components only)	Use the Web pane. In the application area, select Additional Settings > Web in the sidebar. See " Web Pane (Test/Business Component Settings Dialog Box / Application Area - Additional Settings Pane)" on page 68.

Silverlight Add-in Extensibility

UFT 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 UFT Silverlight Add-in.

If the test object class that UFT 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 UFT 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:

- UFT 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 UFT 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 UFT Extensibility Documentation program group (**Start > All Programs > HP Software > HP Unified Functional Testing > Extensibility > Documentation** or the `<UFT installation folder>\help\Extensibility folder`).

Note: For details on accessing UFT and UFT tools and files in Windows 8 and Windows Server 2012, see "[Accessing UFT in Windows 8.X or Higher Operating Systems](#)" on page 517.

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

Troubleshooting and Limitations - Silverlight Add-in

This section describes troubleshooting and limitations for the Silverlight Add-in, and contains the following sections:

- ["General" below](#)
- ["Checkpoints" below](#)
- ["Creating and editing testing documents" below](#)
- ["Running steps on Silverlight applications" on the next page](#)

General

- UFT retrieves incorrect values for the **all items** and **selection** properties for **ListBox** and **ComboBox** controls that are bound to data via a template.
- 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.
- The Silverlight Add-in is not supported on Internet Explorer 11 when the Enhanced Protected Mode is turned on.
- You cannot create a virtual object for an area in a Silverlight application.
- The Silverlight Add-in does not support 64-bit Firefox browsers.

Checkpoints

- 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.

Creating and editing testing documents

- 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
```

Running steps on Silverlight applications

- If a Web page contains a Silverlight application that is windowless and is scrolled out of view when the page opens the first time, UFT will not be able to make this application visible. (For example, in this scenario, UFT will not be able to perform an `SlvWindow.MakeVisible` step).
- In some versions of Internet Explorer, the Silverlight application becomes active only after a **Click** operation is performed. In these cases, UFT 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.

- To improve performance when running legacy tests in UFT, update your Silverlight test object descriptions to include the **devnamepath** property.

.NET Web Forms Add-in

This chapter includes:

- [.NET Web Forms Add-in - Quick Reference](#) 109
- [Considerations for Testing .NET Web Forms](#) 110
- [.NET Windows Forms Objects - Checkpoints and Output Values](#) 110
- [Accessing Internal Properties and Methods of Run-Time .NET Web Forms Objects](#) 112
- [Troubleshooting and Limitations - .NET Web Forms](#) 112

.NET Web Forms Add-in - Quick Reference

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

The following tables summarize basic information about the **.NET Web Forms** application support and how it relates to some commonly-used aspects of UFT.

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 " Web-Based Application Support " on page 34.
Supported Environments	For details on supported .NET Web Forms environments, see the .NET Add-in section of the <i>HP Unified Functional Testing Product Availability Matrix</i> , available from the HP Support Matrix page (requires an HP passport).
Test Object Methods and Properties	The .NET Add-in provides test objects, methods, and properties that can be used when testing objects in .NET Web Forms applications. For details, see the .NET Web Forms section of the <i>HP UFT Object Model Reference for GUI Testing</i> .
Checkpoints and Output Values	<ul style="list-style-type: none"> See the sections describing checkpoints and output values in the <i>HP Unified Functional Testing User Guide</i>. See "GUI Checkpoints and Output Values Per Add-in" on page 512 See "Accessing Internal Properties and Methods of Run-Time .NET Web Forms Objects" on page 112.
Troubleshooting and Limitations	See " Troubleshooting and Limitations - .NET Web Forms " on page 112.

Prerequisites	
Opening Your Application	You must open UFT 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.

Configuration	
Options Dialog Box	Use the Web pane. (Make sure that a GUI test is open and select Tools > Options > GUI Testing tab > Web > General node.) See " Web > General Pane (Options Dialog Box) " on page 58.
Record and Run Settings Dialog Box (tests only)	Use the Web tab. (Record > Record and Run Settings) See " Web Tab (Record and Run Settings Dialog Box) " on page 71
Test Settings Dialog Box	Use the Web pane.

(tests only)	(File > Settings > Web node) See " Web Pane (Test/Business Component Settings Dialog Box / Application Area - Additional Settings Pane) " on page 68.
Custom Active Screen Capture Settings Dialog Box (tests only)	Use the Web section. (Tools > Options > GUI Testing tab > Active Screen node > Custom Level) See the section on the Custom Active Screen Capture Settings dialog box in the <i>HP Unified Functional Testing User Guide</i> .
Application Area Additional Settings pane (business components only)	Use the Web pane. In the application area, select Additional Settings > Web in the sidebar. See " Web Pane (Test/Business Component Settings Dialog Box / Application Area - Additional Settings Pane) " on page 68.

Considerations for Testing .NET Web Forms

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

- When UFT learns .NET Web Forms objects, it does not learn the HTML elements that comprise the test objects. For example, when UFT 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 UFT behaves when you record on other non-.NET Web Forms Web objects. For details, see "[Event Recording Configuration for Web-Based Applications](#)" on page 39.

.NET Windows Forms Objects - Checkpoints and Output Values

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

For details, see:

- "[Checking .NET Windows Forms Tables and Outputting Their Values](#)" below
- "[Accessing Internal Properties and Methods of Run-Time .NET Windows Forms Objects](#)" on the next page

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/Modify 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).
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 details, see the sections on checkpoints and output values in the *HP Unified Functional Testing User Guide*.

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 business 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 details on the **Object** property and for details on .NET Windows Forms test objects, methods, and properties see the **.NET Windows Forms** section of the *HP UFT Object Model Reference for GUI Testing*.

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 details on the **Object** property and for details on .NET Web Forms test objects, methods, and properties see the **.NET Web Forms** section of the *HP UFT Object Model Reference for GUI Testing*.

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. For example, change:

```
WbfTreeView.Select "item1;item2;item3;"
```

to:

```
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 (**InfragisticsUltraWebGrid**) 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"
```

- Note that Visual Basic .NET applications are supported by the UFT .NET Add-in.

Creating, Editing, and Running Testing Documents

- UFT 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 UFT 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: <UFT installation folder>\dat\WebFormsConfiguration.xml.

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

- .NET Web Forms objects are supported only on Microsoft Internet Explorer.
- When recording, you must open UFT and define record options before opening your .NET Web Forms application.

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 Editor 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 Editor 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, UFT retrieves the rows or columns only for the outermost table. Note that the **`rows`** property and **`RowCount`** method count only the non-grouping rows.

.NET Windows Forms Add-in

This section includes:

- [.NET Windows Forms Support - Quick Reference](#) 116
- [.NET Windows Forms Support - Testing and Configuration](#)118
 - [Considerations for Testing .NET Windows Forms Applications](#) 119
 - [.NET Windows Forms Objects - Checkpoints and Output Values](#)119
 - [.NET Add-in Extensibility](#) 120
 - [.NET Windows Forms Spy](#) 121
 - [Troubleshooting and Limitations - .NET Windows Forms](#) 129

.NET Windows Forms Support - Quick Reference

You can use the UFT .NET Add-in to test .NET Windows Forms user-interface objects (controls).

The following tables summarize basic information about .NET Windows Forms application support and how it relates to some commonly-used aspects of UFT.

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 " Windows-Based Application Support " on page 81.
Supported Environments	For details on supported .NET Windows Forms environments, see the .NET Add-in section of the <i>HP Unified Functional Testing Product Availability Matrix</i> , available from the HP Support Matrix page (requires an HP passport).
Important Information	<ul style="list-style-type: none"> 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, Visual Basic .NET, C#, and so on) See "Considerations for Testing .NET Windows Forms Applications"
Test Object Methods and Properties	The .NET Add-in provides test objects, methods, and properties that can be used when testing objects in .NET Windows Forms applications. For details, see the .NET Windows Forms section of the <i>HP UFT Object Model Reference for GUI Testing</i> .
Checkpoints and Output Values	<ul style="list-style-type: none"> See the sections describing checkpoints and output values in the <i>HP Unified Functional Testing User Guide</i>. See "Considerations for Testing .NET Windows Forms Applications" on page 119. See "GUI Checkpoints and Output Values Per Add-in" on page 512
Extending the .NET Add-in	" .NET Add-in Extensibility " (described on page 120 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 UFT .NET Add-in.
Troubleshooting and Limitations	See " Troubleshooting and Limitations - .NET Windows Forms " on page 129.

Prerequisites

Opening Your Application	You must open UFT before opening your .NET Windows Forms application
Add-in Dependencies	The .NET Add-in must be installed.

Configuration

Options Dialog Box	Use the Windows Applications pane. (Select Tools > Options > GUI Testing tab > Windows Applications node). See " Windows Applications > General Pane (Options Dialog Box > GUI Testing Tab) " on page 85.
---------------------------	--

<p>Record and Run Settings Dialog Box (tests only)</p>	<p>Use the Windows Applications tab. (Record > Record and Run Settings)</p> <p>See "Windows Applications Tab (Record and Run Settings Dialog Box)" on page 93.</p> <p>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>
<p>Custom Active Screen Capture Settings Dialog Box (tests only)</p>	<p>Use the Windows applications section. (Tools > Options > GUI Testing tab > Active Screen node > Custom Level)</p> <p>See the section on the Custom Active Screen Capture Settings dialog box in the <i>HP Unified Functional Testing User Guide</i>.</p>
<p>Application Area Additional Settings pane (business components only)</p>	<p>Use the Applications pane. In the application area, select Additional Settings > Applications in the sidebar.</p> <p>See the section on the Applications pane in the <i>HP Unified Functional Testing User Guide</i>.</p>

.NET Windows Forms Support - Testing and Configuration

This section includes

- [Considerations for Testing .NET Windows Forms Applications](#) 119
- [.NET Windows Forms Objects - Checkpoints and Output Values](#) 119
- [.NET Add-in Extensibility](#) 120
- [.NET Windows Forms Spy](#) 121
 - [How to Use the .NET Windows Forms Spy](#) 122
 - [.NET Windows Forms Spy Dialog Box](#) 126
- [Troubleshooting and Limitations - .NET Windows Forms](#) 129

Considerations for Testing .NET Windows Forms Applications

- You can use the Keyword View and Editor 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, UFT stores the selected property values of the object. If your application changes, you can modify the captured values to match the new expected values.

.NET Windows Forms Objects - Checkpoints and Output Values

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

For details, see:

- ["Checking .NET Windows Forms Tables and Outputting Their Values" below](#)
- ["Accessing Internal Properties and Methods of Run-Time .NET Windows Forms Objects" on the next page](#)

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/Modify 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).
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 details, see the sections on checkpoints and output values in the *HP Unified Functional Testing User Guide*.

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 business 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 details on the **Object** property and for details on .NET Windows Forms test objects, methods, and properties see the **.NET Windows Forms** section of the *HP UFT Object Model Reference for GUI Testing*.

.NET Add-in Extensibility

UFT .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 UFT .NET Add-in.

If the test object class that UFT 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 UFT 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 UFT 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:

- UFT 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 UFT 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 UFT Extensibility Documentation program group (**Start > All Programs > HP Software > HP Unified Functional Testing > Extensibility > Documentation** or the <UFT installation folder>\help\Extensibility folder).

Note: For details on accessing UFT and UFT tools and files in Windows 8 and Windows Server 2012, see ["Accessing UFT in Windows 8.X or Higher Operating Systems" on page 517](#).

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

.NET Windows Forms Spy

The .NET Windows Forms Spy (described in [".NET Windows Forms Spy Dialog Box" on page 126](#)) 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 \(Record and Run Settings Dialog Box\)](#)" (described on page 93) 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 details on defining trust levels for .NET applications, see Microsoft documentation.

The .NET Windows Forms Spy is intended for advanced UFT 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 UFT 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 UFT. Since the .NET Windows Forms Spy runs in the context of your .NET application, you can close UFT while you use the .NET Windows Forms Spy. However, UFT 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 UFT .NET Windows Forms Spy window is closed automatically.

How to Use the .NET Windows Forms Spy

This task describes elements of the .NET Windows Forms Spy including how to:

Spy on an object

1. Make sure that the application on which you want to spy is specified in the "[Windows Applications Tab \(Record and Run Settings Dialog Box\)](#)" (described on page 93), 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**. The UFT "[.NET Windows Forms Spy Dialog Box](#)" (described on page 126) opens.

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

For details on using the pointing hand, see the section describing the pointing hand in the *HP Unified Functional Testing 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 UFT .NET

Windows Forms Spy window opens showing the properties and values for the selected object.

7. You can repeat these steps to spy on additional objects and add them to the Objects pane in the UFT .NET Windows Forms Spy window.

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**.

View values of 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.


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.

View properties of embedded objects

1. In the Properties tab, select the property whose embedded object properties you want to view. For details on locating a property by value, see "[Locate a property by its value](#)" below.
2. Click the **Add selected property to the Objects tree** 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 to the Objects tree** button is disabled if the property's value is null, or the property is an object with no properties of its own.

Locate a property by its value

1. Click the **Search a 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.

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.

Modify values of run-time object properties

- 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.
- 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.



View event arguments on an object



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

Listen to specified events of an object

- In the Objects pane, select the object to whose events you want to listen.
- 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 UFT. 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 . UFT 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 . UFT stops listening to and logging the specified events.


Fire selected events on an 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.

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.

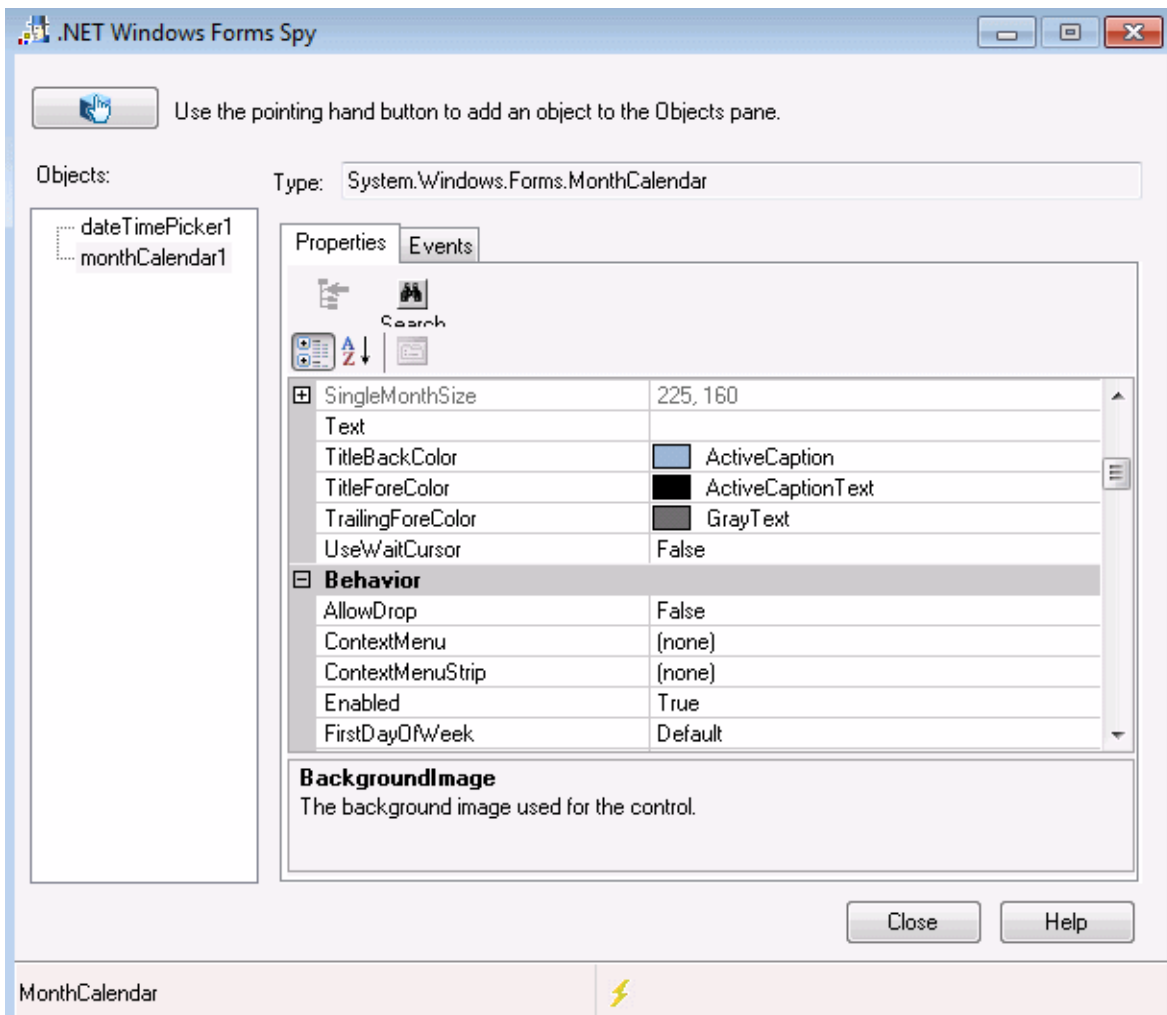
Clear all events from the Fired Events list

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.

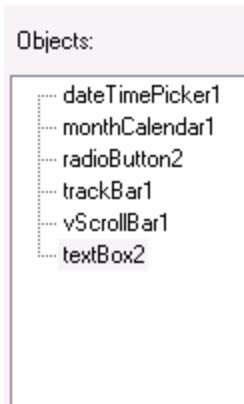
.NET Windows Forms Spy Dialog Box

This dialog box 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.

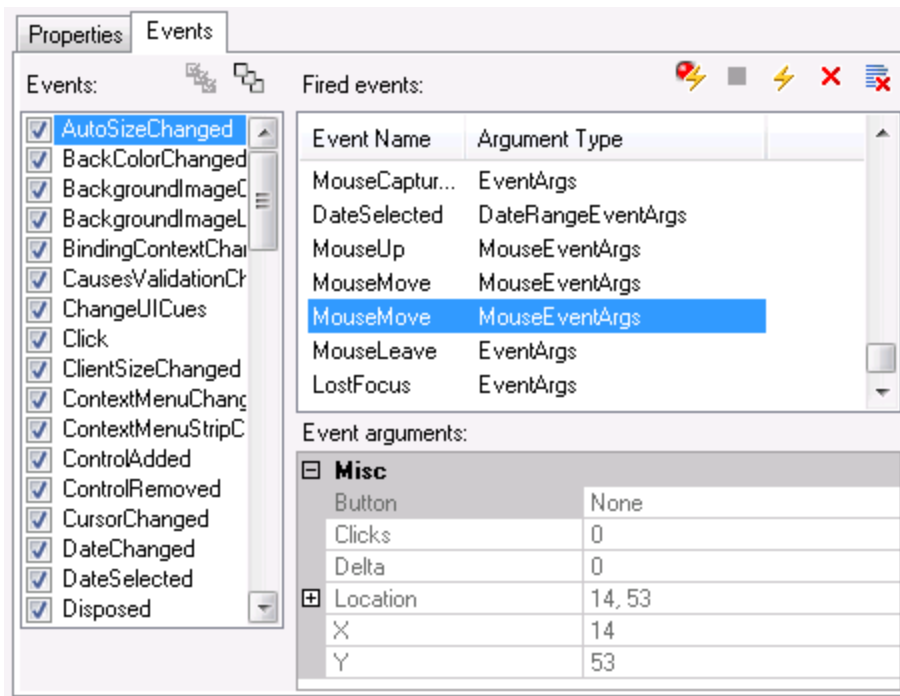
The following illustrates an example of the Form Spy dialog box.



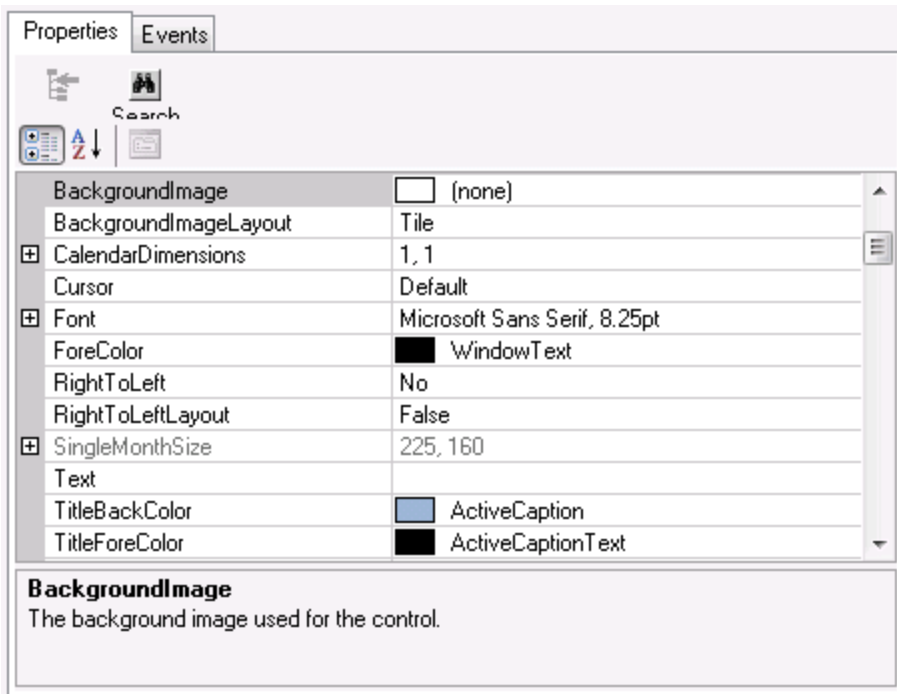
The following illustrates an example of the Form Spy Object Pane.



The following illustrates an example of the Form Spy Events tab.




The following illustrates an example of the Form Spy Properties tab.



To access	Select Tools > .NET Windows Forms Spy .
Relevant tasks	<ul style="list-style-type: none"> • "How to Use the .NET Windows Forms Spy" on page 122
See also	<ul style="list-style-type: none"> • ".NET Windows Forms Objects - Checkpoints and Output Values" on page 119. • ".NET Add-in Extensibility" on page 120. • ".NET Windows Forms Spy" on page 121.

User interface elements are described below:

UI Elements	Description
	<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:</p> <ul style="list-style-type: none"> • If you select an object from a different .NET application, an additional UFT .NET Windows Forms Spy window opens, showing the information for the selected object. • UFT must be open if you want to use the pointing hand mechanism to spy on additional objects.
Type	Displays the full type name of the selected object.
Objects pane	<p>Displays a hierarchical tree of the objects you selected to spy.</p> <p>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 UFT .NET Windows</p>

UI Elements	Description
	<p>Forms Spy window.</p> <p>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. 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. For details, see ".NET Windows Forms Spy" on page 121.</p>
Properties tab	<p>Enables you to view and modify values of run-time object properties in your .NET application. 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.</p> <p>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. For details, see ".NET Windows Forms Spy" on page 121.</p>
Events tab	<p>Enables you to listen to events in your .NET application and fire them at the application.</p> <p>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.</p> <p>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.</p> <p>For details, see ".NET Windows Forms Spy" on page 121.</p>
Status bar	<p>Displays the class name of the object that is selected in the Objects pane, and the event handling status.</p>

Troubleshooting and Limitations – .NET Windows Forms

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

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

objects.

- 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 `regexwndclass` 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 **regexwndclass** 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.*`
- Note that Visual Basic .NET applications are supported by the UFT .NET Add-in.

.NET Windows Presentation Foundation (WPF) Add-in

This section includes:

- [.NET Windows Presentation Foundation \(WPF\) Add-in - Quick Reference](#) 132
- [.NET Windows Presentation Foundation \(WPF\) Add-in - Testing and Configuration](#)134
 - [Considerations for Working with the WPF Add-in](#) 135
 - [WPF Objects, Methods, and Properties to Enhance Your Test or Component](#) 135
 - [About WPF User Interface Automation](#)136
 - [WPF Add-in Extensibility](#) 137
 - [Troubleshooting and Limitations - .NET WPF](#) 137

.NET Windows Presentation Foundation (WPF) Add-in - Quick Reference

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

The following tables summarize basic information about the Windows Presentation Foundation Add-in and how it relates to some commonly-used aspects of UFT.

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 " Windows-Based Application Support " on page 81.
Supported Environments	For details on supported Windows Presentation Foundation environments, see the WPF Add-in section of the <i>HP Unified Functional Testing Product Availability Matrix</i> , available from the HP Support Matrix page (requires an HP passport). Note: See also Child Add-ins, below.
Test Object Methods and Properties	The WPF Add-in provides test objects, methods, and properties that can be used when testing objects in WPF applications. For details, see the .NET Windows Presentation Foundation section of the <i>HP UFT Object Model Reference for GUI Testing</i> .
Checkpoints and Output Values	<ul style="list-style-type: none"> See the sections describing checkpoints and output values in the <i>HP Unified Functional Testing User Guide</i>. To check properties that are not included in the Checkpoint Properties dialog box you can use the <code>Object</code>, <code>AutomationElement</code>, or <code>AutomationPattern</code> property. For details, see "Accessing Internal Properties and Methods of WPF Objects" on page 135. See "GUI Checkpoints and Output Values Per Add-in" on page 512
Extending the WPF Add-in	" WPF Add-in Extensibility " (described on page 137) enables you to develop support for testing third-party and custom WPF controls that are not supported out-of-the-box by the UFT WPF Add-in. For details, see " WPF Add-in Extensibility " on page 137
Troubleshooting and Limitations	See " Troubleshooting and Limitations - .NET WPF " on page 137.
Prerequisites	
Opening Your Application	You can open your WPF application before or after opening UFT.
Add-in Dependencies	The Web and .NET Add-ins must be installed.
Configuration	
Options Dialog Box	Use the Windows Applications pane.

	<p>(Tools > Options > GUI Testing tab > Windows Applications node)</p> <p>See "Windows Applications > General Pane (Options Dialog Box > GUI Testing Tab)" on page 85.</p>
<p>Record and Run Settings Dialog Box (tests only)</p>	<p>Use the Windows Applications tab. (Record > Record and Run Settings)</p> <p>See "Windows Applications Tab (Record and Run Settings Dialog Box)" on page 93.</p>
<p>Custom Active Screen Capture Settings Dialog Box (tests only)</p>	<p>Use the Windows applications section. (Tools > Options > GUI Testing tab > Active Screen node > Custom Level)</p> <p>See the section on the Custom Active Screen Capture Settings dialog box in the <i>HP Unified Functional Testing User Guide</i>.</p>
<p>Application Area Additional Settings pane (business components only)</p>	<p>Use the Applications pane.</p> <p>In the application area, select Additional Settings > Applications in the sidebar.</p> <p>See the section on the Applications pane in the <i>HP Unified Functional Testing User Guide</i>.</p>

.NET Windows Presentation Foundation (WPF) Add-in - Testing and Configuration

This section includes

- [Considerations for Working with the WPF Add-in](#) 135
- [WPF Objects, Methods, and Properties to Enhance Your Test or Component](#) 135
- [About WPF User Interface Automation](#) 136
 - [Automation Elements](#) 136
 - [Control Patterns](#) 136
- [WPF Add-in Extensibility](#) 137
- [Troubleshooting and Limitations - .NET WPF](#) 137

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, Visual Basic, .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.
- You can use the Keyword View and Editor to activate WPF test objects, Automation objects and run-time object methods, retrieve and set the values of properties, and check that objects exist.

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

A test or business component consists of statements coded in Microsoft VBScript. These statements are composed of objects, methods, and/or properties that instruct UFT 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 business component. You create, view, and edit these statements in the Keyword View and/or Editor.

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 details on the methods and properties that are accessible through the **AutomationPattern** property, see the [.NET Framework Developer Center of the Microsoft Developer Network library](http://msdn2.microsoft.com/en-us/library/system.windows.automation.aspx) 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.

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.

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 AutomationPattern properties.

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>.

This section also includes:

- [Automation Elements](#) 136
- [Control Patterns](#) 136

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** property **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.

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

For details on how to work with UI Automation in your test or business component, see "[Accessing Internal Properties and Methods of WPF Objects](#)" on page 135.

WPF Add-in Extensibility

UFT 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 UFT WPF Add-in.

If the test object class that UFT 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 UFT 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:

- UFT 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 UFT setup program.

Troubleshooting and Limitations - .NET WPF

- The UFT .NET Add-in supports:
 - Testing standard .NET Windows Forms controls from the **System.Windows.Forms** library.
 - Testing third-party .NET Windows Forms controls that are inherited from **System.Windows.Forms.Control** class.
- The UFT .NET Add-in supports testing 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 forth), as well as third-party WPF controls that are inherited from the **System.Windows.Controls.Control** class and implement automation interfaces when the WPF Add-in is loaded.
- In addition to the controls listed in the *HP Unified Functional Testing Product Availability Matrix*, custom-built UFT support for a variety of Infragistics .NET Windows Forms controls is provided out-of-the-box by Infragistics TestAdvantage. For more information, refer to:

<http://www.infragistics.com/dotnet/testadvantageoverview.aspx#Overview>

- UFT does not treat text block elements as children of WPF objects, such as lists, treeviews, and tables. Therefore, they will not be returned in a **ChildObjects** statement, nor will they be learned as test objects when you select to learn a WPF object and its children.

If you want to work with the text block elements of a WPF object, use a **GetItem()** or **GetProperty()** statement.

- 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, UFT recognizes the object as a standard Windows object.

Workaround: Close your WPF application. In UFT, open the Record and Run Settings dialog box (**Record > 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 you spy on a WpfComboBox control on a Microsoft Windows 7 operating system, to enable displaying the correct **all items** property value, you must first manually expand and collapse the combo box.
- To view the full type name of a .NET Windows Forms object in your application, view the **SwfTypeName** identification property in the Object Spy.

You can also view a list of the base types of a selected object by running a statement using the following syntax:

```
MsgBox <SwfTestObj>(<descr>).GetROProperty("SwfTypeNames")
```

where `SwfTestObj(<descr>)` is the test object you want to check. Running this statement causes a message box to open displaying the actual class at the top of the list and the base classes below it.

- 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.
- When recording dynamically changing objects in your application, UFT records the object properties of the object after the change instead of on the original object before the change. This causes run sessions using this object to fail.

Workaround: Manually change the object description in the editor.

- UFT does not treat text block elements as child objects of WPF objects, such as lists, treeviews, and tables. Therefore, they will not be returned in a **ChildObjects** statement, nor will they be learned as test objects when you select to learn a WPF object and its children.

If you want to work with the text block elements of a WPF object, use a **GetItem()** or **GetProperty()** statement.

Note: The above behavior was implemented after QTP 11.00. Therefore, if you have **ChildObject** steps in existing tests that previously returned these text block elements, those tests may fail or produce unexpected results. In this case, update such steps as recommended above.

Part 3: ActiveX Add-in

This section includes:

["ActiveX Add-in - Quick Reference" on page 140](#)

["Considerations for Working with the ActiveX Add-in" on page 141](#)

["Troubleshooting and Limitations - ActiveX Add-in" on page 142](#)

ActiveX Add-in - Quick Reference

You can use the UFT ActiveX Add-in to test ActiveX user-interface objects (controls).

The following tables summarize basic information about the ActiveX Add-in and how it relates to some commonly-used aspects of UFT.

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 "Windows-Based Application Support" on page 81.
Supported Environments	For details on supported ActiveX environments, see the ActiveX Add-in section of the <i>HP Unified Functional Testing Product Availability Matrix</i> , available from the HP Support Matrix page (requires an HP passport).
Important Information	See "Considerations for Working with the ActiveX Add-in" on the next page.
Test Object Methods and Properties	The ActiveX Add-in provides test objects, methods, and properties that can be used when testing ActiveX objects in applications. For details, see the ActiveX section of the <i>HP UFT Object Model Reference for GUI Testing</i> .
Checkpoints and Output Values	<ul style="list-style-type: none"> See the sections describing checkpoints and output values in the <i>HP Unified Functional Testing User Guide</i>. See "GUI Checkpoints and Output Values Per Add-in" on page 512
Troubleshooting and Limitations	See "Troubleshooting and Limitations - ActiveX Add-in" on page 142.

Prerequisites	
Opening Your Application	The application containing the ActiveX controls on which you want to record must be closed before you begin a UFT recording session and set the Record and Run options. Open the application only after you begin the recording session.
Add-in Dependencies	Loading the ActiveX and Siebel add-ins together may cause problems when recording on some ActiveX methods.

Configuration	
Options Dialog Box	Use the Windows Applications pane. (Tools > Options > GUI Testing tab > Windows Applications node) See "Windows Applications > General Pane (Options Dialog Box > GUI Testing Tab)" on page 85.
Record and Run Settings Dialog Box (tests only)	Use the Windows Applications tab. (Run > Run Settings or Record > Record Settings)

	<p>See "Windows Applications Tab (Record and Run Settings Dialog Box)" on page 93.</p> <p>Note:</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. UFT 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 > GUI Testing tab > Active Screen node > Custom Level)</p> <p>See the section on the Custom Active Screen Capture Settings dialog box in the <i>HP Unified Functional Testing User Guide</i>.</p>
<p>Application Area Additional Settings pane (business components only)</p>	<p>Use the Applications pane.</p> <p>In the application area, select Additional Settings > Applications in the sidebar.</p> <p>See the section on the Applications pane in the <i>HP Unified Functional Testing User Guide</i>.</p>

Considerations for Working with the ActiveX Add-in

- When you create a checkpoint on an ActiveX control, UFT 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`.
- UFT can record on standard controls within an ActiveX control and if an ActiveX control contains another ActiveX control, then UFT 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, UFT records this step in the Editor 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 details, see the section on using programmatic descriptions in the *HP Unified Functional Testing 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.

Troubleshooting and Limitations - ActiveX Add-in

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

Creating, Editing, and Running Testing Documents

- The ActiveX Add-in is not supported on Internet Explorer 11 when the Enhanced Protected Mode is turned on.
- 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 Editor for a Web application that has a mixed hierarchy of Java objects inside an ActiveX control, then it may take a long time for UFT 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 UFT 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 UFT, 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.
- If an ActiveX control is registered after UFT was started, UFT may not recognize the controls correctly. For example, UFT may recognize an **AcxCalendar** as a **ActiveX** object.
Workaround: Install the application running the ActiveX controls and register all ActiveX objects before starting UFT.

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 (**Run > 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.

Unsupported Controls

UFT does not support certain ActiveX controls or controls with certain prefixes. These controls are listed in the table below.

Unsupported Control Prefixes	Unsupported Controls
Msawt	AMOVIE.ActiveMovieControl.2
SpectrumHR.GrabBag	MediaPlayer.MediaPlayer.1
SpectrumHR.EDataControl	Trident.HTMLEditor.1
SpectrumHR.SSDBGGridEventHandler	htmlfile
ShockwaveFlash	xmlfile
Spider90	htmlfile_FullWindowEmbed

Unsupported Control Prefixes	Unsupported Controls
XGO	xmlfile_FullWindowEmbed
	Inkfile
	JScript
	VBScript
	MSJava
	PDF.PdfCtrl.1
	ScriptBridge.ScriptBridge.1
	JavaSoft.JavaBeansBridge.1
	Oracle.JavaBeansBridge.1
	Spider.Loader.1
	COMCTL.ImageListCtrl.1
	ActiveTabs.SSTabPanel.4
	ActiveTabs.SSTabPanel.2
	ActiveTabs.SSTabPanel.3
	{3050f67D-98b5-11cf-bb82-00aa00bdce0b}
	{3050F5C8-98B5-11CF-BB82-00AA00BDCE0B}
	TriEditDocument.TriEditDocument.1
	Miner3D.Miner3DObj.1
	ActiveBar2Library.ActiveBar2.2
	{275C23E2-3747-11D0-9FEA-00AA003F8646}
	SpectrumHR.GrabBag.1
	SpectrumHR.EDataControl.1
	SpectrumHR.SSDBGridEventHandler.1

Part 4: Delphi Add-in

This section includes:

["Delphi Add-in - Quick Reference" on page 146](#)

["Delphi Add-in Extensibility" on page 147](#)

["How to Enable Communications Between UFT and Your Delphi Application" on page 148](#)

["Troubleshooting and Limitations - Delphi Add-in" on page 149](#)

Delphi Add-in - Quick Reference

You can use the UFT Delphi Add-in to test Delphi user-interface objects (controls).

The following tables summarize basic information about the Delphi Add-in and how it relates to some commonly-used aspects of UFT.

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 "Windows-Based Application Support" on page 81 .
Supported Environments	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 <i>HP Unified Functional Testing Product Availability Matrix</i> , available from the HP Support Matrix page (requires an HP passport).
Test Object Methods and Properties	The Delphi Add-in provides test objects, methods, and properties that can be used when testing objects in Delphi applications. For details, see the Delphi section of the <i>HP UFT Object Model Reference for GUI Testing</i> .
Checkpoints and Output Values	<ul style="list-style-type: none"> See the sections describing checkpoints and output values in the <i>HP Unified Functional Testing User Guide</i>. See "GUI Checkpoints and Output Values Per Add-in" on page 512
Extending the Delphi Add-in	"Delphi Add-in Extensibility" (described on page 147) enables you to develop support for testing third-party and custom Delphi controls that are not supported out-of-the-box by the UFT Delphi Add-in.

Prerequisites	
Opening Your Application	You can open your Delphi application before or after opening UFT.
Add-in Dependencies	None
Other	Before running a test on a Delphi application, the application being tested must be compiled with the UFT agent MicDelphiAgent . See "How to Enable Communications Between UFT and Your Delphi Application" on page 148 .

Configuration	
Options Dialog Box	Use the Windows Applications pane. (Tools > Options > GUI Testing tab > Windows Applications node) See "Windows Applications > General Pane (Options Dialog Box > GUI Testing Tab)" on page 85 .
Record and Run	Use the Windows Applications tab.

<p>Settings Dialog Box (tests only)</p>	<p>(Record > Record and Run Settings)</p> <p>See "Windows Applications Tab (Record and Run Settings Dialog Box)" on page 93.</p> <div style="background-color: #f0f0f0; padding: 10px;"> <p>Note:</p> <ul style="list-style-type: none"> • UFT recognizes only Delphi applications that have been precompiled with the MicDelphiAgent.pas module. For details, see "How to Enable Communications Between UFT and Your Delphi Application" on the next page. • 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. </div>
<p>Custom Active Screen Capture Settings Dialog Box (tests only)</p>	<p>Use the Windows section. (Tools > Options > GUI Testing tab > Active Screen pane > Custom Level button)</p> <p>See the section on the Custom Active Screen Capture Settings dialog box in the <i>HP Unified Functional Testing User Guide</i>.</p>
<p>Application Area Additional Settings pane (business components only)</p>	<p>Use the Applications pane. In the application area, select Additional Settings > Applications in the sidebar.</p> <p>See the section on the Applications pane in the <i>HP Unified Functional Testing User Guide</i>.</p>

Delphi Add-in Extensibility

UFT 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 UFT Delphi Add-in.

If the test object class that UFT 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 UFT 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:

- UFT 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.

UFT 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 UFT Extensibility Documentation program group (**Start > All Programs > HP Software > HP Unified Functional Testing > Extensibility > Documentation** or the <UFT installation folder>\help\Extensibility folder).

Note: For details on accessing UFT and UFT tools and files in Windows 8 and Windows Server 2012, see ["Accessing UFT in Windows 8.X or Higher Operating Systems" on page 517](#).

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

How to Enable Communications Between UFT and Your Delphi Application

This task describes how to:

- Use the `MicDelphiAgent.pas` module to enable communications between UFT and each Delphi project you want to test.
- Configure support for the **TwwDBGGrid** from InfoPower.

[Link to the MicDelphiAgent.pas Module to Enable Communications](#)

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

1. Add the <UFT Installation folder>\dat\Extensibility\Delphi folder to your Delphi project search path, or copy the contents of the <UFT 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..

Configure Support for TwwDBGrid

If your application includes the **TwwDBGrid** from InfoPower, do the following 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.

Troubleshooting and Limitations - Delphi Add-in

Button controls in message boxes are identified as **WinButton** objects instead of **DelphiButton** objects.

Workaround: Replace the button control test objects in the object repository with **DelphiButton** objects.

Part 5: Flex Add-in

This section includes:

["Flex Add-in - Quick Reference" on page 151](#)

["Flex Add-in - Testing and Configuration" on page 152](#)

Flex Add-in - Quick Reference

You can use the UFT Flex Add-in to test Flex user-interface objects (controls).

The following tables summarize basic information about the Flex Add-in and how it relates to some commonly-used aspects of UFT.

General Information	
Supported Environments	Tested applications must be built with Flex SDK versions that are supported by the UFT Flex Add-in. For details on supported Flex SDK versions, see the Flex section of the <i>HP Unified Functional Testing Product Availability Matrix</i> , available from the HP Support Matrix page (requires an HP passport).
Important Information	<ul style="list-style-type: none"> You can use UFT with Flex applications that satisfy one of the following conditions: <ul style="list-style-type: none"> Applications opened with Adobe Flash Player Debugger Applications opened using the UFT Runtime Loader Applications prepared manually for testing. <p>Preparing the application consists of embedding the application in a Web page together with the Runtime Loader, or recompiling the application with the relevant Adobe or Apache Flex automation libraries and a UFT Flex pre-compiled agent.</p> <p>For details, see:</p> <ul style="list-style-type: none"> "Considerations for Working with the Flex Add-in" on page 155 "How to Set Up the Adobe Flash Player Debugger to Enable UFT GUI Testing" on page 156 "How to Open Flex Applications Using the Runtime Loader" on page 158 "How to Compile Flex Applications for UFT Testing" on page 162
Test Object Methods and Properties	The Flex Add-in provides Flex test objects, methods, and properties that can be used when testing Flex objects in Flex applications. For details, see the Flex section of the <i>HP UFT Object Model Reference for GUI Testing</i> .
Checkpoints and Output Values	<ul style="list-style-type: none"> See the sections describing checkpoints and output values in the <i>HP Unified Functional Testing User Guide</i> See "GUI Checkpoints and Output Values Per Add-in" on page 512
Troubleshooting and Limitations	"Troubleshooting and Limitations - Flex Add-in " on page 171
Prerequisites	
Opening Your Application	You can open your Flex application before or after opening UFT.
Add-in Dependencies	<ul style="list-style-type: none"> Different versions of the Flex SDK require different versions of Adobe Flash Player, Adobe Flash Player Debugger, or Adobe Air. <p>The Flex Add-in requires the versions of Adobe Flash Player / Debugger or Adobe Air that are required by the version of the Flex SDK used to build the application being tested. For more details, see the Adobe Flex SDK or Apache Flex SDK documentation.</p>

	<ul style="list-style-type: none"> For Flex applications that you recompile with UFT's pre-compiled agent: If the Flex applications were compiled using Adobe Flex SDK versions 4.5.x or 4.6.x, verify that you have licensed versions of the relevant Adobe Automation libraries before running full UFT GUI tests and components. If you do not have licensed versions of the libraries, consider upgrading to Apache Flex SDK version 4.9.x or 4.12.x. UFT interacts with the Flex application it is testing via a local TCP socket object, selecting an available communication port in the range 24654 - 24663. Make sure that at least one of these ports is available on the UFT computer. If no ports in this range are available, the add-in fails to load properly. On a Windows server, multiple users can run multiple instances of UFT. To test Flex applications, you must have one port in this range available for each UFT instance.
--	---

Configuration	
Record and Run Settings Dialog Box (tests only)	Use the Flex tab. (Record > Record and Run Settings) See "Flex Tab (Record and Run Settings Dialog Box)" on page 166.

Flex Add-in - Testing and Configuration

This chapter includes:

- [Enabling UFT to Identify Objects in your Flex Application](#) 152
- [Considerations for Working with the Flex Add-in](#) 155
- [How to Set Up the Adobe Flash Player Debugger to Enable UFT GUI Testing](#) 156
- [How to Open Flex Applications Using the Runtime Loader](#) 158
- [How to Embed a Flex Application in a Web Page with the Runtime Loader](#) 160
- [How to Compile Flex Applications for UFT Testing](#) 162
- [How to Work With Embedded Objects in Flex Lists, Tables, or Tree-Views](#) 164
- [Flex Tab \(Record and Run Settings Dialog Box\)](#) 166
 - [Add/Edit Flex Application Dialog Box](#) 168
- [Troubleshooting and Limitations - Flex Add-in](#) 171

Enabling UFT to Identify Objects in your Flex Application

Some preparation is required to enable UFT to communicate with and identify objects in your Flex application. Specifically, you must do one of the following:

- Install and pre-configure the relevant Flash Player Debugger
- Set up the UFT Runtime Loader on the computer or server where the application runs.
- Recompile your Flex application with the UFT Flex Agent

Depending on the type of Flex application you are testing, you may be able to choose from any of the above, or you may be limited to one or two of the above options.

The table below summarizes these possibilities, and the basic process to follow for each option:

Open Flex application using → Application path ↓	Preconfigured Flash Player Debugger (Configure once per testing computer; Flex application not modified)	Runtime Loader (Configure once per host computer/server; Flex application not modified)	Precompiled Application (Configure once per application; Must recompile Flex application)
*.html/.htm	<ol style="list-style-type: none"> 1. Install and preconfigure the relevant debugger. 2. Open with: Internet Explorer 	* Relevant only for Web pages that embed an .swf application and the UFT Runtime loader <ol style="list-style-type: none"> 1. Make sure the RTL exists on the application host computer/server. 2. Embed the Swf application and RTL in the Web page. 3. Open with: Internet Explorer 	<ol style="list-style-type: none"> 1. Precompile your application with the UFT agent. 2. Open with: Internet Explorer
*.swf	<ol style="list-style-type: none"> 1. Install and preconfigure the relevant debugger. 2. Open with: <ul style="list-style-type: none"> • Internet Explorer • Flash Player Projector debugger 	<ol style="list-style-type: none"> 1. Make sure the RTL exists on the application host computer/server. 2. Open with: <ul style="list-style-type: none"> • Internet Explorer using the relevant command. • Flash Player Projector 	<ol style="list-style-type: none"> 1. Precompile your application with the UFT agent. 2. Open with: <ul style="list-style-type: none"> • Internet Explorer • Flash Player Projector
*.exe	Not Supported	Not Supported	<ol style="list-style-type: none"> 1. Precompile your application with the UFT agent. 2. Open the

Open Flex application using → Application path ↓	Preconfigured Flash Player Debugger (Configure once per testing computer; Flex application not modified)	Runtime Loader (Configure once per host computer/server; Flex application not modified)	Precompiled Application (Configure once per application; Must recompile Flex application)
			application directly.

Considerations for Working with the Flex Add-in

Consider the following when working with the Flex Add-in :

- ["Working with or without the Web Add-in" below](#)
- ["Register local Web-based Flex applications as trusted applications" below](#)

Working with or without the Web Add-in

You can use the Flex Add-in with or without the Web Add-in enabled. The test object hierarchy differs as follows:

- **With the Web Add-in enabled.** Flex test objects have a Web parent hierarchy.

For example:

```
Browser.Page.FlexWindow.FlexButton
```

- **Without the Web Add-in enabled.** Flex objects have a Windows parent hierarchy.

For example:

```
Window.WinObject.FlexWindow.FlexButton
```

Tip: It is recommended to enable the Web Add-in when you are testing Flex applications so that you can test Flex applications in browser windows.

Register local Web-based Flex applications as trusted applications

Local Web-based Flex applications are Flex applications that are stored locally and run in a browser window. UFT does not recognize local Web-based Flex applications as Flex test objects unless all relevant elements are registered as trusted applications:

- In all cases, your local Flex application and its HTML wrapper must be registered.
- If you are opening your local Web applications using a (local) runtime loader file you also need to register the (local) Flex Runtime Loader file.
- If you use the **Open the following applications when a record or run session begins** option in the Flex tab of the Record and Run dialog box and you set a local Web-based Flex application to be opened using a runtime loader, then UFT automatically creates an HTML wrapper for your application and stores the wrapper in your user profile **%temp%** folder (For example, C:\Users\myname\AppData\Local\Temp). In this case, you must additionally register the **%temp%** folder.

To register your local Web-based Flex applications, the HTML-wrappers, the and Runtime Loader file, add the paths of the folders that contain them (along with the **%temp%** folder if relevant) to one of the following:

- If you have an Internet connection, you can use the **Trusted Locations** list in the Flash Player Global Settings:

Use this link to access the Settings pane:

http://www.macromedia.com/support/documentation/en/flashplayer/help/settings_manager04.html and select **Edit locations** > **Add location** to edit the list).

- Otherwise, add or edit a text file located in the FlashPlayerTrust folder in the following location:
%appdata%\Macromedia\Flex Player\#Security\FlexPlayerTrust

Each line in the text file must contain the name of a folder to trust. For each specified folder, all files in that folder or any sub-folders are trusted. For example:

```
# Trust all files in the Employee online calendar application folder
%ProgramFiles%\Personnel\Employees\OnlineCalendar
```

Note:

- The %appdata% folder is hidden in Windows by default. To show hidden folders, open the Windows Explorer **Folder Options** dialog box and select **Show hidden files and folders**.
- Create the #Security\FlexPlayerTrust folder, if it does not exist.
- The UFT Flex Runtime Loader files are installed with UFT, in the **<UFT installation folder>\dat\Flex\Flex\Runtime Loader** folder.

How to Set Up the Adobe Flash Player Debugger to Enable UFT GUI Testing

This task describes how to set up the Adobe Flash Player Debugger on your UFT computer, and configure it for UFT testing.

After you do this, you can run Flex applications using the Adobe Flash Player Debugger (or the Adobe Flash Player Projector Debugger) and test them using UFT like you would any other type of application. You do not need to prepare the application for testing or load it in any special way.

This method can be used for SWF and HTML Flex applications.

Note: If you do not want to use the debugger to run your Flex applications, you can choose an alternative method of enabling UFT to communicate with your Flex application. For details, see ["Enabling UFT to Identify Objects in your Flex Application" on page 152](#).

Make sure that you have the Adobe Flash Player Debugger installed

If you do not have the Adobe Flash Player Debugger, [download](#) and install the program file that is relevant for your operating system and browser from this site:

<https://www.adobe.com/support/flashplayer/downloads.html>.

Tip: Some tips on locating and installing the Flash Player Debugger:

- The names **Flash Player Debugger** and **Flash Player ActiveX control content debugger** are interchangeable.
- To successfully complete the Flash Player Debugger installation on Windows 2012 or 2012 R2, add the **Desktop Experience** feature in the Windows Server Manager before installing the Flash Player Debugger from Adobe's site.

Set up the Adobe Flash Player Debugger to integrate with UFT

In this step you edit the Adobe Flash Player Debugger configuration file and add UFT's Flex Agent to the trusted locations in the Flash Player's global security settings.

- Configure the Flash Player to load the UFT Flex Agent every time it runs an application and to refrain from opening message boxes during the run session (direct the messages to the player's log file instead):
 - a. Create or open the **%USERPROFILE%\MM.CFG** file
 - b. Add the following lines to the file (replace <UFT installation folder> with the relevant path):

```
PreloadSWF=<UFT installation  
folder>\dat\Flex\UFTFlexAgentInjector.swf  
SuppressDebuggerExceptionDialogs=1  
ErrorReportingEnable=1  
TraceOutputFileEnable=1
```

You can use this default file as an example: **<UFT Installation folder>\dat\Flex\MM.CFG**

- (Optional) Add UFT's Flex Agent folder to the trusted locations in the Flash Player's global security settings.
(You need to perform this step only if you find that UFT does not properly interact with your Flex applications.)

If you have an Internet connection:

- a. Open http://www.macromedia.com/support/documentation/en/flashplayer/help/settings_manager04.html

This opens the Flash Player Help, which opens the actual Global Security Settings panel.

- b. Open the **Edit locations** drop-down list and select **Add location**.
- c. In the dialog box that opens, enter <UFT installation folder>\dat\Flex\ in the text box (replace <UFT installation folder> with the relevant path), and click **Confirm**.

Otherwise:

- a. Create or open a text file located in the `FlashPlayerTrust` folder in the following location:
`%appdata%\Macromedia\Flash Player\#Security\FlashPlayerTrust`

Note:

- The `%appdata%` folder is hidden in Windows by default. You may need to show hidden folders to see it.
- Create the `#Security\FlashPlayerTrust` folder, if it does not exist.

- b. Each line in the text file contains the name of a file or folder to trust.

Add the following line (replace `<UFT installation folder>` with the relevant path):

```
<UFT installation folder>\dat\Flash\Flex\
```

How to Open Flex Applications Using the Runtime Loader

This task describes how to open Flex applications for UFT testing in Internet Explorer, using the UFT Flex Runtime Loader.

The UFT Flex Runtime Loader enables you to test Flex applications (**.swf** files) directly without having to prepare the application manually for testing.

This method is supported only when testing **.swf** files directly.

Note:

- If you are testing an **.swf** application that must remain embedded in an HTML file, or other types of Flex applications, you can choose an alternative method of enabling UFT to communicate with your Flex application. For details, see ["Enabling UFT to Identify Objects in your Flex Application" on page 152](#).
- Do not use the Runtime Loader to load applications that you already compiled with the UFT Flex pre-compiled agent.

Prerequisites

The UFT Flex Runtime Loader files are installed with UFT, in the **<UFT installation folder>\dat\Flash\Flex\Runtime Loader** folder.

- Use one of the following Flex Runtime Loaders:
 - **UFTFlexAUTLoader_4_9_1.swf** - for testing Flex applications developed using the Flex SDK 4.9.1 or earlier
 - **UFTFlexAUTLoader_4_12_1.swf** - for testing Flex applications developed using the Flex SDK 4.12.x

- Make sure that the UFT Flex Runtime Loader is located in the same application and security domain as the Flex application you are testing.

If the application you are testing resides on a Web server, you must place a copy of the Runtime Loader on the same Web server and use that copy to open the application.

Note:

If you use tests that were recorded on pre-compiled Flex applications to test Flex applications opened with the Runtime Loader (or vice versa), you may need to modify the object repositories associated with the test and any test scripts that use programmatic descriptions to identify Flex test objects.

The **uid** property value in all Flex test objects and the **id** property value in FlexWindow test objects will differ between applications opened with the Runtime Loader and pre-compiled Flex applications. Before running your test, make sure that test objects whose descriptions includes these properties match the objects found in the application you are testing.

If the application is on the file system, use the Runtime Loader stored in the file system.

- If you use a copy of the Runtime Loader, and not one of the ones stored in the **<UFT installation folder>\dat\Flex\Runtime Loader** folder, make sure to recopy the file after any UFT upgrade to ensure that you use the most recent file version provided with UFT.

Open the Flex Web application using the Runtime Loader

Open the application in a 32-bit Internet Explorer, using the following syntax in the URL box:

```
<UFTFlexAUTLoaderPath.swf>?swf_url=<ApplicationName.swf>&<param_name1>=<param_value1>&<param_name2>=<param_value2>
```

<p><i>UFTFlexAUTLoaderPath</i></p>	<p>The URL or file system path to the UFT Flex Runtime Loader file.</p> <p>If using file system paths, prefix the path with file://.</p> <p>For example: file://C:\...\UFTFlexAUTLoader.swf?swf_url=C:\...\ApplicationName.swf</p> <p>Make sure to use the Runtime Loader file suitable for your application:</p> <ul style="list-style-type: none"> • UFTFlexAUTLoader_4_9_1.swf - for testing Flex applications developed using the Flex SDK 4.9.1 or earlier • UFTFlexAUTLoader_4_12_1.swf - for testing Flex applications developed using the Flex SDK 4.12.x
<p><i>ApplicationName</i></p>	<p>The file name of the Flex application that you want to open.</p> <p>If the application is stored in a different folder than the Runtime Loader, include the URL or file system path to the application.</p> <div style="background-color: #f0f0f0; padding: 5px; margin-top: 10px;"> <p>Tip: A file system path can be a full path or the path relative to the location of the Runtime Loader.</p> </div>

<code>param_names=param_values</code>	(Optional) A list of parameters and their values to pass to the application being opened. Parameters are separated by the ampersand (&) character.
---------------------------------------	--

How to Embed a Flex Application in a Web Page with the Runtime Loader

This task describes how to embed a Flex application in a Web page together with the UFT Flex Runtime Loader. UFT can then test the application when this Web page is opened in Internet Explorer.

This method is useful if you are testing a Flex Web (.swf) application that is not already embedded in an HTML file.

Note: If this method does not fit your needs, you can choose an alternative method of enabling UFT to communicate with your Flex application. For details, see ["Enabling UFT to Identify Objects in your Flex Application" on page 152.](#)

1. Prerequisites

The UFT Flex Runtime Loader files are installed with UFT, in the **<UFT installation folder>\dat\Flex\Flex\Runtime Loader** folder.

- Use one of the following Flex Runtime Loaders:
 - **UFTFlexAUTLoader_4_9_1.swf** - for testing Flex applications developed using the Flex SDK 4.9.1 or earlier
 - **UFTFlexAUTLoader_4_12_1.swf** - for testing Flex applications developed using the Flex SDK 4.12.x
- Make sure that the UFT Flex Runtime Loader is located in the same application and security domain as the Flex application you are testing.

If the application you are testing resides on a Web server, you must place a copy of the Runtime Loader on the same Web server and use that copy to open the application.

Note:

If you use tests that were recorded on pre-compiled Flex applications to test Flex applications opened with the Runtime Loader (or vice versa), you may need to modify the object repositories associated with the test and any test scripts that use programmatic descriptions to identify Flex test objects.

The **uid** property value in all Flex test objects and the **id** property value in FlexWindow test objects will differ between applications opened with the Runtime Loader and pre-compiled Flex applications. Before running your test, make sure that test objects whose descriptions includes these properties match the objects found in the application you are testing.

If the application is on the file system, use the Runtime Loader stored in the file system.

- If you use a copy of the Runtime Loader, and not one of the ones stored in the **<UFT installation folder>\dat\Flex\Runtime Loader** folder, make sure to recopy the file after any UFT upgrade to ensure that you use the most recent file version provided with UFT.

2. Create the Web page

Make a copy of the UFT sample Web page located in **<UFT installation folder>\dat\Flex\Runtime Loader\UFTFlexAUTLoader_Sample.html**. Store this file in the same application and security domain as the UFT Flex Runtime Loader and the Flex application you are testing.

When you test your application using UFT, run the application by opening this file in Internet Explorer.

3. Update the Runtime Loader location specified in the Web page

- a. (Optional) If the Runtime Loader is located in a different folder than the **html** file you created, modify the Runtime Loader file name to include a path.

The path can be a URL (if the Runtime Loader is located on a Web server), a full file system path, or a path relative to the location of the **html** file.

Locate this line to make the change:

```
<param name="movie" value="UFTFlexAUTLoader.swf" />
```

For example:

```
<param name="movie" value="C:\MyApps\FlexRT\UFTFlexAUTLoader.swf" />
```

- b. In both places that the Runtime Loader file name appears in the file as **UFTFlexAUTLoader.swf**, change it to **UFTFlexAUTLoader_4_9_1.swf** or **UFTFlexAUTLoader_4_12_1.swf** according to the version of the Flex SDK used to develop the application you are testing.

4. Embed the Flex application in the Web page

Enter your application file name and, optionally, parameters, in the **swf_url** parameter, in the following lines (2 places):

```
<param name="FlashVars" value="swf_url=YourApplication.swf" />  
<embed id="loader"  
  width="100%" height="100%" align="middle"  
  src="UFTFlexAUTLoader<version number>.swf"  
  flashvars="swf_url=YourApplication.swf"/>
```

Use the following syntax:

```
swf_url=<ApplicationName.swf>&<param_name1>=<param_value1>&<param_name2>=<param_value2>
```

<i>ApplicationName</i>	<p>The file name of the Flex application that you want to open.</p> <p>If the application is stored in a different folder than the Runtime Loader, provide the URL or file system path to the application.</p> <p>For example:</p> <pre>swf_url=http://some_server/MyApp.swf swf_url=C:\\Flex\\AUTs\\MyApp45.swf</pre> <p>Tip: A file system path can be a full path or the path relative to the location of the Runtime Loader.</p>
<i>param_names=param_values</i>	<p>(Optional) A list of parameters and their values to pass to the application being opened. Parameters are separated by the ampersand (&) character.</p> <p>For example:</p> <pre>swf_url=MyApplication.swf&param_name=param_value&param2_name=param2_value</pre>

How to Compile Flex Applications for UFT Testing

This task describes how to compile your Flex applications for UFT testing.

This method can be used for all supported Flex application types.

Note:

- **For all Flex applications except Adobe AIR:** When you compile your Flex application with a UFT Flex pre-compiled agent, you must set the Flash Player target version to 10.0 or later.
- If you do not want to recompile your Flex application, and you are testing an HTML or SWF application, you can choose an alternative method of enabling UFT to communicate with your Flex application. For details, see ["Enabling UFT to Identify Objects in your Flex Application" on page 152.](#)

This section includes:

- ["Prepare a Flex application for Web, when the Flex application is embedded in an HTML file" on the next page](#)
- ["Prepare a Flex application for Adobe AIR for testing" on the next page](#)

- "Prepare a Flex application hosted by the Adobe Flash Player ActiveX control or Adobe Flash Player Projector (including the content debuggers)" below
- "Prepare a Flex application that uses the Flex charting or AdvancedDataGrid classes" on the next page

Prepare a Flex application for Web, when the Flex application is embedded in an HTML file

1. Link the Flex application to Adobe or Apache Flex automation libraries and a UFT Flex pre-compiled agent. To do this, add the following compiler arguments in the Flex project, and then recompile the application:

```
-include-libraries "<PATH_TO_UFT_ROOT>\dat\Flash\Flex\HpQTAgent.swc"  
-include-libraries "${flexlib}\libs\automation\automation_agent.swc"  
-include-libraries "${flexlib}\libs\automation\automation.swc"  
-include-libraries "${flexlib}\libs\automation\automation_spark.swc"
```

2. Embed the Flex application in a host **.html** document.
3. When testing, run your application by opening the host document in a Web browser.

Prepare a Flex application for Adobe AIR for testing

Link the Flex AIR application to Adobe or Apache Flex automation libraries and a UFT Flex pre-compiled agent. To do this, add the following compiler argument in the Flex AIR project, and then recompile the application:

```
-include-libraries "<PATH_TO_UFT_ROOT>\dat\Flash\Flex\HpQTAgent.swc"  
-include-libraries "${flexlib}\libs\automation\automation_agent.swc"  
-include-libraries "${flexlib}\libs\automation\automation.swc"  
-include-libraries "${flexlib}\libs\automation\automation_spark.swc"  
-include-libraries "${flexlib}\libs\automation\automation_air.swc"  
-include-libraries "${flexlib}\libs\automation\automation_airspace.swc"
```

Prepare a Flex application hosted by the Adobe Flash Player ActiveX control or Adobe Flash Player Projector (including the content debuggers)

1. Link the Flex application to Adobe or Apache Flex automation libraries and a UFT Flex pre-compiled agent. To do this, add the following compiler argument in the Flex project, and then recompile the application:

```
-include-libraries "<PATH_TO_UFT_ROOT>\dat\Flash\Flex\HpQTAgent.swc"  
-include-libraries "${flexlib}\libs\automation\automation_agent.swc"  
-include-libraries "${flexlib}\libs\automation\automation.swc"  
-include-libraries "${flexlib}\libs\automation\automation_spark.swc"
```

2. When testing, run your application by opening it in one of the following:
 - an Adobe Flash Player ActiveX control
 - the Adobe Flash Player Projector

Prepare a Flex application that uses the Flex charting or AdvancedDataGrid classes

Link the Flex application to the `automation_dmv.swc` library. To do this, add the following compiler argument in the Flex project and then recompile your application:

```
-include-libraries "${flexlib}\libs\automation\automation_dmv.swc"
```

How to Work With Embedded Objects in Flex Lists, Tables, or Tree-Views

Sometimes, Flex objects are embedded inside of other non-container Flex objects. For example, a Flex table cell or a Flex list item might contain edit boxes, text boxes, check boxes, and so on.

UFT does not identify these objects when using the Object Spy, recording on Flex applications, or learning Flex objects.

However, for Flex objects embedded or contained inside `FlexList`, `FlexTable`, or `FlexTreeView` objects, you can manually add steps to your test or component that retrieve the embedded objects. Once you retrieve these child objects, you can use them as you would other test objects, though they are not stored in the object repository.

This task describes the steps you can use to access and test Flex objects embedded in `FlexList`, `FlexTable`, or `FlexTreeView` objects.

1. Record or learn the containing `FlexTable`, `FlexList`, or `FlexTreeView` objects.
2. (Optional) Activate the containing table cell, list item, or tree-view node, using the **`FlexTable.SelectCell`**, **`FlexList.Select`**, or **`FlexTreeView.Select`** methods.
In some situations, this changes the embedded objects. In others, it is required in order to bring the containing object into view.
3. Retrieve the embedded Flex objects using the **`FlexTable.GetCellChildObjects`**, **`FlexTreeView.GetItemChildObjects`**, or **`FlexList.GetItemChildObjects`** methods.
In these methods, you can provide a *Description* parameter that limits the returned child objects to the ones that match the description.
4. (Optional) Iterate through the collection of returned test objects to check what objects are contained inside the `FlexTable` cell, `FlexList` item, or `FlexTreeView` node to perform operations on the different objects.
5. Add steps to your test or component that perform operations on the embedded objects. For example, add **Set** steps on `FlexSpin`, `FlexEdit`, or `FlexCheckBox` objects.

You can also perform steps on embedded objects without retrieving them, by clicking the relevant location inside the containing object. To do this, for example, use the **FlexTable.SelectRow**, **FlexTreeView.Select** or **FlexTable.SelectCell** method, providing the relevant coordinates within the row, node, or cell.

6. Check the properties of embedded objects. You can do this using the **CheckProperty** or **GetROProperty** methods, or checkpoints.

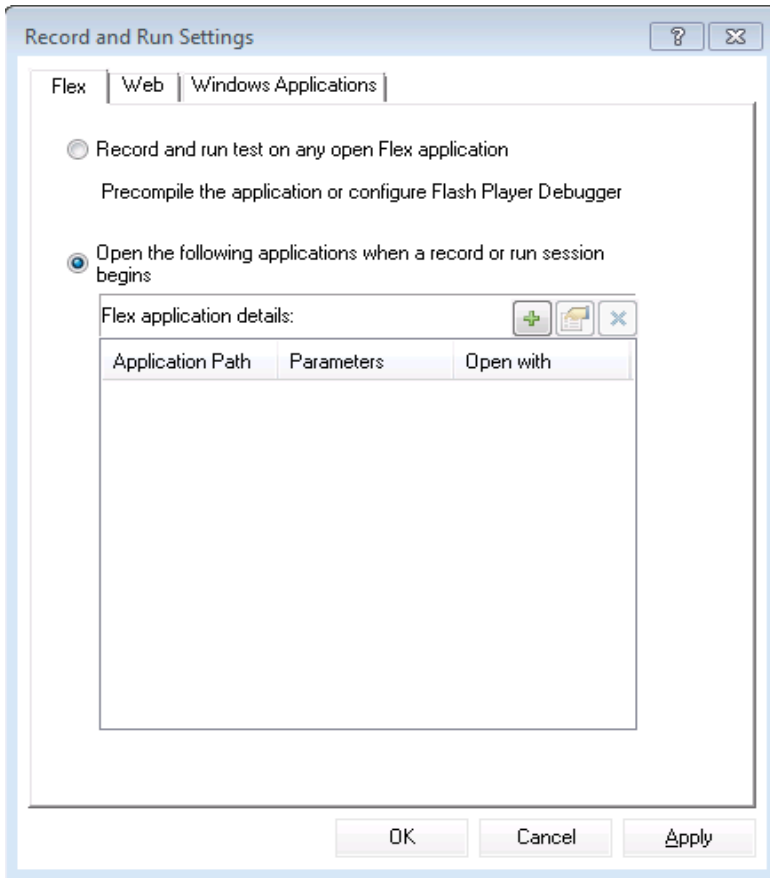
To create a checkpoint to use for an embedded object, create the checkpoint on an object of the same type that is not embedded. The checkpoint is stored in the object repository and you can then use it for the embedded object, as demonstrated in the following example:

```
'Retrieve child objects from Options column in the table's first row
Set child_buttons = grid.GetCellChildObjects(0, "Options")
'iterate through all retrieved options and run a checkpoint on each
For n=0 to (child_buttons.count-1)
    child_buttons(n).Check CheckPoint("Option_radiobutton")
Next
```

For details and examples on using the FlexTable and FlexList methods in your tests, see the **Flex** section of the *HP UFT Object Model Reference for GUI Testing* (**Help > HP Unified Functional Testing Help > Object Model Reference for GUI Testing > Flex**).




Flex Tab (Record and Run Settings Dialog Box)

This tab enables you to define preferences for recording and running tests on Flex applications.



<p>To access</p>	<p>Select Record > Record and Run Settings and then select the Flex tab.</p> <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>
<p>Important information</p>	<ul style="list-style-type: none"> • If you load only the UFT Flex Add-in, then only the Flex 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>Relevant tasks</p>	<ul style="list-style-type: none"> • "How to Set Up the Adobe Flash Player Debugger to Enable UFT GUI Testing" on page 156 • "How to Compile Flex Applications for UFT Testing" on page 162 • "How to Open Flex Applications Using the Runtime Loader" on page 158
<p>See also</p>	<p>"Enabling UFT to Identify Objects in your Flex Application" on page 152</p> <p>"Considerations for Working with the Flex Add-in" on page 155</p>

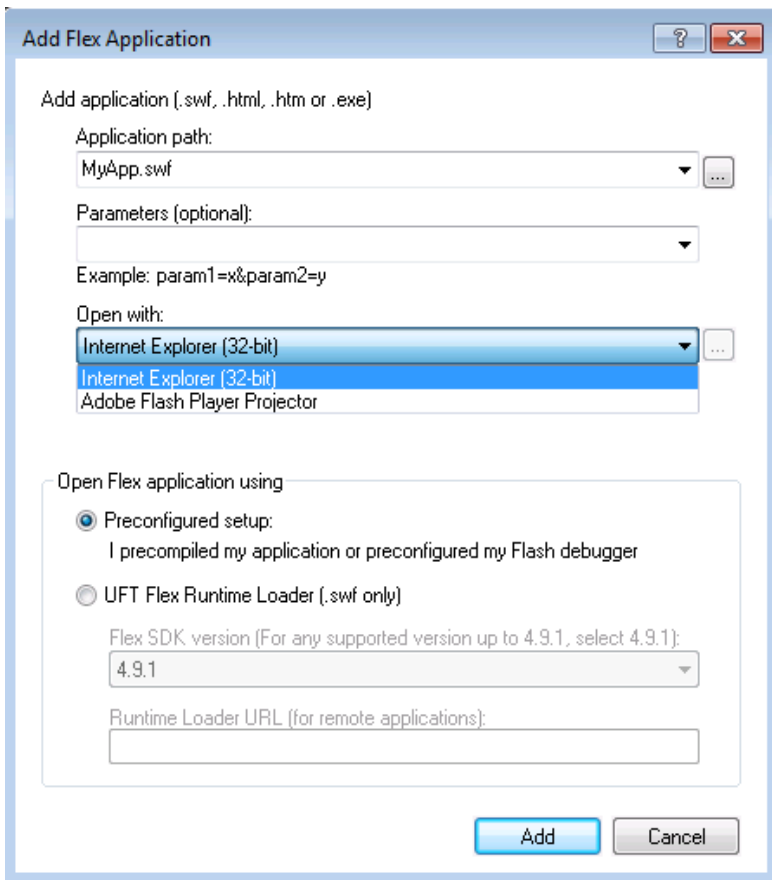
User interface elements are described below:



UI Elements	Description
<p>Record and run test on any open Flex application</p>	<p>Instructs UFT to record all operations performed on any Flex application that is open while recording your test, and to run steps on any such application during a run session.</p> <p>To enable UFT to record on Flex applications, do one of the following:</p> <ul style="list-style-type: none"> • Run the Flex application using the Adobe Flash Player Debugger. • Run the application using the UFT Flex Runtime Loader • Manually prepare the Flex application for UFT testing. <p>For details, see "Enabling UFT to Identify Objects in your Flex Application" on page 152.</p>
<p>Open the following applications when a record or run session begins</p>	<p>Instructs UFT to open specific Flex applications at the beginning of a record or run session.</p> <div style="background-color: #f0f0f0; padding: 5px;"> <p>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 UFT recognizes. Even if this radio button is selected and no application is specified, UFT can still record, recognize, and run on any open UFT-ready Flex application.</p> </div>
<p>Flex application details</p>	<p>The details of the applications to open.</p> <p>Each row in the table lists one application and specifies the following:</p> <ul style="list-style-type: none"> • The location of the application • The parameters to pass to the application (optional) • The browser or Flash player with which to open the Flex application <p>For more details, see "Add/Edit Flex Application Dialog Box".</p>
	<p>Add. Opens the "Add/Edit Flex Application Dialog Box" (described on page 168), which enables you to add a new application to the application list. You can add up to ten applications.</p>
	<p>Edit. Opens the "Add/Edit Flex Application Dialog Box" (described on page 168), which enables you to edit the application details for the selected application.</p>
	<p>Delete. Removes the selected application from the application list.</p>

Add/Edit Flex Application Dialog Box

This dialog box enables you to add details for an application or to edit details for the application that you selected in the Flex tab of the Record and Run Settings dialog box.

Before you select options in this dialog box, you must determine the method you plan to use to enable UFT to communicate with your Flex application. For details, see ["Enabling UFT to Identify Objects in your Flex Application" on page 152](#).



To access	In the "Flex Tab (Record and Run Settings Dialog Box)" (described on page 166), click the Add button  or the Edit button  .
Important information	<ul style="list-style-type: none"> For each application, the details entered in this dialog box are displayed as a single row in the application details table in the Flex tab of the Record and Run Settings dialog box.
Relevant tasks	<ul style="list-style-type: none"> "How to Set Up the Adobe Flash Player Debugger to Enable UFT GUI Testing" on page 156 "How to Open Flex Applications Using the Runtime Loader" on page 158 "How to Compile Flex Applications for UFT Testing" on page 162
See also	"Flex Tab (Record and Run Settings Dialog Box)" on page 166 "Enabling UFT to Identify Objects in your Flex Application" on page 152

"Considerations for Working with the Flex Add-in" on page 155

User interface elements are described below:

UI Elements	Description
Application path	<p>The file system path or URL to the Flex application to open.</p> <p>Note:</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 Parameters box. If your application is the default page on the Web server host, the URL that you enter does not have to include the file name. For example, if http://mysite/home and http://mysite/myApp.html open the same application, you can enter http://mysite/home. Unicode character support: The Flex Runtime Loader fails to open the application if the file system path includes certain Unicode fullwidth characters. For example, make sure not to include English fullwidth letters or Japanese fullwidth number characters.
Parameters	<p>(Optional) Parameters to pass to the application when UFT opens it.</p> <p>For Flex applications, enter the parameter names and values, separated by the ampersand (&) symbol, using the following syntax:</p> <pre data-bbox="412 957 1370 1062"><parameter name>=<parameter value>&<parameter name>=<parameter value></pre> <p>For Adobe Air applications (.exe), enter the parameter values, separated by spaces:</p> <pre data-bbox="412 1136 1370 1209"><value1> <value2></pre>
Open with	<p>The program with which to open the Flex application (not relevant for standalone (.exe) applications).</p> <p>Possible values:</p> <ul style="list-style-type: none"> <Empty> (for .exe applications) Internet Explorer (32bit) Adobe Flash Player Projector (for .swf applications only) <p>If you select Adobe Flash Player Projector, you can use the browse button to select a specific Flash Player program to use for this application. The path to the selected program is then displayed under the drop down list.</p>
Open Flex application using preconfigured setup	<p>Select this option to indicate that you have already either:</p> <ul style="list-style-type: none"> recompiled your Flex application as described in "How to Compile Flex Applications for UFT Testing" on page 162 <p>or</p> <ul style="list-style-type: none"> you have set up the relevant Flash Player Debugger as described in "How to Set Up the Adobe Flash Player Debugger to Enable UFT GUI Testing" on page 156. (not relevant for *.exe applications).
Open Flex	<p>Instructs UFT to open the specified Flex application using the UFT Flex Runtime Loader.</p>

UI Elements	Description
application using UFT Flex Runtime Loader (.swf only)	This option is relevant only when testing .swf files directly, and if you are not using the Flash Player Debugger to run the application. For more details, see " How to Open Flex Applications Using the Runtime Loader " on page 158.
Flex SDK version	The version of Runtime Loader for UFT to use. <ul style="list-style-type: none"> • 4.9.1 - (default) For testing Flex applications developed using Flex SDK version 4.9.1 and earlier supported versions. • 4.12.1 - For testing Flex applications developed using Flex SDK version 4.12.x. This field is not relevant if you specify a Runtime Loader URL .
Runtime Loader URL	Only for applications hosted on a Web server: The URL to the Runtime Loader you want to use to open the Flex application. <div style="background-color: #f0f0f0; padding: 10px; margin-top: 10px;"> <p>Note:</p> <ul style="list-style-type: none"> • You must place a copy of the Runtime Loader on the same server as the Flex application, and provide the URL to this file. When you place the Runtime Loader file on the Web Server, select the one that matches the version used to develop your application, as described in "How to Open Flex Applications Using the Runtime Loader" on page 158. • If your Flex application is located on the file system, do not specify a Runtime Loader. UFT uses one of the Runtime Loaders installed with UFT (according to the specified SDK version). </div>

Troubleshooting and Limitations - Flex Add-in

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

General Functionality

- The Flex Add-in does not provide backward compatibility with the Adobe Flex Add-in for QuickTest, and uses a different set of test objects, methods, and properties. Legacy QuickTest tests recorded using the Adobe Flex Add-in cannot be used, and they cannot be upgraded to be used with the UFT Flex Add-in.
- The Flex Add-in does not support cross-domain or cross-host Flex applications. These types of applications are Flex applications where the HTML and SWF files are served from different domains, or from different hostnames within the same domain. For example, if an HTML page on **www.mysite.com** references an SWF file located on **www.anothersite.com**, or in **content.mysite.com**.
- The Flex Add-in is not supported on Internet Explorer 11 when the Enhanced Protected Mode is turned on.
- Testing Flex applications in UFT is only supported in Internet Explorer 32-bit browser versions.
- The UFT Flex Runtime Loader does not support applications that contain **mx::AreaChart** controls.
- To communicate with the Flex application, UFT selects an available port within the range 24654 - 24663. Make sure that at least one of these ports is available on the UFT computer.
 - On a Windows server, multiple users can run multiple instances of UFT. To test Flex applications, you must have one port in this range available for each UFT instance.
 - If you are testing a Flex application, you may experience a delay (up to one minute) from the moment you open UFT and your Flex application, until UFT can recognize objects in the application. This is due to the time it may take for UFT to locate an available port, as it cycles through the ports in this range, waiting for the socket connection timeout for each.
- If you are using Flex SDK version 3.6.0, you must set the Flash Player target version to 10.2 or higher when compiling your Flex application.

Active Screen

- The Active Screen pane is not fully supported for Flex test objects, and may not display the recorded steps correctly.

Object Identification and Functionality

- When identifying objects in a Flex application opened in a Web browser, the FlexWindow top-level test object is contained in a Page object.
- The UFT Flex add-in recognizes Advanced Data Grid controls as FlexTable test objects, and supports basic table functionality for these controls. In addition, UFT supports **ExpandRow**, **CollapseRow**, and

SortByColumn operations for tables of this type. Other abilities of Advanced Data Grid controls are not supported.

- The Flex Add-in does not support the **mx.controls::OLAPDataGrid** Flex control.
- The Navigate and Learn option is not supported in the following cases:
 - Windowless Flex applications.
 - Flex applications opened on Windows 8 or Windows 2012 (or later). (Such applications are opened as windowless Flex applications)
 - Flex applications opened in Internet Explorer using a URL with an **.swf** file extension.

Workaround: To simultaneously add all or specific child objects from a windowless Flex application to an object repository, do the following:

- a. Start by adding one of the Flex child objects to the repository. In the **Object Selection - Add to Repository** dialog box, select the parent FlexWindow object instead of the original object you had selected.
- b. In the **Define Object Filter** dialog box, select either **All object types** to learn all child objects, or click **Select** to select the specific types of child objects you want to add.

For more details about adding objects to object repositories, see the chapter on test objects in the *HP Unified Functional Testing User Guide*.

Part 6: Java Add-in

This section includes:

["Java Add-in - Quick Reference" on page 174](#)

["Java Add-in - Testing and Configuration" on page 176](#)

["Java Add-in - Test Objects" on page 201](#)

Java Add-in - Quick Reference

You can use the UFT Java Add-in to test Java user-interface objects (controls).

The following tables summarize basic information about the Java Add-in and how it relates to some commonly-used aspects of UFT.

General Information	
Supported Environments	<ul style="list-style-type: none"> 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 <i>HP Unified Functional Testing Product Availability Matrix</i>, available from the HP Support Matrix page (requires an HP passport).
Important Information	"Considerations - Java Add-in" on page 178
Test Object Methods and Properties	The Java Add-in provides customized Java test objects, methods, and properties that can be used when testing objects in Java applications. For details, see the Java section of the <i>HP UFT Object Model Reference for GUI Testing</i> .
Checkpoints and Output Values	<ul style="list-style-type: none"> See the sections describing checkpoints and output values in the <i>HP Unified Functional Testing User Guide</i>. See "Text Checkpoint and Text Output Value Steps for Java Objects" on page 203. See "GUI Checkpoints and Output Values Per Add-in" on page 512
Extending the Java Add-in	"Java Add-in Extensibility" (described on page 179) enables you to develop support for testing third-party and custom Java controls that are not supported out-of-the-box by the UFT Java Add-in.
Troubleshooting and Limitations	" Troubleshooting and Limitations - Java Add-in" on page 193

Prerequisites	
Opening Your Application	<p>You can open your Java application before or after opening UFT.</p> <div style="background-color: #f0f0f0; padding: 5px;"> <p>Note: If you cannot open your Java application after starting UFT, you may have a memory fragmentation issue. Check your memory settings, and see "Opening Java Applications After Opening UFT" on page 196.</p> </div>
Add-in Dependencies	The UFT Java Add-in can be installed and run together with any other UFT 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.

Configuration	
Options Dialog Box	<p>Use the Java pane.</p> <p>(Make sure that a GUI test is open and select Tools > Options > GUI Testing tab > Java node.)</p>

	See "Java Pane (Options Dialog Box > GUI Testing Tab)" on page 182.
Record and Run Settings Dialog Box (tests only)	Use the Java tab. (Record > Record and Run Settings) See "Java Tab (Record and Run Settings Dialog Box)" on page 187.
Test Settings Dialog Box (tests only)	Use the Java pane. (File > Settings > Java node) See "Java Pane (Test/Business Component Settings Dialog Box / Application Area - Additional Settings Pane)" on page 189.
Custom Active Screen Capture Settings Dialog Box (tests only)	Use the Java section. (Tools > Options > GUI Testing tab > Active Screen node > Custom Level) See the section on the Custom Active Screen Capture Settings dialog box in the <i>HP Unified Functional Testing User Guide</i> .
Application Area Additional Settings pane (business components only)	<ul style="list-style-type: none"> Use the Java pane. In the application area, select Additional Settings > Java in the sidebar. See "Java Pane (Test/Business Component Settings Dialog Box / Application Area - Additional Settings Pane)" on page 189. (For business components, the settings displayed in this pane are read-only. To change the Java pane settings for a business component, open its associated application area and use the application area's Additional Settings > Java pane.)

Java Add-in - Testing and Configuration

This chapter includes:

- [Java Add-in - Overview](#)177
 - [Java Add-in Environments](#) 177
- [Considerations - Java Add-in](#) 178
- [Java Add-in Extensibility](#) 179
- [How to Disable Dynamic Transformation Support \(Advanced\)](#) 180
- [Java Pane \(Options Dialog Box > GUI Testing Tab\)](#) 182
 - [Advanced Java Options Dialog Box](#) 184
- [Java Tab \(Record and Run Settings Dialog Box\)](#)187
- [Java Pane \(Test/Business Component Settings Dialog Box / Application Area - Additional Settings Pane\)](#)189
- [Java Environment Variables Settings](#)192
 - [Record and Run Environment Variables for Java Objects](#)193
- [Troubleshooting and Limitations - Java Add-in](#) 193
 - [Running Another Java Application or Applet with the Same Settings](#) 193
 - [Identifying and Solving Common Problems](#) 194
 - [General Notes and Limitations](#) 195

Java Add-in - Overview

This chapter explains how to use UFT to set testing preferences and to record and run steps on Java applets and applications. The chapter assumes basic knowledge of UFT features and capabilities. For details on working with UFT, see the *HP Unified Functional Testing User Guide*.

Note: Some of the features described in this chapter are relevant only for tests and scripted components. For details on the features that are available when working with keyword components, see the *HP Unified Functional Testing User Guide*.

Java Testing Options

You can use the **Java** pane of the Options dialog box (**Tools > Options > GUI Testing** tab > **Java** node) to set UFT 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.

For details, see:

- "[Java Pane \(Options Dialog Box > GUI Testing Tab\)](#)" on page 182
- "[Advanced Java Options Dialog Box](#)" on page 184

Java Settings - Tests and Components

You define and view Java test or business component settings using one of the following:

- The Java pane of the Test Settings dialog box.
- The Java pane of the Business Component Settings dialog box (view only).
- The Java pane of the application area's Additional Settings pane.

For details, see "[Java Pane \(Test/Business Component Settings Dialog Box / Application Area - Additional Settings Pane\)](#)" on page 189.

Java Add-in Environments

The Java Add-in uses a mechanism that supports multiple Java environments (such as IBM JRE, Oracle 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 Unified Functional Testing Product Availability Matrix*, available from the [HP Support Matrix page](#) (requires an HP passport).

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 **-Xrunjvmsupport** option, which is supplied to the JVM. If the **-Xrunjvmsupport** 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:jvmsupport**, which is defined in the `JAVA_TOOL_OPTIONS` environment variable.

Note: When working with Oracle Java 6 or Java 7 there is no conflict between **-agentlib:jvmsupport** (defined in the `JAVA_TOOL_OPTIONS` environment variable) and **-Xrunjvmsupport** (defined in the `_JAVA_OPTIONS` environment variable) because Java 6 and Java 7 ignore **-Xrunjvmsupport**.

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 200.](#)

The Java agent searches for the `jvmsupport.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 `jvmsupport.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 `jvmsupport.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**.

For task details, see ["How to Disable Dynamic Transformation Support \(Advanced\)" on page 180.](#)

Considerations - 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 Unified Functional Testing Java Support (version x.x.x.x) (<App> version x.x.x.x)." (where <App> is IE, IBM, or Oracle).
- The **Object** property can access only **public** methods and properties. A recommended alternative to using the **Object** property is to extend UFT support for the required Java object using UFT Java Add-in Extensibility. For details, see the *HP UFT 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 objects 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 details, see the section on adding test objects using the Navigate and Learn toolbar in the *HP Unified Functional Testing User Guide*.

- If you want to use a control's native property for object identification, you can add the property to a Java test object as an identification property in the Add/Remove Properties Dialog Box dialog box. If you do this, consider the following:
 - You can add only native properties for which the control has a public **get** or **is** method that returns the property value.
 - If the native property name includes upper-case letters, then in the corresponding identification property name that you create, you must replace each of the upper-case letters except the first one with `<lower-case letter>`. For example, to use the native property `OneSmallProp`, add an identification property named `One_small_prop`.

For more details on adding identification properties, see the section on the Object Identification dialog box in the *HP Unified Functional Testing User Guide*.

- In early 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 details, see the section on Updating Test Object Descriptions in the *HP Unified Functional Testing User Guide*.
- In UFT, 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.
- When working with tests, if you create a checkpoint on an SWT-based Java tree with columns, a table checkpoint is created.
- For details on UFT functionality, see the *HP Unified Functional Testing User Guide*.

Java Add-in Extensibility

UFT 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 UFT Java Add-in.

If the test object class that UFT 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 UFT 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:

- UFT 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 UFT 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 UFT Extensibility Documentation program group (**Start > All Programs > HP Software > HP Unified Functional Testing > Extensibility > Documentation** or the <UFT installation folder>\help\Extensibility folder).

Note: For details on accessing UFT and UFT tools and files in Windows 8 and Windows Server 2012, see ["Accessing UFT in Windows 8.X or Higher Operating Systems" on page 517](#).

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

How to Disable Dynamic Transformation Support (Advanced)

This task describes how to disable the dynamic transformation support mechanism if it does not work properly, and how to manually configure the Java environment to use the Java Add-in without dynamic

transformation support. For general details about using the Java Add-in with multiple environments, see ["Java Add-in Environments" on page 177](#).

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.

This task includes the following steps:

- ["Save the dynamically transformed classes" below](#)
- ["Disable dynamic transformation support" below](#)
- ["Results" on the next page](#)

1. Save the dynamically transformed classes

- a. 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.

- b. 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.
- c. 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`).
- d. If you temporarily removed the `-Xincgc` option from the command line, you can restore it now.

Now that you saved the transformed classes, you are ready to disable dynamic transformation support.

2. Disable dynamic transformation support

- a. 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.

- b. 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, such as `C:\JavaSupportClasses`. For example, after your modification the `_JAVA_OPTIONS` environment variable might look like this:

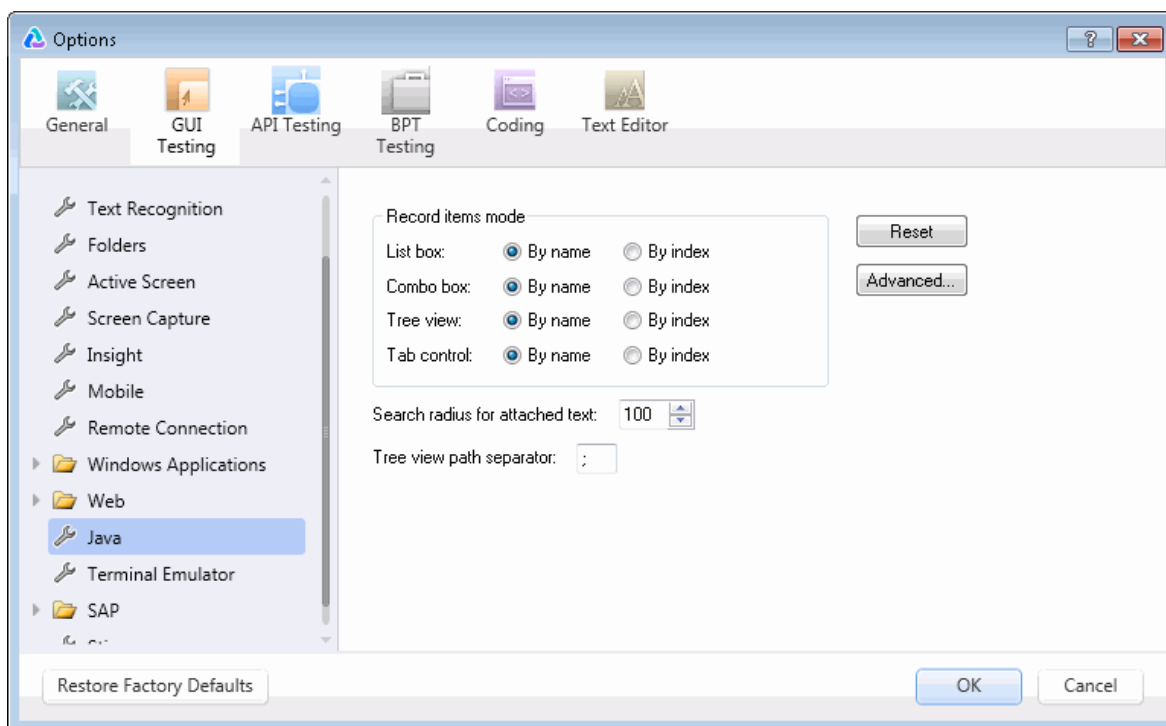
```
-Xbootclasspath/p:C:\JavaSupportClasses\Final -  
Xbootclasspath/a:C:\PROGRA~1\HP\  
UNIFIE~1\bin\JAVA_S~1\classes;C:\PROGRA~1\HP\UNIFIE~1\bin\JAVA_  
S~1\classes\jasmine.jar
```

3. Results

The saved transformed classes are now used instead of dynamic transformation.

Java Pane (Options Dialog Box > GUI Testing Tab)

This pane enables you to configure how UFT records and runs tests on Java applets or applications.



To access	<ol style="list-style-type: none">Do one of the following:<ul style="list-style-type: none">Ensure that a GUI test, action, or component is in focus in the document pane.In the Solution Explorer, select a GUI test or component node or one of its child nodes.Select Tools > Options > GUI Testing tab > Java node.
Important	<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

information	<p>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.</p> <ul style="list-style-type: none"> The Restore Factory Defaults button resets all Options dialog box options to their defaults.
Related tasks	<ul style="list-style-type: none"> "How to Modify Options for Recording on Java Tables" on page 207 "Record and Run Environment Variables for Java Objects" on page 193
See also	<ul style="list-style-type: none"> "Advanced Java Options Dialog Box" on the next page. "Java Pane (Test/Business Component Settings Dialog Box / Application Area - Additional Settings Pane)" on page 189 "Java Tab (Record and Run Settings Dialog Box)" on page 187 "Recording Steps on Java Objects" on page 202

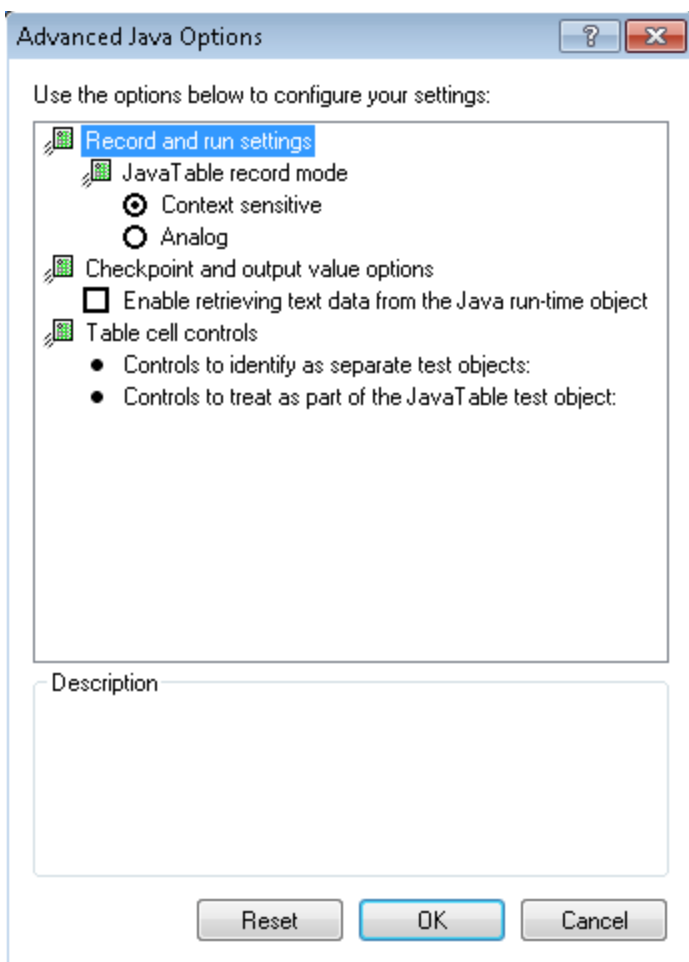
User interface elements are described below:

UI Element	Description
Record items mode	<p>Determines how UFT records operations on items in List box, Combo box, Tree view, and Tab control objects. The following options are available 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>Note:</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>The maximum distance in pixels to search for attached text.</p> <p>Default value: 100</p> <p>Note:</p> <ul style="list-style-type: none"> This option is relevant only when the <code>label</code> identification property is unavailable. This option corresponds to the <code>Setting.Java("max_text_distance")</code> variable.
Tree view path separator	<p>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>Note:</p> <ul style="list-style-type: none"> If you enter more than one character, UFT treats each of the characters as a separator (but not both of them in sequence). If a path contains two consecutive separators, UFT 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 <code>MyNode%\$MySubNode</code>, then UFT treats the % character as a separator for a node with no name, and the \$ character as the separator for an additional node named <code>MySubNode</code>.

UI Element	Description
	<ul style="list-style-type: none"> 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 settings to their default values.
Advanced	Opens the Advanced Java Options dialog box. For details, see "Advanced Java Options Dialog Box" below.

Advanced Java Options Dialog Box

This dialog box 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.



To access Make sure that a GUI test is open and select **Tools > Options > GUI Testing** tab > **Java** node > **Advanced** button. Then do the following:

	<ol style="list-style-type: none"> 1. 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. <p>Note: Specify editor class names separated by a space, tab, newline, or return character. Values are case sensitive.</p> <ol style="list-style-type: none"> 4. When you finish editing the value, click another location in the dialog box to set the value.
Important information	<ul style="list-style-type: none"> • 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. • Any changes you make are not applied to the currently open test or business component. To apply your changes, close your test or business component and reopen it. • You can restore the default settings in the Advanced Java Options dialog box by clicking the Reset button.
Related tasks	<p>"How to Modify Options for Recording on Java Tables" on page 207</p>
See also	<ul style="list-style-type: none"> • "Java Pane (Options Dialog Box > GUI Testing Tab)" on page 182 • "Java Pane (Test/Business Component Settings Dialog Box / Application Area - Additional Settings Pane)" on page 189 • "Java Tab (Record and Run Settings Dialog Box)" on page 187 • "Recording Steps on Java Objects" on page 202 • "Text Checkpoint and Text Output Value Steps for Java Objects" on page 203

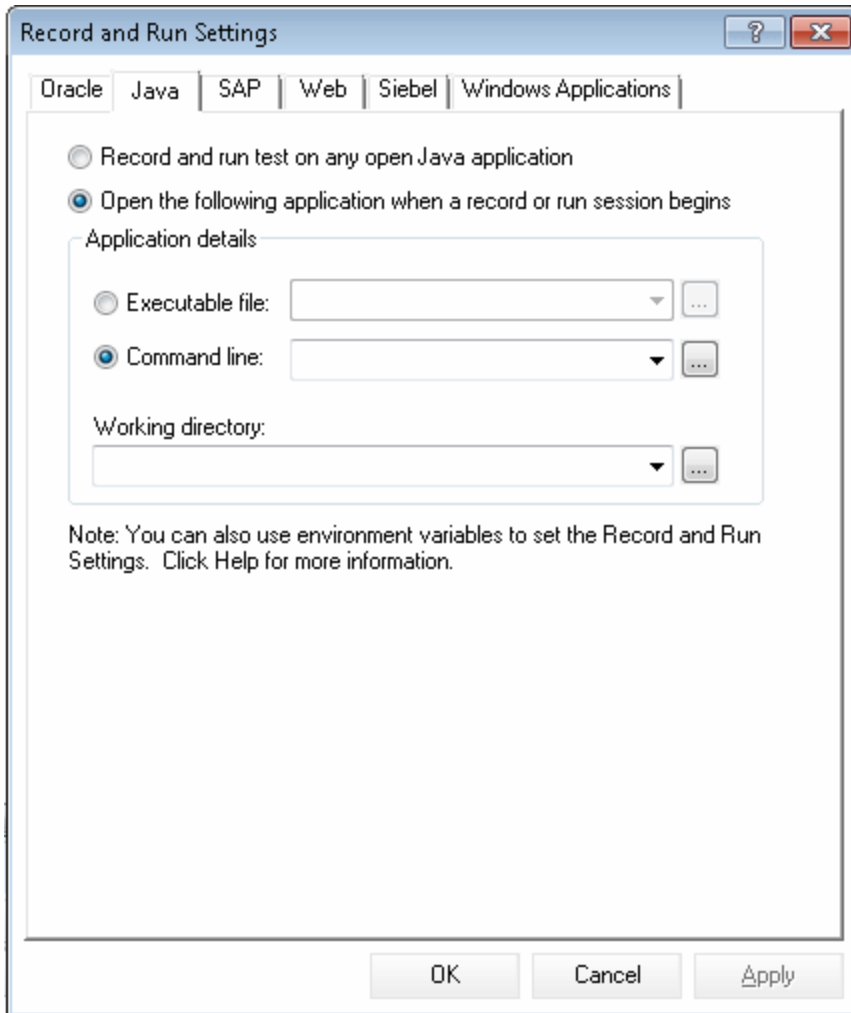
User interface elements are described below:

UI Element	Description
JavaTable record mode	<p>The record mode for table objects. The following modes are available:</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>
Checkpoint and output value options	<p>The preferences for checkpoint and output value steps on Java objects. The following option is available:</p> <p>Enable retrieving text data from the Java run-time object: Enables UFT to retrieve text information from the Java objects in the application for checkpoints and output value steps. This option is not relevant if UFT is configured to use the OCR mechanism for text recognition (Tools > Options > GUI Testing tab > Text Recognition node)</p> <p>Note:</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. • For a list of supported checkpoints and output values for the Java Add-in, see "Java Add-in - Quick Reference" on page 174.

UI Element	Description
Table cell controls	<p>The preferences for the way that UFT identifies controls inside table cells. The following options are available:</p> <ul style="list-style-type: none">• Controls to identify as separate test objects: Specifies the list of controls that you want UFT to identify as separate test objects and not as part of a <code>JavaTable</code> object. Use this option to access methods that are specific to the object type or to otherwise improve the functionality of steps that UFT would normally record and run as operations on a <code>JavaTable</code> object. <div data-bbox="402 499 1370 751"><p>Note:</p><ul style="list-style-type: none">• This option is relevant for <code>JTable</code> Swing toolkit tables.• Control class names must be 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.</div> <ul style="list-style-type: none">• Controls to treat as part of the <code>JavaTable</code> test object: Specifies the list of controls for which you want UFT to record and run <code>JavaTable</code> operations. Use this option to record and run <code>JavaTable</code> operations (such as <code>SetCellData</code> and <code>Select</code>) on controls that UFT would normally treat as separate test objects. <div data-bbox="402 877 1370 1129"><p>Note:</p><ul style="list-style-type: none">• This option is relevant for <code>JTable</code> Swing toolkit tables.• Editor class names must be 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.</div> <p>See also:</p> <ul style="list-style-type: none">• "Modify table cell control options" on page 207• "Find the toolkit class of a <code>JTable</code> cell editor" on page 208

Java Tab (Record and Run Settings Dialog Box)

This tab enables you to instruct UFT to open your Java applet or application each time you begin a recording session, or to instruct UFT to record on any open Java application.



<p>To access</p>	<p>Record > Record and Run Settings and select the Java tab.</p> <p>Note: The Record and Run Settings dialog box opens automatically each time you begin recording a new test or business component (unless you open the dialog box and set your preferences manually before you begin recording).</p>
<p>Important information</p>	<ul style="list-style-type: none"> • 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. • Verifying record and run settings. When you run a test, or if you begin a new recording session on an existing test, UFT automatically uses the existing record and run settings for the test and does not open the

	<p>Record and Run Settings dialog box. However, it is important to confirm that the options in the Java Tab of the Record and Run Settings Dialog Box 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 recording session.</p> <ul style="list-style-type: none"> • Testing business components. Components do not require specific record and run settings to work with Java applets and applications. To record a business component, you need to first open the Java applet or application manually. Alternatively, you can include steps in your business component that connect to the Java applet or application, for example, you can include a step that contains the OpenApp operation. <p>When you begin recording a new business component, the Applications dialog box opens (unless you previously specified a Windows environment in the Business Component Settings dialog box or the application area's Additional Settings pane). Click OK in the dialog box without making modifications to begin recording. For details on the Applications pane and Applications dialog box, see "Record and run test on any open Windows-based application" on page 94.</p>
Related tasks	"How to Define Record and Run Settings for UFT Add-ins"
See also	<ul style="list-style-type: none"> • "Java Pane (Options Dialog Box > GUI Testing Tab)" on page 182 • "Advanced Java Options Dialog Box" on page 184 • "Recording Steps on Java Objects" on page 202 • "Record and Run Environment Variables for Java Objects" on page 193 • For details on the Web tab of the Record and Run Settings dialog box, see "Web-Based Application Support" on page 34. • For general details on the Record and Run Settings dialog box, see the <i>HP Unified Functional Testing User Guide</i>.

User interface elements are described below:

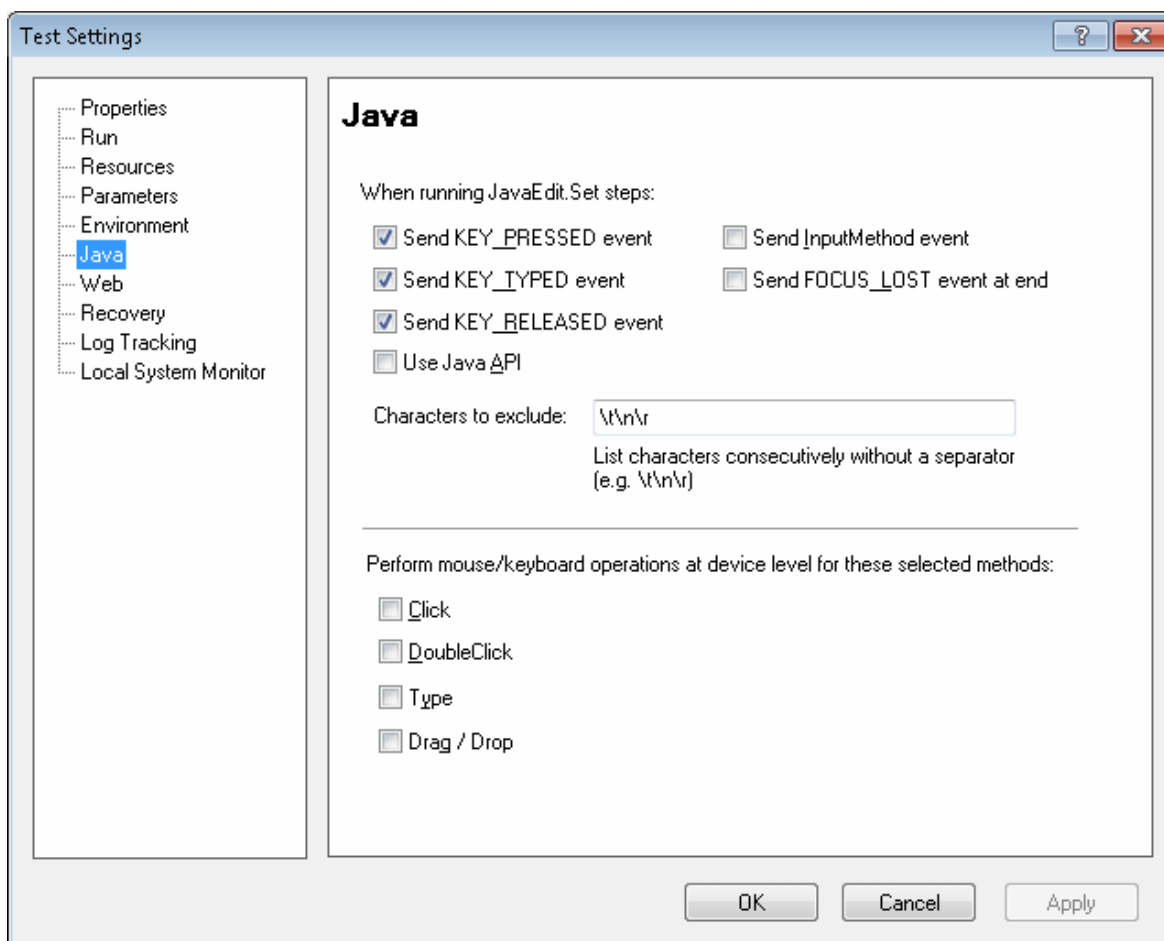
UI Element	Description
Record and run test on any open Java application	Instructs UFT to record and run the test on any open Java application or applet.
Open the following application when a record or run session begins	<p>Instructs UFT to open a new Java application or applet using the specified application details.</p> <p>Note:</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 UFT recognizes. Even if this radio button is selected and no application is specified, UFT 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.
Application details	<p>The details of the Java application on which to run the test:</p> <ul style="list-style-type: none"> • Executable file. Instructs UFT to open the specified executable or batch file. • Command line. Instructs UFT to open the application from the specified command line. • Working directory. Instructs UFT to run the specified executable file or command line from the specified directory. Make sure you specify the full directory path, for example,

UI Element	Description
	<p data-bbox="483 285 938 317">%ProgramFiles%\Java\jre1.6.0\bin.</p> <p data-bbox="469 344 526 369">Note:</p> <ul data-bbox="469 390 1330 590" style="list-style-type: none"><li data-bbox="469 390 1330 516">• 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 details, see "Record and Run Environment Variables for Java Objects" on page 193.<li data-bbox="469 533 1330 590">• Always enter a value in the Working directory field, otherwise UFT cannot open your Java application.

Java Pane (Test/Business Component Settings Dialog Box / Application Area - Additional Settings Pane)

This pane enables you to define Java test or business component settings. For business components, you use the application area's Additional Settings pane to define the settings, and the Business Component Settings dialog box to view the settings in read-only mode.

The following image displays the Java pane of the Test Settings dialog box. The content of this pane is identical to the content of the Java pane of the Business Component Settings dialog box and the application area's Additional Settings pane.



To access	Do one of the following: <ul style="list-style-type: none"> For a test or component: Select File > Settings > Java node. For an application area: Open the application area and select Additional Settings > Java in the sidebar.
Important information	<ul style="list-style-type: none"> The options shown in the Java pane are the same in the application area's Additional Settings pane, the Business Component Settings dialog box, and the Test Settings dialog box. The Business Component Settings dialog box displays the settings in read-only mode. To define the settings for a business component, use its application area's Additional Settings pane. 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. For details on the Test Settings and Business Component Settings dialog boxes, and on application areas, see the <i>HP Unified Functional Testing User Guide</i>.
Related tasks	"Record and Run Environment Variables for Java Objects" on page 193
See also	<ul style="list-style-type: none"> "Java Pane (Options Dialog Box > GUI Testing Tab)" on page 182

	<ul style="list-style-type: none"> • "Java Tab (Record and Run Settings Dialog Box)" on page 187 • "Recording Steps on Java Objects" on page 202
External resources	<ul style="list-style-type: none"> • For details on JFC or AWT-based Java key events and input methods, see Java documentation at http://www.oracle.com/technetwork/java/index.html. • For details on SWT-based Java key events, see Java documentation at http://www.eclipse.org.

User interface elements are described below:

UI Element	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 <code>JavaEdit.Set</code> 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 <code>KEY_PRESSED</code> 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 <code>KEY_TYPED</code> 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 <code>KEY_RELEASED</code> 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.
Characters to exclude	<p>Instructs UFT to ignore the specified characters during a run session. List characters consecutively, without a separator.</p> <p>Default value: <code>\t\n\r</code></p> <div style="background-color: #f0f0f0; padding: 5px;"> <p>Note:</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. </div>
Perform mouse/keyboard operations at device level for	<p>By default, UFT 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>

UI Element	Description
these selected methods	<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>

Java Environment Variables Settings

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

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

```
-XrunjvmsHook  
-Xbootclasspath/a:"<UFT installation folder>\bin\java_shared\classes";  
"<UFT installation folder>\bin\java_shared\classes\jasmine.jar"
```

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

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

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 5 or 6, 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.

Note:

You can also use short paths in these commands. For example:

```
-XrunjvmsHook -Xbootclasspath/a:C:\PROGRA~2\  
HP\UNIFIE~1\bin\ JAVA_S~1\classes;C:\PROGRA~2\  
HP\UNIFIE~1\bin\JAVA_S~1\classes\jasmine.jar
```

In this example, UFT is installed in the default installation folder (C drive, Program Files) on a Windows 7 computer. `PROGRA~2` denotes the Program Files (x86) folder, which is the Program Files folder on 64-bit operating systems.

Record and Run Environment Variables for Java Objects

You can override the values in the **Executable file**, **Command line**, and **Working directory** boxes in the ["Java Tab \(Record and Run Settings Dialog Box\)"](#) (described on page 187) by defining the Java application details using the following variables:

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 details on defining and working with environment variables, see the *HP Unified Functional Testing User Guide*.

Troubleshooting and Limitations - Java Add-in

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

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. For details, see

This section includes:

- ["Running Another Java Application or Applet with the Same Settings" below](#)
- ["Identifying and Solving Common Problems" on the next page](#)
- ["General Notes and Limitations" on page 195](#)

Running Another Java Application or Applet with the Same Settings

You can run another Java application or applet with the same settings as the one you are currently running to help determine whether you are encountering a general problem with the Java Add-in or an application-specific problem.

When running another Java application or applet, do the following:

- 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 another 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.

Identifying and Solving Common Problems

The UFT 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>	<p>Make sure that the Java Add-in is loaded with UFT. To check this, select Help > About Unified Functional Testing and verify that the Java Add-in check box is selected.</p> <p>You load the Java Add-in using the Add-in Manager. For details, see "How to Manage UFT 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 using Oracle Java JRE 5 or 6 on Microsoft Internet Explorer, 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 > All Programs > HP Software > HP Unified Functional Testing > Tools program group or the UFT installation folder\bin\java\classes\QTPJavaEnabler.jar file</p> <div style="background-color: #f0f0f0; padding: 5px; margin: 10px 0;"> <p>Note: For details on accessing UFT and UFT tools and files in Windows 8 and Windows Server 2012, see "Accessing UFT in Windows 8.X or Higher Operating Systems" on page 517.</p> </div> <p>For details, see:</p> <ul style="list-style-type: none"> • "Java Environment Variables Settings" on page 192 • "Using the Java Add-in on Applets Running on Internet Explorer" on page 200
<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 details, see:</p> <ul style="list-style-type: none"> • "Java Environment Variables Settings" on page 192

Indicator	Solution
	<p>Note: For details on accessing UFT and UFT tools and files in Windows 8 and Windows Server 2012, see "Accessing UFT in Windows 8.X or Higher Operating Systems" on page 517.</p>
<p>A different applet or application works with the 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 details, see:</p> <ul style="list-style-type: none"> • "Java Environment Variables Settings" on page 192
<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 "Java Environment Variables Settings" on page 192, 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 <code>-Xrunjvmhook</code> from the <code>_JAVA_OPTIONS</code> and <code>IBM_JAVA_OPTIONS</code> environment variables. Remove -Xrunjvmhook from the <code>_JAVA_OPTIONS</code> and <code>IBM_JAVA_OPTIONS</code> environment variables. 2. Add <code>-agentlib:jvmhook</code> at the beginning of the <code>_JAVA_OPTIONS</code> and <code>IBM_JAVA_OPTIONS</code> environment variables. Add -agentlib:jvmhook at the beginning of the <code>_JAVA_OPTIONS</code> and <code>IBM_JAVA_OPTIONS</code> environment variables. 3. Delete the <code>JAVA_TOOL_OPTIONS</code> environment variable.
<p>The add-in does not function properly with applications that run with the <code>-Xincgc</code> option.</p>	<p>Either remove the <code>-Xincgc</code> option, or run without dynamic transformation support.</p> <p>For details, see: "How to Disable Dynamic Transformation Support (Advanced)" on page 180.</p>
<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. Check that the <code>jvmhook.dll</code> is located within your java.library.path.</p>
<p>None of the indicators above describe my problem.</p>	<p>See "General Notes and Limitations" below</p>

General Notes and Limitations

This section contains general information and limitations about the Java Add-in, and includes the following sections:

- ["Opening Java Applications After Opening UFT"](#) on the next page
- ["Loading a Child Add-in of the Java Add-in"](#) on the next page

- ["Creating and Running Testing Documents" on the next page](#)
- ["Record and Run Options" on the next page](#)
- ["Working with Java Controls" on page 198](#)
- ["Test Objects and Methods" on page 199](#)
- ["Checkpoints and Output Values " on page 199](#)
- ["Running Java applications on the IBM Java Runtime Environment \(JRE\) 1.6 " on page 200](#)
- ["Using the Java Add-in on Applets Running on Internet Explorer" on page 200](#)

Opening Java Applications After Opening UFT

- If you are not able to open your Java application after you've opened UFT, you may have a memory space fragmentation issue, caused by loading a Windows .dll file. If Eclipse fails to start with higher memory settings, do one of the following:
 - Use a 64-bit Windows operating system and 64-bit JVM, with 64-bit Eclipse. Have a 64-bit virtual memory space can prevent you from encountering memory fragmentation issues.
 - Force Eclipse to start using the `java.exe` or `javaw.exe` file instead of the default `jvm.dll` startup file. To do this edit the `eclipse.ini` file by adding the following text, on two separate lines:

```
-vm  
<full path to the java.exe or javaw.exe file>
```

- Modify the Eclipse memory setting in the `eclipse.ini` file. For example, if the application fails to start with a parameter setting of `-Xmx512m`, use a parameter setting of `-Xmx256m` or `-Xmx384m` instead.
- In some cases, Java applications that run successfully when UFT is closed, fail to run if you open them while UFT is open. An error message is displayed: **Could not create the Java Virtual Machine.**

Workaround: Activate UFT's memory defragmenting by adding a line to the `mercury.ini` file:

Locate the `mercury.ini` file in your Windows folder (`%windir%`) and add a line to the **Memory_Defrag** section, as follows:

```
[Memory_Defrag]  
<process_name>.exe=1
```

`process_name`: The name of the Java engine or application that you are using.

1: Turns on defragmenting (use 0 to turn off, if necessary).

Loading a Child Add-in of the Java Add-in

When you select a child add-in under Java in the Add-in Manager, you load Java Add-in extensibility support for the selected environment.

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 UFT 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. This file is located in the following locations:
 - <UFT Installation Folder>\dat\Extensibility\Java\<add-in name>TestObjects.xml.
 - If working with ALM: <UFT Add-in for ALM Installation Folder>\dat\Extensibility\Java\<add-in name>TestObjects.xml.
2. 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.
3. Save the file and reopen UFT.
4. 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 business 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 UFT Java support, do not use `-Xincgc` in your command line, or, alternatively, do not use the dynamic transformation support mechanism. For details, see the *HP Unified Functional Testing 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 Unified Functional Testing User Guide*.

- A call to **.Object.startModal** of a `JavaInternalFrame` or `JavaDialog` object may cause UFT to behave unexpectedly until the dialog box is closed.
- The use of multi-byte 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.
- When you record a step that closes a Java dialog box, UFT records an additional **Close** statement.

Workaround: Manually delete the extraneous **Close** statement.

- 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 business 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 business 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 business**

components steps when you rename test objects option is selected in the **General** pane of the GUI Testing tab in the Options dialog box (**Tools > Options > GUI Testing** tab > **General** node).

- 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 details, see ["Text Checkpoint and Text Output Value Steps for Java Objects" on page 203](#).
- To create a new table checkpoint on a Java table while editing a test or business 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 business 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.

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 Internet Explorer

In some cases, when running Java applets using Oracle Java JRE 5 or 6 on Microsoft Internet Explorer, 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:"<UFT installation folder>\bin\java_shared\classes";"<UFT installation folder>\bin\java_shared\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 > All Programs > HP Software > HP Unified Functional Testing > Tools > Java Add-in JRE Support Tool** or `UFT installation folder\bin\java\classes\QTPJavaEnabler.jar` file

Note: For details on accessing UFT and UFT tools and files in Windows 8 and Windows Server 2012, see ["Accessing UFT in Windows 8.X or Higher Operating Systems" on page 517](#).

Java Add-in - Test Objects

This chapter includes:

- [Recording Steps on Java Objects](#) 202
 - [Recording Steps on Jtable Cell Editors](#)203
 - [Text Checkpoint and Text Output Value Steps for Java Objects](#)203
 - [Full Object Hierarchy Views](#) 204
- [Advanced Java Test Object Methods](#)204
 - [CreateObject Method](#) 205
 - [GetStatics Method](#) 205
 - [FireEvent / FireEventEx Methods](#)206
- [How to Modify Options for Recording on Java Tables](#)207

Recording Steps on Java Objects

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

If you try to record an operation on an unsupported or custom Java object, UFT records a generic Java Object.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 UFT Java Add-in Extensibility. For details, see the *HP UFT Java Add-in Extensibility Developer Guide*.

Note: The way in which UFT records operations depends on the type of JTable cell editor in the table cell. For details, see ["Recording Steps on Jtable Cell Editors" on the next page](#).

The UFT 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**.

Example

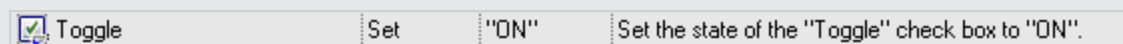
In a test, if you record a click on a Java check box, the Keyword View may be displayed as follows:

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".

UFT records this step in the Editor as:

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

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



You can view the recorded hierarchy of a test object in the object repository. You can also 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 a related task, see ["How to Modify Options for Recording on Java Tables" on page 207.](#)

This section also includes:

- [Recording Steps on Jtable Cell Editors](#) 203
- [Text Checkpoint and Text Output Value Steps for Java Objects](#) 203
- [Full Object Hierarchy Views](#)204

Recording Steps on Jtable Cell Editors

UFT records operations differently, depending on the type of **JTable** cell editor in the table cell.

If you are recording on standard cell editors in Swing **Jtable** tables, UFT records operations by default in the same way it records other table objects, using **SetCellData** statements.

However, 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, UFT 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, UFT might record the following statement:

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

instead of:

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

Text Checkpoint and Text Output Value Steps for 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 UFT 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 business 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](#)" (described on page 184).

Full Object Hierarchy Views

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. You can also access non-recorded objects from the Object Selection dialog box that opens when using the Step Generator dialog box (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 business 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 details on the Object Spy and Object Selection dialog box, see the *HP Unified Functional Testing User Guide*.

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 UFT and the application being tested.

This section includes:

- [CreateObject Method](#)205
- [GetStatics Method](#)205
- [FireEvent / FireEventEx Methods](#) 206

CreateObject Method

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 details on the syntax of this method, see the **Java** section of the *HP UFT Object Model Reference for GUI Testing*.

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.

GetStatics Method

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 details on the syntax of this method, see the **Java** section of the *HP UFT Object Model Reference for GUI Testing*.

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!"
```

FireEvent / FireEventEx Methods

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 details on the syntax of these methods and for the list of pre-defined event constants, see the **Java** section of the *HP UFT Object Model Reference for GUI Testing*.

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).

How to Modify Options for Recording on Java Tables

This task describes how to modify some recording options for recording on Java table, as well as identify the toolkit class for an editor for use with the **table_external_editors_list**

This task includes the following steps:

- ["Modify the default JTable recording behavior for SetCellData statements \(advanced\)" below](#)
- ["Modify table cell control options" below](#)
- ["Find the toolkit class of a JTable cell editor" on the next page](#)

Modify the default JTable recording behavior for SetCellData statements (advanced)

If the default recording behavior for JTables does not provide the desired value for the SetCellData statement of a particular editor, set that editor to be recorded, like a custom cell editor, in terms of the operation performed on the object inside the cell.

Do one of the following:

- In the ["Advanced Java Options Dialog Box"](#) (described on page 184), select **Table cell controls > Controls to identify as separate test objects**, and then specify specific cell editor types that should always be treated as separate objects, and not as part of a JavaTable object.
- Create a Setting.Java ("table_internal_editors_list") statement. For details, see ["Advanced Java Options Dialog Box" on page 184](#), and the *HP UFT Object Model Reference for GUI Testing*.

Modify table cell control options

You can specify a list of table cell controls that you want UFT to identify as separate test object, or for which you want UFT to record and run JavaTable operations.

Note:

- Any changes you make are not applied to the currently open test or business component.
- You can restore the default settings in the ["Advanced Java Options Dialog Box"](#) (described on page 184) by clicking the **Reset** button.

1. In the ["Advanced Java Options Dialog Box"](#) (described on page 184), 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. To apply your changes to the currently open test or business component, close the document and then reopen it.

Find 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 by doing one of the following:

- **Use the Object Spy to retrieve the value.**
- **Run a short test in UFT to retrieve the value.** You may want to do this when working with a cell that does not stay activated for long enough to capture the cell with the Object Spy. For example, a cell that is no longer active after a check box is selected or cleared.
- **Create a user-defined function and insert it as a step in your test.** You may want to do this when working with business components.

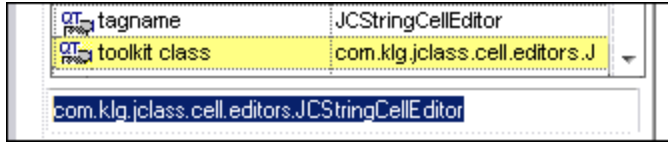
You can insert steps similar to the following example:

```
' 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()
```

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 details on using the Object Spy dialog box, see the *HP Unified Functional Testing 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.

Part 7: Mobile Add-in

This section includes:

["Mobile Add-in - Quick Reference" on page 211](#)

["Mobile Tab \(Record and Run Settings\)" on page 212](#)

["Mobile Pane \(Options Dialog Box > GUI Testing Tab\) " on page 212](#)

Mobile Add-in - Quick Reference

Built on native HP technology, HP Mobile Center integrates mobile testing with Sprinter, Unified Functional Testing, LoadRunner, and Performance Center.

You can use UFT to record and run GUI tests and components on mobile applications that are running on real mobile devices hosted on HP Mobile Center. After you connect your devices to HP Mobile Center, you can immediately start testing them using the Mobile Add-in.

The following tables summarize basic information about the Mobile Add-in and how it relates to some commonly-used aspects of UFT.

Note: For additional information on using UFT with HP Mobile Center, see the UFT section in the [HP Mobile Center Help](http://mobilecenter.hp.com/docs/en/1.10/) at <http://mobilecenter.hp.com/docs/en/1.10/>.

General Information	
Supported Environments	<ul style="list-style-type: none"> For details on supported versions of HP Mobile Center, see: http://mobilecenter.hp.com/docs/en/1.10/Content/My_Product_Matrix.htm For supported mobile devices, see: http://mobilecenter.hp.com/docs/en/1.10/Content/Configuring_and_connecting_devices.htm
Test Object Methods and Properties	The Mobile Add-in provides test objects, methods, and properties that can be used when testing objects in mobile applications. For details, see the Mobile section of the <i>HP UFT Object Model Reference for GUI Testing</i> .
Checkpoints and Output Values	<ul style="list-style-type: none"> See the sections describing checkpoints and output values in the <i>HP Unified Functional Testing User Guide</i>. See "GUI Checkpoints and Output Values Per Add-in" on page 512
Other	You must have the HP Mobile Center installed and configured on a server in order to use the Mobile Add-in. Once the HP Mobile Center is set up, you can connect real mobile devices to the Mobile Center. Then, UFT communicates with the mobile devices to create tests of the mobile applications.
Considerations and Limitations	See: http://mobilecenter.hp.com/docs/en/1.10/Content/mobile_on_UFT_Lims.htm .

Prerequisites	
Opening Your Application	The application is accessed through the remote access window that opens when you start recording or running a test, or that you open manually.
Add-in Dependencies	None

Configuration	
Options Dialog Box	Make sure that a GUI test is open and select Tools > Options > GUI Testing tab > Mobile

	<p>node.</p> <p>For details, see: http://mobilecenter.hp.com/docs/en/1.10/Content/mobile_on_UFT_define_cnx.htm#Options</p>
Record and Run Settings Dialog Box (tests only)	<p>Make sure that a GUI test is open and select Record > Record and Run Settings.</p> <p>For details, see: http://mobilecenter.hp.com/docs/en/1.10/Content/mobile_on_UFT_define_cnx.htm#TestSettings</p>

Mobile Tab (Record and Run Settings)

For details on defining record and run settings for mobile devices, see:

http://mobilecenter.hp.com/docs/en/1.10/Content/mobile_on_UFT_define_cnx.htm#TestSettings

Mobile Pane (Options Dialog Box > GUI Testing Tab)

For details on defining the UFT connection to HP Mobile Center, see:

http://mobilecenter.hp.com/docs/en/1.10/Content/mobile_on_UFT_define_cnx.htm#Options

Open Application in HP Mobile Center

For details on manually opening HP Mobile Center with an application pre-loaded on a device, see:

http://mobilecenter.hp.com/docs/en/1.10/Content/mobile_on_UFT_OpenRAWindow_Rec.htm

Part 8: Oracle Add-in

This section includes:

["Oracle Add-in - Quick Reference" on page 214](#)

["Oracle Add-in - Testing and Configuration" on page 217](#)

Oracle Add-in - Quick Reference

You can use the UFT Oracle Add-in to test Oracle Applications and Oracle Forms objects (controls).

The following tables summarize basic information about the Oracle Add-in and how it relates to some commonly-used aspects of UFT.

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 "Web-Based Application Support" on page 34 .
Supported Environments	For details on supported Oracle environments, see the Oracle Add-in section of the <i>HP Unified Functional Testing Product Availability Matrix</i> , available from the HP Support Matrix page (requires an HP passport).
Important Information	When working with the Oracle Add-in, you must: <ul style="list-style-type: none"> • Verify that the Oracle Name attribute is unique. See "How to Verify or Enable the Oracle Server Unique Name Attributes" on page 221. • Enable the Oracle Name attribute. See "How to Enable the Oracle Name Attribute" on page 221.
Test Object Methods and Properties	The Oracle Add-in provides test objects, methods, and properties that can be used when testing objects in Oracle applications. For detail, see the Oracle section of the <i>HP UFT Object Model Reference for GUI Testing</i> .
Checkpoints and Output Values	<ul style="list-style-type: none"> • See the sections describing checkpoints and output values in the <i>HP Unified Functional Testing User Guide</i>. • See "GUI Checkpoints and Output Values Per Add-in" on page 512
Troubleshooting and Limitations	See "Troubleshooting and Limitations - Oracle Add-in" on page 228 .

Prerequisites	
Opening Your Application	You can open your Oracle application before or after opening UFT.
Testing in Mozilla Firefox	<p>To test your Web application in Mozilla Firefox version 33 and later, you must enable the Functional Testing Extension in one of the following ways:</p> <ul style="list-style-type: none"> • If the Select Your Add-ons screen is displayed when you open Firefox, select to enable the Functional Testing Extension. • If the Install Add-on tab opens and displays Functional Testing Extension when opening Firefox, select the Allow this installation check box and click Continue. <p>Otherwise:</p> <ol style="list-style-type: none"> a. In Firefox, open the browser menu. b. In the menu, click Add-ons.

	<p>c. In the Add-ons Manager tab, select the Extensions node.</p> <p>d. In the Functional Testing Extension row, click the Enable button.</p> <p>To test your Web application in Mozilla Firefox versions earlier than 33, you must enable the legacy Firefox agent:</p> <ol style="list-style-type: none"> 1. Open the <UFT installation folder>\Installations\Firefox folder. 2. From the Firefox folder, drag the Legacy.xpi file into Firefox. 3. In Firefox, open the browser menu. 4. In the menu, click Add-ons. 5. In the Add-ons Manager tab, select the Extensions node. 6. In the Functional Testing Extension row, disable the Functional Testing 12.50 extension and enable the extension you added to Firefox.
<p>Add-in Dependencies</p>	<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 business component includes Java test objects.

Configuration	
<p>Options Dialog Box</p>	<p>Use the Java pane if your Oracle test or business component includes Java test objects. (Make sure that a GUI test is open and select Tools > Options > GUI Testing tab > Java node).</p> <p>See "Java Pane (Options Dialog Box > GUI Testing Tab)" on page 182.</p>
<p>Record and Run Settings Dialog Box (tests only)</p>	<p>Use the Oracle tab. (Record > Record and Run Settings)</p> <p>See "Oracle Tab (Record and Run Settings Dialog Box)" on page 225.</p>
<p>Test Settings Dialog Box (tests only)</p>	<ul style="list-style-type: none"> • Use the Web pane. (File > Settings > Web node) See "Web Pane (Test/Business Component Settings Dialog Box / Application Area - Additional Settings Pane)" on page 68. • Use the Java pane if your Oracle test or business component includes Java test objects. (File > Settings > Java node) See "Java Pane (Test/Business Component Settings Dialog Box / Application Area - Additional Settings Pane)" on page 189.
<p>Custom Active Screen Capture Settings Dialog Box (tests only)</p>	<p>Use the Oracle applications section. (Tools > Options > GUI Testing tab > Active Screen node > Custom Level)</p> <p>For details, see the section on the Custom Active Screen Capture Settings dialog box in the <i>HP Unified Functional Testing User Guide</i>.</p>
<p>Application Area Additional Settings pane (business components only)</p>	<ul style="list-style-type: none"> • Use the Web pane if your test includes Web test objects. In the application area, select Additional Settings > Web in the sidebar. For details, see "Web Pane (Test/Business Component Settings Dialog Box / Application Area - Additional Settings Pane)" on page 68. • Use the Java pane if your Oracle test or business component includes Java test objects.

	<p>In the application area, select Additional Settings > Java in the sidebar.</p> <p>See "Java Pane (Test/Business Component Settings Dialog Box / Application Area - Additional Settings Pane)" on page 189.</p> <p>(The options shown in the Java pane of the Test Settings dialog box are the same as the options that are available in the Additional Settings in the application area.)</p>
--	--

Oracle Add-in - Testing and Configuration

This chapter includes:

- [Considerations for Working with the Oracle Add-in](#) 218
- [Recording Tests on Oracle Applications](#) 219
- [Dynamic Transformation Support](#) 220
- [How to Verify or Enable the Oracle Server Unique Name Attributes](#) 221
- [How to Enable the Oracle Name Attribute](#) 221
- [How to Set Oracle Environment Variables](#) 222
- [How to Disable Dynamic Transformation Support](#) 223
- [Oracle Tab \(Record and Run Settings Dialog Box\)](#) 225
- [Oracle Record and Run Environment Variables](#) 227
- [Troubleshooting and Limitations - Oracle Add-in](#) 228

Considerations for Working with the Oracle Add-in

- 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 business component. Versions of JInitiator 1.1.x installed after you install the Oracle Add-in are not supported by UFT. 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 details, see the section on repairing your UFT installation in the *HP Unified Functional Testing Installation Guide*.

If you try to record an action on an Oracle object with an unsupported version of JInitiator 1.1.x, UFT records a generic `WinObject.Click` statement that includes the coordinates of the click and the mouse button that was clicked.

- 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).
```

For details, see "[Dynamic Transformation Support](#)" on page 220.

Note: The UFT 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. details, see "[How to Enable the Oracle Name Attribute](#)" on page 221.
- 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 UFT in most test object descriptions to identify Oracle objects. For details, see "[How to Enable the Oracle Name Attribute](#)" on page 221.
- In UFT, 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 need 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 details on UFT functionality, see the *HP Unified Functional Testing User Guide*.

Recording Tests on Oracle Applications

As you record on an Oracle Applications session, UFT inserts statements into your test or business component that represent the operations you perform. The UFT 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: UFT does not record the selection of Oracle tabs. Each object in an Oracle tab is included in the object repository within the tab hierarchy. UFT then uses this hierarchy when the test or business component is run, switching to the appropriate tab if needed.

The UFT 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`.

Example of a Step on an Oracle Object

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
 Responsibilities	Select	"Assets, Vision Operations (USA)"	Select the "Assets, Vision Operations (USA)" item from the "Responsibilities"

UFT records this step in the Editor as:

```
OracleListOfValues("Responsibilities").Select "Assets, Vision Operations (USA)"
```

Working with Tests

Each time you begin recording a test, you can use the ["Oracle Tab \(Record and Run Settings Dialog Box\)"](#) (described on page 225) to instruct UFT to connect to a specified Oracle Applications server. Alternatively, you can instruct UFT to record on any open browser. For details, see ["Recording Tests on Oracle Applications"](#) above.

Working with Components

The Record and Run Settings dialog box is used for tests only. When you record a business component on an Oracle Applications session, you cannot instruct UFT to open or connect to a specified Oracle

Applications server. You can open and connect to it manually or include statements in your business component (using the **OpenApp** operation or the `SystemUtil` utility object and the `OracleLogon` test object) that open and connect to the Oracle Applications server. For details on the Applications pane and Applications dialog box, see the *HP Unified Functional Testing User Guide*.

Working with Java Test Objects

You may have a combination of Oracle and Java test objects in your Oracle test or business component. This occurs when UFT 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 business components using the new Oracle test object model. For details on Java objects, methods, and properties, see the **Java** section of the *HP UFT Object Model Reference for GUI Testing*, installed together with the Oracle Add-in. For details on Oracle objects, methods, and properties, see the **Oracle** section of the *HP UFT Object Model Reference for GUI Testing*.

There are specific options and settings you can use in your test or business 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 > GUI Testing** tab > **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 business component.

Dynamic Transformation Support

The Oracle Add-in uses a mechanism for supporting multiple Java environments (Oracle 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 `-Xrunjvmsupport` option (for JInitiator 1.3.1.x and Sun Plug-in 1.4.1) or the `_classload_hook=jvmsupport` 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 details, see ["How to Disable Dynamic Transformation Support" on page 223](#).
- 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. For details, see ["How to Disable Dynamic Transformation Support" on page 223](#)

How to Verify or Enable the Oracle Server Unique Name Attributes

This task includes the following steps:

- ["Prerequisite" below](#)
- ["Enable the Oracle server to supply unique Name attributes" below](#)

1. Prerequisite

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.

2. Enable the Oracle server to supply unique Name attributes

- a. Add the following line to the server configuration file (for example, \$OA_HTML/bin/appsweb_UKTRN_hwu00001.cfg):

```
otherparams=record=names
```

- b. Restart the Oracle server.

How to Enable the Oracle Name Attribute

This task describes the different ways in which you can enable the **Name** attribute supplied by the Oracle Applications server before using the Oracle Add-in to test Oracle Applications.

This task includes:

- ["Enable the Name attribute when accessing the application directly" on the next page](#)
- ["Enable the Name attribute when using HTML to launch the Oracle application" on the next page](#)
- ["Enable the Name attribute when using the Personal Home Page to launch your Forms 6 application" on the next page](#)

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
```

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:\FNDS CSGN userid=XYZ fndnam=apps  
record=names">
```

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.

How to Set Oracle Environment Variables

This task describes how to set the environment variables you need for loading your Oracle application with UFT Oracle Add-in support. For all the environments, you need to set one or more environment variables with the path name of the Oracle Add-in support classes folder.

This task includes the following steps:

- ["Sun Plug-in 1.4.1 and Oracle Jinitiator 1.3.1.x" below](#)
- ["Oracle Jinitiator 1.1.x" below](#)

Sun Plug-in 1.4.1 and Oracle Jinitiator 1.3.1.x

Set the `_JAVA_OPTIONS` environment variable as follows:

```
-XrunjvmsHook  
-Xbootclasspath/a:"<UFT installation folder>\bin\java  
shared\classes";"<UFT installation folder>\bin\java_sharedclasses\jasmine.jar"
```

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

Note:

You can also use short paths in this command. For example:

```
-XrunjvmsHook -Xbootclasspath/a:C:\PROGRA~2\HP\UNIFIE~1\bin\JAVA_  
S~1\classes;C:\PROGRA~2\HP\UNIFIE~1\bin\JAVA_S~1\classes\jasmine.jar
```

In this example, UFT is installed in the default installation folder (C drive, Program Files) on a Windows 7 computer. `PROGRA~2` denotes the Program Files (x86) folder, which is the Program Files folder on 64-bit operating systems.

Oracle Jinitiator 1.1.x

Set the `_classload_hook` environment variable to `jvmsHook`.

How to Disable Dynamic Transformation Support

This task includes the following steps:

- ["Save the dynamically transformed classes" below](#)
- ["Disable dynamic transformation support " on the next page](#)

This task describes how to disable dynamic transformation support if the dynamic transformation support mechanism does not work properly.

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:

- a. Open the registry editor (select **Start > Run**, type `regedit` in the **Open** box and click **OK**)
- b. Navigate to the JavaAgent main key, located in: `HKEY_LOCAL_MACHINE\SOFTWARE\Mercury`

- Interactive\JavaAgent.
- c. 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`.
 - d. 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`).

Disable dynamic transformation support

For Sun Plug-in 1.4.1 or Jinitiator 1.3.1.x:

1. Remove the `-XrunjvmsHook` 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, such as `C:\JavaSupportClasses`, appended by the `Final` subfolder. For example, after your modification the `_JAVA_OPTIONS` environment variable might look like this:

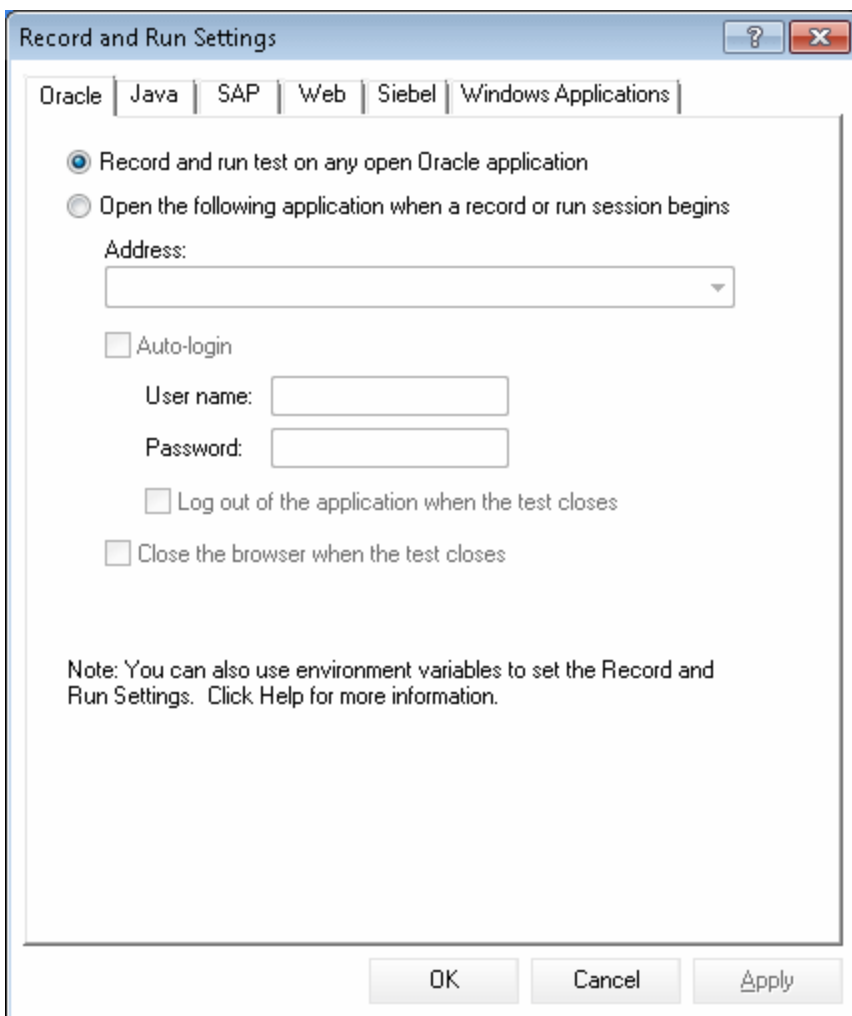
```
-Xbootclasspath/p:C:\JavaSupportClasses\Final -  
Xbootclasspath/a:"%ProgramFiles%\HP Software\Unified Functional  
Testing\bin\java_shared\classes";
```

For Initiator 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, such as `C:\JavaSupportClasses`, appended by the `Final` subfolder, to the Jinitiator 1.1.x classes folder. You can find the Jinitiator 1.1.x classes folder under `%ProgramFiles%\Oracle\JInitiator 1.1.x\classes`.

Oracle Tab (Record and Run Settings Dialog Box)

This tab 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 UFT to automatically log on to the server each time a record or run session begins (instead of recording the log-in steps).



To access	Select Record > Record and Run Settings and select the Oracle tab <div style="background-color: #f0f0f0; padding: 5px;"> Note: 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 Record > Record)). </div>
Important information	If you load only the UFT 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.

Relevant tasks	<ul style="list-style-type: none"> • "How to Verify or Enable the Oracle Server Unique Name Attributes" on page 221 • "How to Enable the Oracle Name Attribute" on page 221 • "How to Set Oracle Environment Variables" on page 222 • "How to Disable Dynamic Transformation Support" on page 223
See also	<ul style="list-style-type: none"> • "Recording Tests on Oracle Applications" on page 219 • "Dynamic Transformation Support" on page 220 • "Considerations for Working with the Oracle Add-in" on page 218

User interface elements are described below:

UI Elements	Description
Record and run test on any open Oracle Application	Instructs UFT to record and run the test on any open Oracle application.
Open the following application when a record or run session begins	<p>Instructs UFT to connect to the Oracle Applications server at the specified URL address.</p> <div style="background-color: #f0f0f0; padding: 5px;"> <p>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 UFT recognizes. Even if this radio button is selected and no application is specified, UFT can still record, recognize, and run on any open Oracle application.</p> </div>
Address	Indicates the URL of the Oracle Applications server to which you want to connect.
Auto-login	<p>Instructs UFT 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 UFT 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 UFT 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>

Oracle 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 \(Record and Run Settings Dialog Box\)"](#) (described on page 225). For details, see ["Recording Tests on Oracle Applications" on page 219](#).

Use the variable names listed in the table below to define Oracle record and run variables:

UI Elements	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 UFT to log on automatically to the Oracle Applications server. Possible values: <ul style="list-style-type: none"> • 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.
Log out of the application when the test closes	ORACLE_ LOGOUT_ENV	Instructs UFT to log out of the Oracle Applications session specified in "Oracle Tab (Record and Run Settings Dialog Box)" (described on page 225) when the test is closed. Possible values: True False
Close the browser when the test closes	ORACLE_ CLOSE_ BROWSER_ ENV	Instructs UFT to close the browser on which the test is recorded when the test is closed. Possible values: True False

For details on defining and working with environment variables, see the *HP Unified Functional Testing User Guide*.

Troubleshooting and Limitations - Oracle Add-in

This section contains general information and limitations about the Oracle add-in, and includes the following sections:

- ["Installing the Oracle Add-in " below](#)
- ["Creating and Running Testing Documents" on the next page](#)
- ["Record and Run Options" on the next page](#)
- ["Checkpoints" on the next page](#)

For limitations relevant to specific test objects and methods, see the **Oracle** section of the *HP UFT Object Model Reference for GUI Testing*.

Installing the Oracle Add-in

- If you install an Oracle JInitiator 1.1.x version after you install the UFT Oracle Add-in, you must repair UFT to test applications running in the newly installed JInitiator version. For details, see the *HP Unified Functional Testing Add-ins Guide*.

Note: It is not necessary to re-install or otherwise configure the UFT Oracle Add-in if you installed a new Oracle environment other than JInitiator 1.1.x.

- The UFT 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 details, see "How to Manage UFT 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 details, see: <ul style="list-style-type: none"> • "Recording Tests on Oracle Applications" on page 219. • " Dynamic Transformation Support " on page 220.
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, Windows\System32, or Windows\SysWOW64, depending on your operating system).
You cannot use UFT 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 UFT 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 details, see the section

Indicator	Solution
	on repairing your UFT installation in the <i>HP Unified Functional Testing 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.

Creating and Running Testing Documents

- 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 window to add these test objects directly from your Oracle Applications instead.
- 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.
- Simultaneous testing of multiple Oracle Applications sessions 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.

Part 9: PeopleSoft Add-in

This section includes:

["PeopleSoft Add-in - Quick Reference" on page 231](#)

["Considerations for Working with the PeopleSoft Add-in" on page 232](#)

["Troubleshooting and Limitations - PeopleSoft Add-in" on page 233](#)

PeopleSoft Add-in - Quick Reference

You can use the UFT PeopleSoft Add-in to test PeopleSoft user-interface objects (controls).

The following tables summarize basic information about the PeopleSoft Add-in and how it relates to some commonly-used aspects of UFT.

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 " Web-Based Application Support " on page 34.
Supported Environments	For details on supported PeopleSoft environments, see the PeopleSoft Add-in section of the <i>HP Unified Functional Testing Product Availability Matrix</i> , available from the HP Support Matrix page (requires an HP passport).
Important Information	See " Considerations for Working with the PeopleSoft Add-in " on the next page.
Test Object Methods and Properties	The PeopleSoft Add-in provides test objects, methods, and properties that can be used when testing objects in PeopleSoft applications. For details, see the PeopleSoft section of the <i>HP UFT Object Model Reference for GUI Testing</i> .
Checkpoints and Output Values	<ul style="list-style-type: none"> See the sections describing checkpoints and output values in the <i>HP Unified Functional Testing User Guide</i>. See "GUI Checkpoints and Output Values Per Add-in" on page 512
Troubleshooting and Limitations	See " Troubleshooting and Limitations - PeopleSoft Add-in " on page 233.

Prerequisites	
Opening Your Application	You must open UFT before opening your PeopleSoft application.
Testing in Mozilla Firefox	<p>To test your Web application in Mozilla Firefox version 33 and later, you must enable the Functional Testing Extension in one of the following ways:</p> <ul style="list-style-type: none"> If the Select Your Add-ons screen is displayed when you open Firefox, select to enable the Functional Testing Extension. If the Install Add-on tab opens and displays Functional Testing Extension when opening Firefox, select the Allow this installation check box and click Continue. <p>Otherwise:</p> <ol style="list-style-type: none"> In Firefox, open the browser menu. In the menu, click Add-ons. In the Add-ons Manager tab, select the Extensions node. In the Functional Testing Extension row, click the Enable button.

	<p>To test your Web application in Mozilla Firefox versions earlier than 33, you must enable the legacy Firefox agent:</p> <ol style="list-style-type: none"> 1. Open the <UFT installation folder>\Installations\Firefox folder. 2. From the Firefox folder, drag the Legacy.xpi file into Firefox. 3. In Firefox, open the browser menu. 4. In the menu, click Add-ons. 5. In the Add-ons Manager tab, select the Extensions node. 6. In the Functional Testing Extension row, disable the Functional Testing 12.50 extension and enable the extension you added to Firefox.
Add-in Dependencies	The Web Add-in must be loaded.

Configuration	
Options Dialog Box	<p>Use the Web pane. (Make sure that a GUI test is open and select Tools > Options > GUI Testing tab > Web > General node.) For details, see "Web > General Pane (Options Dialog Box)" on page 58.</p>
Record and Run Settings Dialog Box (tests only)	<p>Use the Web tab. (Record > Record and Run Settings) See "Web Tab (Record and Run Settings Dialog Box)" on page 71.</p>
Test Settings Dialog Box (tests only)	<p>Use the Web pane. (File > Settings > Web node) See "Web Pane (Test/Business Component Settings Dialog Box / Application Area - Additional Settings Pane)" on page 68.</p>
Custom Active Screen Capture Settings Dialog Box (tests only)	<p>Use the Web section. (Tools > Options > GUI Testing tab > Active Screen node > Custom Level) See the section on the Custom Active Screen Capture Settings dialog box in the <i>HP Unified Functional Testing User Guide</i>.</p>
Application Area Additional Settings pane (business components only)	<p>Use the Web pane. In the application area, select Additional Settings > Web in the sidebar. See "Web Pane (Test/Business Component Settings Dialog Box / Application Area - Additional Settings Pane)" on page 68.</p>

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:

- WebElement
- WebTable
- Images of 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 details on PeopleSoft and Web test objects, methods, and properties, see the **PeopleSoft** and **Web** sections of the *HP UFT Object Model Reference for GUI Testing*.

- For the purposes of Web event recording, UFT 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 details on Web event recording configurations, see "[Event Recording Configuration for Web-Based Applications](#)" on page 39.

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, UFT 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 10: PowerBuilder Add-in

This section includes:

"PowerBuilder Add-in - Quick Reference " on page 235

"Considerations for Working with the PowerBuilder Add-in " on page 236

"Troubleshooting and Limitations - PowerBuilder Add-in" on page 237

PowerBuilder Add-in - Quick Reference

You can use the UFT PowerBuilder Add-in to test PowerBuilder user-interface objects (controls).

The following tables summarize basic information about PowerBuilder Add-in and how it relates to some commonly-used aspects of UFT.

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 "Windows-Based Application Support" on page 81 .
Supported Environments	For details on supported PowerBuilder environments, see the PowerBuilder Add-in section of the <i>HP Unified Functional Testing Product Availability Matrix</i> , available from the HP Support Matrix page (requires an HP passport).
Important Information	See "Considerations for Working with the PowerBuilder Add-in " on the next page .
Test Object Methods and Properties	The PowerBuilder Add-in provides test objects, methods, and properties that can be used when testing objects in PowerBuilder applications. For details, see the PowerBuilder section of the <i>HP UFT Object Model Reference for GUI Testing</i> .
Checkpoints and Output Values	<ul style="list-style-type: none"> See the sections describing checkpoints and output values in the <i>HP Unified Functional Testing User Guide</i>. See "Considerations for Working with the PowerBuilder Add-in " on the next page. See "GUI Checkpoints and Output Values Per Add-in" on page 512
Troubleshooting and Limitations	See "Troubleshooting and Limitations - PowerBuilder Add-in" on page 237 .

Prerequisites	
Opening Your Application	You can open your PowerBuilder application before or after opening UFT.

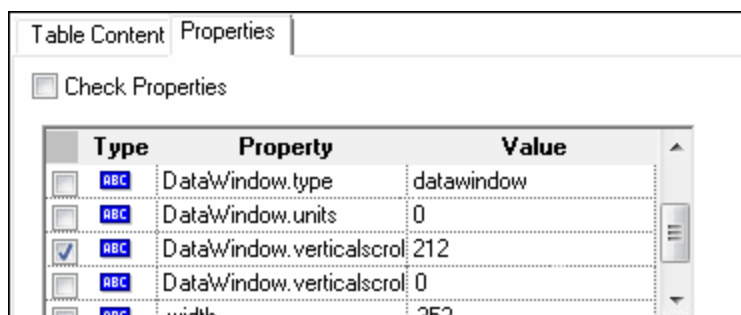
Configuration	
Options Dialog Box	Use the Windows Applications pane. (Tools > Options > GUI Testing tab > Windows Applications node) See "Windows Applications > General Pane (Options Dialog Box > GUI Testing Tab)" on page 85 .
Record and Run Settings Dialog Box (tests only)	Use the Windows Applications tab. (Record > Record and Run Settings) See "Windows Applications Tab (Record and Run Settings Dialog Box)" on page 93 . 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.

<p>Custom Active Screen Capture Settings Dialog Box (tests only)</p>	<p>Use the Windows applications section. (Tools > Options > GUI Testing tab > Active Screen node > Custom Level)</p> <p>See the section on the Custom Active Screen Capture Settings dialog box in the <i>HP Unified Functional Testing User Guide</i>.</p>
<p>Application Area Additional Settings pane (business components only)</p>	<p>Use the Applications pane.</p> <p>In the application area, select Additional Settings > Applications in the sidebar.</p> <p>See the section on the Applications pane in the <i>HP Unified Functional Testing User Guide</i>.</p>

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, UFT treats it as a table and opens the Table Checkpoint Properties or Table Output Value Properties dialog box (not supported for business 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 details on the `DataWindow` test object, see the **PowerBuilder** section of the *HP UFT Object Model Reference for GUI Testing*.

Troubleshooting and Limitations - PowerBuilder Add-in

- When learning or recording on toolbars in PowerBuilder applications, UFT no longer records the `PbToolbar` test object. Instead, it records a `PbObject.Click` step. The `PbToolbar` test object is no longer available in UFT 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. To fix this, update object repositories and tests to use the `PbObject` test object for toolbar steps.
- By default, UFT recognizes objects in your application as PowerBuilder objects only if the application was built with a supported version of PowerBuilder.

If you want to try to use UFT with an unsupported PowerBuilder version, you can make the following change:

- a. Open **C:\Windows\wrun.ini** in a text editor. If this file does not exist, create it.
- b. Under the `[WrCfg]` section, add a line in the format: `PBRuntimeDllName=<YourPB_Dll>`. For example:

```
[WrCfg]
PBRuntimeDllName=pbvm126.dll
```

Caution: This is an 'As-Is', unsupported option.

For details on supported versions of PowerBuilder, see the UFT Product Availability Matrix, available from the [HP Support Matrix page](#) site.

Part 11: Qt Add-in

This section includes:

["Qt Add-in - Quick Reference" on page 239](#)

["Considerations - Qt Add-in" on page 241](#)

Qt Add-in - Quick Reference

You can use the Qt testing support provided by UFT to test user-interface objects (controls) developed using the Qt framework for mobile devices.

The following tables summarize basic information about Qt testing support and how it relates to some commonly-used aspects of UFT.

General Information	
Add-in Type	The Qt Add-in is a Windows-based add-in. Much of its functionality is the same as other Windows-based add-ins. See "Windows-Based Application Support" on page 81
Important Information	"Considerations - Qt Add-in" on page 241
Test Object Methods and Properties	The Qt Add-in uses a sub-set of the standard Windows test objects, methods, and properties, which can be used when testing objects (controls) in Qt applications. For details, see the Standard Windows section of the <i>HP UFT Object Model Reference for GUI Testing</i> . For a list of supported Qt controls, see "Considerations - Qt Add-in" on page 241 .
Checkpoints and Output Values	<ul style="list-style-type: none"> See the sections describing checkpoints and output values in the <i>HP Unified Functional Testing User Guide</i>. See "GUI Checkpoints and Output Values Per Add-in" on page 512
Prerequisites	
Opening Your Application	You must open UFT before opening your Qt application.
Add-in Dependencies	None
Low Level Recording	To enable low level recording on Qt controls, you must first modify the object identification properties list for the WinObject test object class, as follows: <ul style="list-style-type: none"> Add the regexpwndtitle property to the mandatory properties list. Move the object class property up the assistive properties list so that it is learned before the text property. For details, see the section describing the Object Identification Dialog Box in the <i>HP Unified Functional Testing User Guide</i> .
Configuration	
Options Dialog Box	Use the Windows Applications pane. (Tools > Options > GUI Testing tab > Windows Applications node) See "Windows Applications > General Pane (Options Dialog Box > GUI Testing Tab)" on page 85 .

Record and Run Settings Dialog Box (tests only)	Use the Windows Applications tab. (Run > Run Settings or Record > Record Settings) See " Windows Applications Tab (Record and Run Settings Dialog Box) " on page 93. Note: UFT recognizes Qt objects only in applications that are opened after changing 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 > GUI Testing tab > Active Screen node > Custom Level) See the section on the Custom Active Screen Capture Settings dialog box in the <i>HP Unified Functional Testing User Guide</i> .
Application Area Additional Settings tab (business components only)	Use the Applications pane. In the application area, click Additional Settings in the sidebar and select the Java node See the section on the Applications Pane in the <i>HP Unified Functional Testing User Guide</i> .

Considerations - Qt Add-in

- Qt Add-in support is provided using standard Windows test objects. The following table lists each supported Qt control and its corresponding standard Windows test object.

Qt Control	Standard Windows Test Object
QCheckBox	WinCheckBox
QComboBox	WinComboBox
QComboBoxPrivateContainer	Window
QExpandingLineEdit	WinEdit
QLabel	Static
QLineEdit	WinEdit
QListWidget	WinList
QMenu	Window
QMenuBar	WinToolBar
QPlainTextEdit	WinEditor
QPushButton	WinButton
QRadioButton	WinRadioButton
QSpinBox	WinSpin
QTabWidget	WinTab
QToolButton	WinButton
QTreeWidget	WinTreeView

- UFT supports only Visual Studio as the compiler for QT applications.

Part 12: Add-in for SAP Solutions

This section includes:

["Web-based SAP Support" on page 244](#)

["Windows-based SAP Support" on page 257](#)

Add-in for SAP Solutions - Overview

You can use the UFT Add-in for SAP Solutions to test user-interface objects (controls) in SAP GUI for Windows applications and in Web-based SAP applications. You can create and run tests and business components on these objects, and check their properties.

After you create your test or business component, you can enhance it by adding checkpoints, retrieving output values, and parameterizing values. Where relevant, you can also add SAP GUI for Windows or SAP Web objects, methods and properties to it.

This section contains:

- ["Web-based SAP Support" on page 244](#)
- ["Windows-based SAP Support" on page 257](#)

Web-based SAP Support

This chapter includes:

- [Web-Based SAP Support - Quick Reference](#)245
- [Considerations for Working with SAP GUI for HTML](#)247
- [Web > Page/Frame Options Pane \(Options Dialog Box > GUI Testing Tab\)](#)249
- [Web > Advanced Pane \(Options Dialog Box > GUI Testing Tab\)](#)251
- [Troubleshooting and Limitations - Web-based SAP](#)253

Web-Based SAP Support - Quick Reference

You can use the SAP Web testing support provided with the UFT Add-in for SAP Solutions to test user-interface objects in Web-based SAP applications. These applications include SAP Enterprise Portal, Internet Transaction Server, SAP Customer Relationship Management (CRM), and the Interaction Centre Web Client.

The following tables summarize basic information about the Web-based SAP environment and how it relates to some commonly-used aspects of UFT.

General Information	
Add-in Type	<p>Web-based SAP testing support is similar to other Web-based add-ins.</p> <p>For details, see "Web-Based Application Support" on page 34.</p>
Supported Environments	<p>For details on supported Web-based SAP environments, see the Add-in for SAP Solutions section of the <i>HP Unified Functional Testing Product Availability Matrix</i>, available from the HP Support Matrix page (requires an HP passport).</p> <p>Note: Firefox is supported for testing SAPUI5 Desktop applications, but not other Web-based SAP environments.</p>
Important Information	<ul style="list-style-type: none"> • 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 UFT to record, specify any environment variables, and select the required Web options to optimize performance. For details, see "SAP Tab (Record and Run Settings Dialog Box)" on page 271. • When the UFT Add-in for SAP Solutions is loaded, UFT can learn objects and run steps on both Web-based (requires the Web Add-in, too) and Windows-based SAP applications. <ul style="list-style-type: none"> • For details on recording and running tests and business components on SAP GUI for Windows applications, see "Low-Level or Analog Mode Recording on SAP GUI for Windows" on page 288 and "How to Record on Standard Windows Controls During an SAP GUI for Windows Recording Session" on page 289. • For details on working with SAP GUI from HTML, see "Considerations for Working with SAP GUI for HTML" on page 247. • The SAPUI5 Add-in and SAPWDJ Add-ins are installed when you install the Add-in for SAP Solutions. However, in order for full support for SAPUI5 and SAP WebDynpro (WDJ) applications, you must do the following: <ul style="list-style-type: none"> • Install the Add-in for SAP Solutions and the Web Add-in • When starting UFT, in the Add-ins Manager, select the SAPUI5 and/or the SAPWDJ add-in under the SAP add-in and the Web Add-in. • The SAPWDJ Add-in is supported only on Internet Explorer.
Test Object Methods and Properties	<p>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 details, see the SAP Web section of the <i>HP UFT Object Model Reference for GUI Testing</i>.</p> <p>Note: If you working with SAPUI5 applications, see the SAPUI5 section of the <i>HP UFT Object Model</i></p>

	<i>Reference for GUI Testing</i> , in the Web Child Add-ins section.
Checkpoints and Output Values	<ul style="list-style-type: none"> See the sections describing checkpoints and output values in the <i>HP Unified Functional Testing User Guide</i>. See "Considerations for Working with SAP GUI for HTML" on the next page. See "GUI Checkpoints and Output Values Per Add-in" on page 512
Troubleshooting and Limitations	See "Troubleshooting and Limitations - Web-based SAP" on page 253.
Prerequisites	
Opening Your Application	<ul style="list-style-type: none"> Open UFT before you open your Web-based SAP application. If you are working in an SAP GUI application that contains HTML objects, you can log on to your application before opening UFT, but you must open UFT before navigating to the transaction containing any HTML objects. For SAP GUI for HTML, Interaction Centre Web Client (ICWC) applications, and Customer Relationship Management (CRM) applications, confirm that your SAP server and client are configured properly. See "How to Enable Support for SAP GUI for Windows" on page 264.
Testing in Mozilla Firefox	<p>To test your Web application in Mozilla Firefox version 33 and later, you must enable the Functional Testing Extension in one of the following ways:</p> <ul style="list-style-type: none"> If the Select Your Add-ons screen is displayed when you open Firefox, select to enable the Functional Testing Extension. If the Install Add-on tab opens and displays Functional Testing Extension when opening Firefox, select the Allow this installation check box and click Continue. <p>Otherwise:</p> <ol style="list-style-type: none"> In Firefox, open the browser menu. In the menu, click Add-ons. In the Add-ons Manager tab, select the Extensions node. In the Functional Testing Extension row, click the Enable button. <p>To test your Web application in Mozilla Firefox versions earlier than 33, you must enable the legacy Firefox agent:</p> <ol style="list-style-type: none"> Open the <UFT installation folder>\Installations\Firefox folder. From the Firefox folder, drag the Legacy.xpi file into Firefox. In Firefox, open the browser menu. In the menu, click Add-ons. In the Add-ons Manager tab, select the Extensions node. In the Functional Testing Extension row, disable the Functional Testing 12.50 extension and enable the extension you added to Firefox.
Add-in Dependencies	The Web Add-in must be loaded.

Configuration	
Options Dialog Box	Use the Web pane. (Make sure that a GUI test is open and select Tools > Options > GUI Testing tab > Web > General node.)
Record and Run Settings Dialog Box (tests only)	<ul style="list-style-type: none"> Use the SAP tab (Record > 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 "How to Enable Support for SAP GUI for Windows" on page 264. Use the Web tab (Record > Record Settings) to instruct UFT to use a specific URL and browser to open a Web-based SAP application, or the SAP Enterprise Portal, at the beginning of each record and run session. Alternatively, you can instruct UFT to record and run on any open browser. See "Web Tab (Record and Run Settings Dialog Box)" on page 71.
Test Settings Dialog Box (tests only)	Use the Web pane. (File > Settings > Web node) See "Web Pane (Test/Business Component Settings Dialog Box / Application Area - Additional Settings Pane)" on page 68.
Custom Active Screen Capture Settings Dialog Box (tests only)	Use the Web section. (Tools > Options > GUI Testing tab > Active Screen node > Custom Level button) See the section on the Custom Active Screen Capture Settings dialog box in the <i>HP Unified Functional Testing User Guide</i> .
Application Area Additional Settings pane (business components only)	Use the Web pane. In the application area, select Additional Settings > Web in the sidebar. See "Web Pane (Test/Business Component Settings Dialog Box / Application Area - Additional Settings Pane)" on page 68.
Web Event Recording Configuration Dialog Box	When you load the Add-in for SAP Solutions, the settings in the "Web Event Recording Configuration Dialog Box" on page 445 (Record > Web Event Recording Configuration) are automatically customized. You do not need to make any Web event configuration changes.

Considerations for Working with SAP GUI for HTML

- You can add a table checkpoint while recording or editing your test.
- 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 if the Active Screen **Capture level** was set to **Complete** when the object was captured, and the **Active Screen** is currently selected (**View > Active Screen**).

(You set the **Capture level** in the **Active Screen** pane of the Options dialog box (**Tools > Options > GUI Testing** tab > **Active Screen** node). For details, see the section on Active Screen options in the

HP Unified Functional Testing User Guide.)

- You can spool all of the available data from an SAP GUI for HTML application table into an external file using the `GetCellData` method, which loops through each cell in the table. You can then save the information to an external file.

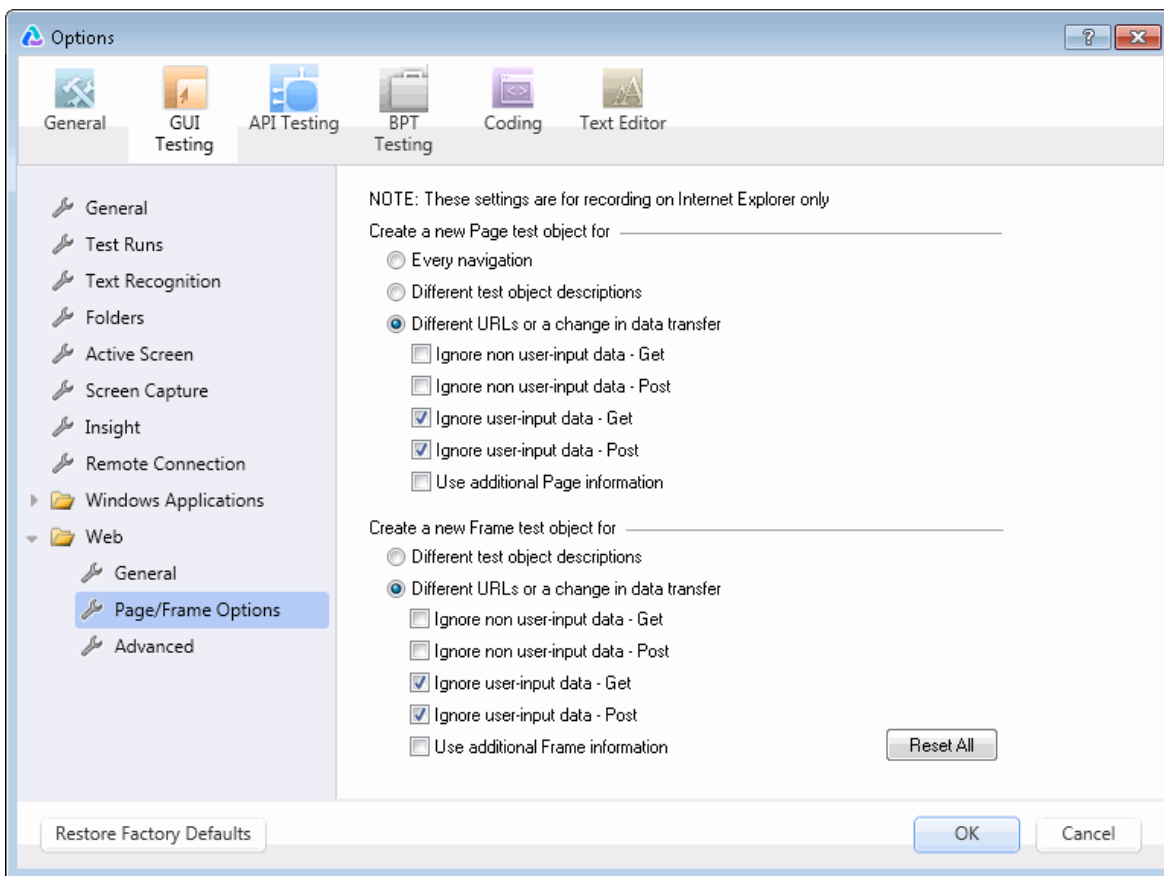
The following example uses the **GetCellData** method to list the data of each cell in a table of 10 rows and 10 columns:

```
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
Next
```

For details on the **GetCellData** method, see the **SAP Web** section of the *HP UFT Object Model Reference for GUI Testing*.

Web > Page/Frame Options Pane (Options Dialog Box > GUI Testing Tab)

This pane enables you to configure the settings that are best-suited for testing Page and Frame test objects in Web-based SAP applications. Applying these recommended settings helps to optimize UFT performance.



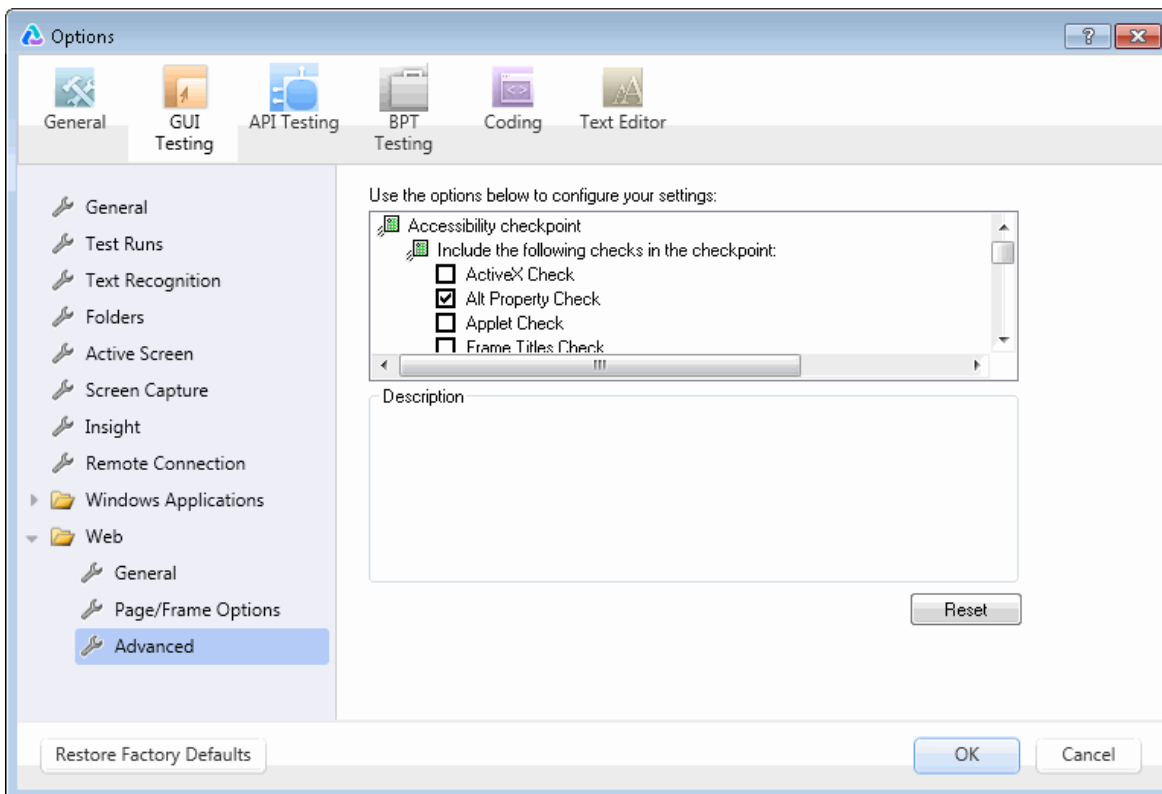
To access	<ol style="list-style-type: none"> Do one of the following: <ul style="list-style-type: none"> Ensure that a GUI test, action, or component is in focus in the document pane. In the Solution Explorer, select a GUI test or component node or one of its child nodes. Select Tools > Options > GUI Testing tab > Web node > Page/Frame Options node.
See also	<ul style="list-style-type: none"> "Web-Based SAP Support - Quick Reference" on page 245 "Web > Advanced Pane (Options Dialog Box > GUI Testing Tab)" on page 251

Recommended settings are described below:

UI Elements	Description
<p>Create a new Page test object for</p>	<p>Different URLs or a change in data transfer</p> <p>Instructs UFT 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 • Ignore additional Page information
<p>Create a new Frame test object for</p>	<p>Different URLs or a change in data transfer</p> <p>Note: This option is selected by default.</p> <p>Instructs UFT 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>

Web > Advanced Pane (Options Dialog Box > GUI Testing Tab)

This pane enables you to configure the advanced settings that are best-suited for testing Web-based SAP applications. Applying these recommended settings helps to optimize UFT performance.



To access	<ol style="list-style-type: none"> 1. Do one of the following: <ul style="list-style-type: none"> • Ensure that a GUI test, action, or component is in focus in the document pane. • In the Solution Explorer, select a GUI test or component node or one of its child nodes. 2. Select Tools > Options > GUI Testing tab > Web node > Advanced node.
See also	<ul style="list-style-type: none"> • "Web-Based SAP Support - Quick Reference" on page 245 • "Web > Page/Frame Options Pane (Options Dialog Box > GUI Testing Tab)" on page 249

Recommended settings are described below:

UI Elements	Description
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

UI Elements	Description
	<ul style="list-style-type: none"><li data-bbox="430 289 574 317">• OnMouseUp <p data-bbox="430 331 1247 359">This instructs UFT to use these standard Windows mouse events instead of browser events.</p>

Troubleshooting and Limitations - Web-based SAP

This section contains general troubleshooting and limitation information about the Web-based SAP add-in, and includes the following sections:

- ["General" below](#)
- ["SAPUI5 Objects" below](#)
- ["SAP WDJ Objects" on the next page](#)
- ["SAP Enterprise Portal" on the next page](#)
- ["SAP GUI for HTML—Internet Transaction Server \(ITS\) " on page 255](#)
- ["Using the Active Screen " on page 255](#)
- ["Web-based CRM Systems" on page 256](#)

General

- It is not recommended to work with other Web-based add-ins when the UFT Add-in for SAP Solutions is loaded. The Add-in for SAP Solutions modifies certain Web configuration settings that may affect other add-ins or applications.
- When recording and running steps on a table control, only the table content that is visible on the client is actually available.
- During a run session, the SAP platform response time may be slower than the time it takes for UFT to run the corresponding step.

Workaround: Add a **Wait** statement prior to the relevant step.

- In some cases, when running tests on SAPEdit, SAPNavigationBar, or SAPPortal, you may receive a **Cannot find object** error.

Workaround: Do one of the following:


- Ensure that the object properties are unique and correct.
- Modify the registry as specified below:

For 32 bit computers: In the **HKEY_LOCAL_MACHINE\Software\Mercury Interactive\QuickTestProfessional\MicTest\AbortIfHangInSendData** key, set the value of **DWORD** to **0**.

For 64-bit computers: In the **HKEY_LOCAL_MACHINE\Software\Wow6432Node\Mercury Interactive\QuickTestProfessional\MicTest\AbortIfHangInSendData** key, set the value of **DWORD** to **0**.

SAPUI5 Objects

- Because the SAPUI5 add-in supports both desktop and mobile applications, not all methods for all objects are supported for both desktop and mobile applications.

- When running a test or component on SAPUI5 test objects, application Alert messages for some objects are not displayed.
Workaround: Do one of the following:
 - Make sure your test does not contain steps that need to be performed on the Alert message.
 - Add an **If** statement to your test for the object that triggers the alert to check if the Alert exists.
- If your SAPUI5 application contains nested table objects, inserting a checkpoint on the nested table object shows only the parent SAPUITable object.
Workaround: Select the nested WebTable objects and create checkpoints for the WebTable object instead of the parent SAPUITable object.
- When using the Object Spy to view an SAPUIMenu object in Firefox, you cannot add the menu object to the repository with the **Add Object to Repository** button .
- When working with mobile SAPUI5 applications, UFT cannot use the Object Spy on some disabled controls (like the SAPUIButton and SAPUITextEdit) due to the application object properties.

SAP WDJ Objects

- If you try to record objects in a SAP WDJ application without the SAPWDJ add-in loaded, UFT records only some of the objects correctly.
Workaround: Restart UFT and load the SAP WDJ Add-in.
- During a run session, the response time of the SAP platform running a SAP Web Dynpro Java (WDJ) application may be slower than the time it takes for UFT to run the corresponding step.
Workaround: Add a **Wait** statement prior to the relevant step or set the Delay each step execution option in the **Test Runs** pane of the Options dialog box (**Tools > Options > GUI Testing** tab > **Test Runs** node) to the necessary wait period (in milliseconds)
- The following functionalities are not supported for SAP WDJ applications:
 - Selection of non-visible table rows
 - Interaction with the application scroll bar
 - Menus inside table cells
 - Date navigator windows inside the application
- When running tests of your SAPWDJ application, the **SAPWDJTable.SelectItemInCell** method does not click a link (for a pop-up window) in a table cell.
Workaround: Add a **Link.Click** step to click the link inside the table cell.
- When recording steps on an SAPWDJTable object, the SelectCell method is not always recorded if you are selecting non-editable cells.

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, UFT is unable to correctly identify certain objects.

Workaround: Clear the **Enable Smart Identification** check box for the Browser test objects in the Object Repository window. 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 UFT will be able to recognize it during the test run.

SAP GUI for HTML—Internet Transaction Server (ITS)

- When using the Object Spy or creating a checkpoint on an object inside an SAP Web table cell, UFT 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 Editor 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.

- When recording a SAPList object, you need to click the input part of the list, not its button part in order to enable UFT to recognize the object.

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 (**File > Settings > Web**).

- 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.

Web-based CRM Systems

- When using UFT to test Web-based CRM systems, make sure that the CRM system is in test mode. You can do this by adding "?sap-testmode=X" to the URL.

Windows-based SAP Support

This chapter includes:

- Windows-based SAP Support - Quick Reference 259
- Windows-based SAP Support - Testing and Configuration 261
 - SAP GUI Scripting API and UFT 262
 - How to Enable Support for SAP GUI for Windows 264
 - Considerations - Windows-based SAP Add-in for SAP Solutions 269
 - Environment Variables for Windows-based SAP Applications 270
 - Package and Patch Versions Requirements - SAP Application Server and SAP GUI for Windows .. 270
 - SAP Tab (Record and Run Settings Dialog Box) 271
 - SAP > General Pane (Options Dialog Box > GUI Testing Tab) 274
 - Troubleshooting and Limitations - Windows-based SAP 276
- Windows-Based SAP Support - Test Objects 281
 - Checkpoints and Output Values in SAP GUI for Windows 282
 - Using the Auto-Parameterize Option to Parameterize Table and Grid Cell Values 282
 - Low-Level or Analog Mode Recording on SAP GUI for Windows 288
 - Spooling Data from a Table 288
 - How to Record on Standard Windows Controls During an SAP GUI for Windows Recording Session 289
- UFT-SAP Solution Manager Integration 290
 - UFT-SAP Solution Manager Integration - Overview 291
 - Solution Manager Testing Modes: Standalone or Integrated 292
 - SAP Structured Parameters 294
 - How to Configure Solution Manager to Work with UFT 295
 - How to Work with Tests in Solution Manager in Standalone Mode 297
 - How to Run a Test Stored in Solution Manager 301
 - How to Display or Edit a GUI Test from Solution Manager in Integrated Mode 303
 - How to Transfer Data To and From GUI Tests in Integrated Mode Using Test Parameters 304
 - How to Work with SAP Structured Parameters 305
 - Solution Manager Connection Dialog Box 309
 - Save GUI Test to Solution Manager Dialog Box 310

- [Open GUI Test from Solution Manager Dialog Box](#)311
- [Upload File to Solution Manager Dialog Box](#) 312
- [Save External File to Solution Manager Dialog Box](#)313
- [Download File from Solution Manager](#) 314
- [SAP > SAP Solution Manager Pane \(Options Dialog Box > GUI Testing Tab\)](#) 315
- [Value Map Dialog Box](#) 317

Windows-based SAP Support - Quick Reference

You can use the Windows-based SAP testing support provided with the UFT Add-in for SAP Solutions to test user-interface objects in SAP GUI for Windows user-interface objects.

The following tables summarize basic information about the Windows-based SAP environment and how it relates to some commonly-used aspects of UFT.

General Information	
Add-in Type	When testing SAP GUI for Windows applications, much of the functionality is the same as other Windows-based add-ins. See "Windows-Based Application Support" on page 81 .
Supported Environments	For details on supported Windows-based SAP environments, see the Add-in for SAP Solutions section of the <i>HP Unified Functional Testing Product Availability Matrix</i> , available from the HP Support Matrix page (requires an HP passport)..
Important Information	<ul style="list-style-type: none"> When the Add-in for SAP Solutions is loaded, UFT can learn objects and run steps on both Web-based and Windows-based SAP applications. For details on recording and running tests and business components on Web-based SAP applications, see "Web-based SAP Support" on page 244. The SAPNWBC Desktop Add-in is installed with the Add-in for SAP solutions. However, in order for full support for NWBC Desktop applications, you must do the following: <ul style="list-style-type: none"> Install both the Add-in for SAP Solutions and the WPF Add-in When opening UFT, in the Add-in Manager, select the SAPNWBC Desktop Add-in under the SAP Add-in and the WPF Add-in.
Test Object Methods and Properties	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 details, see the SAP GUI for Windows section of the <i>HP UFT Object Model Reference for GUI Testing</i> .
Checkpoints and Output Values	<ul style="list-style-type: none"> See the sections describing checkpoints and output values in the <i>HP Unified Functional Testing User Guide</i>. See "Checkpoints and Output Values in SAP GUI for Windows" on page 282. See "Supported Checkpoints " on page 513 and "Supported Output Values " on page 515 Per Add-In. See "GUI Checkpoints and Output Values Per Add-in" on page 512
Troubleshooting and Limitations	See "Troubleshooting and Limitations - Windows-based SAP" on page 276

Prerequisites	
Before Using this Add-in	For details on the following prerequisites, see "How to Enable Support for SAP GUI for Windows" on page 264 . <ul style="list-style-type: none"> The SAP GUI Scripting option must be installed.

	<ul style="list-style-type: none"> Your server and client must have the proper package and patch versions installed. For details, see "Package and Patch Versions Requirements - SAP Application Server and SAP GUI for Windows" on page 270. See also: SAP OSS note # 480149 The Scripting API must be enabled on both the server and client. For details, see "Enable scripting on the SAP application (server-side)" on page 264. Your client must be configured to use the Dialog display mode for F4 Help screens. Make sure that the server is not set to use a Low speed connection. The F1 and F4 Help display setting must be configured correctly to support testing the use of the F1 and F4 Help screens in your SAP GUI for Windows application. If you plan to use the UFT-Solution Manager integration features, you must also install the appropriate support package and configure the Solution Manager server to work with UFT. For details, see "How to Configure Solution Manager to Work with UFT" on page 295.
Add-in Dependencies	None

Configuration	
Options Dialog Box	Use the SAP > General pane. (Tools > Options > GUI Testing tab > SAP > General node) See "SAP > General Pane (Options Dialog Box > GUI Testing Tab) " on page 274.
Record and Run Settings Dialog Box (tests only)	Use the SAP tab. (Record > Record and Run Settings) See "SAP Tab (Record and Run Settings Dialog Box)" on page 271.
Custom Active Screen Capture Settings Dialog Box (tests only)	Use the SAP GUI for Windows section. (Tools > Options > GUI Testing tab > Active Screen node > Custom Level button) See the section on the Custom Active Screen Capture Settings dialog box in the <i>HP Unified Functional Testing User Guide</i> .
Application Area Additional Settings pane (business components only)	Use the Applications pane. In the application area, select Additional Settings > Applications in the sidebar. See the section on the Applications pane in the <i>HP Unified Functional Testing User Guide</i> .

Windows-based SAP Support - Testing and Configuration

This section includes

- [SAP GUI Scripting API and UFT](#)262
- [How to Enable Support for SAP GUI for Windows](#) 264
 - [How to Enable Scripting on the SAP Application \(Server-Side\)](#)266
- [Considerations - Windows-based SAP Add-in for SAP Solutions](#)269
- [Environment Variables for Windows-based SAP Applications](#)270
- [Package and Patch Versions Requirements - SAP Application Server and SAP GUI for Windows](#)270
- [SAP Tab \(Record and Run Settings Dialog Box\)](#)271
- [SAP > General Pane \(Options Dialog Box > GUI Testing Tab\)](#)274
- [Troubleshooting and Limitations - Windows-based SAP](#)276

SAP GUI Scripting API and UFT

UFT works directly with the SAP GUI Scripting API to record your operations. Therefore, UFT adds steps to your test or business component only when API events are sent to the server. This means that while recording a test or business 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, UFT inserts steps with the relevant Windows-based SAP objects in the Keyword View (tests and business components) and adds corresponding statements in the Editor (tests only).

Note: The UFT 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 details, see "[Low-Level or Analog Mode Recording on SAP GUI for Windows](#)" on page 288.

For details on the test object model, the object repository, and the Object Spy, see the *HP Unified Functional Testing User Guide*.

Example 1: Check Boxes

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.

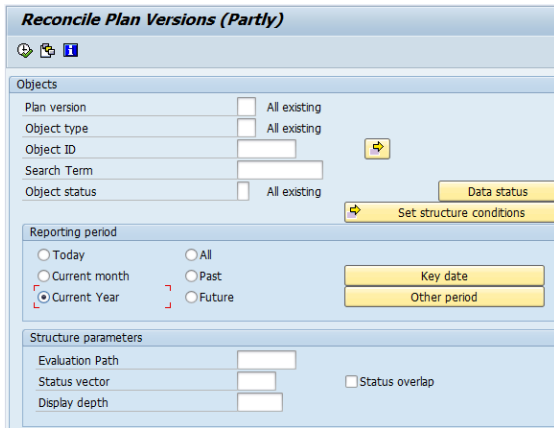
Price Simulation for Material			
<input checked="" type="checkbox"/> Incl. cash discount	Set	"ON"	Set the state of the "Incl. cash discount" check box to "ON".
<input checked="" type="checkbox"/> Delivery costs	Set	"ON"	Set the state of the "Delivery costs" check box to "ON".
<input checked="" type="checkbox"/> Effective price	Set	"ON"	Set the state of the "Effective price" check box to "ON".
<input checked="" type="checkbox"/> Effective price	SetFocus		Set the focus on the "Effective price" check box.
<input type="checkbox"/> Continue (Enter)	Click		Click the "Continue (Enter)" button.

UFT records these steps in the Editor 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: Radio Buttons

Suppose you select a radio button to change the reporting period in the **Reconcile Plan Versions** transaction of your SAP GUI for Windows application. This radio button is labeled **Current Year**.



UFT uses the SAP GUI business component type (41) to identify the object as a SAPGuiRadioButton object. It creates a SAPGuiRadioButton test object with the name **Current Year** and records the following properties and values as the description for the radio button.

Type	Property	Value
GUI	attachedtext	Current Year
GUI	enabled	True
GUI	guicomponenttype	41
GUI	height	20
GUI	name	PCHZTR_Y

Note: The **guicomponenttype** and **name** property values are supplied by the SAP GUI Scripting API.

UFT also records that you performed a **Set** method to turn ON the radio button.

UFT displays your step in the Keyword View as follows:

Item	Operation	Value	Documentation
Session			
Reconcile Plan Versions			
Current Year	Set		Select the "Current Year" radio button.

UFT displays your step in the Editor like this:

```
SAPGuiSession("Session").SAPGuiWindow("Reconcile Plan Versions").SAPGuiRadioButton("Current Year").Set
```

When you run a test or business component, UFT 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, UFT looks up the description for the SAPGuiRadioButton object with the name **Current Year** by searching the object repository. UFT finds the following description:

```
guicomponenttype:=41  
name:=PCHZTR_Y  
attachedtext:=Current Year
```

UFT 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).

How to Enable Support for SAP GUI for Windows

This task describes how to enable SAP GUI for Windows to work with UFT.

1. Prerequisite: Make sure that SAP GUI Scripting is installed

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 enable 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 details on the various security options, see the online SAP GUI Scripting Security Guide at the SAP Service Marketplace.

2. Enable scripting on the SAP application (server-side)

- a. Confirm that you have the proper support package and kernel patch levels installed. For details, see ["Package and Patch Versions Requirements - SAP Application Server and SAP GUI for Windows" on page 270](#).
- b. Enable scripting on your SAP application. (By default, scripting is disabled.) You do this by entering the Maintain Profile Parameters window with administrative permissions and setting the `sapgui/user_scripting` profile parameter to `TRUE` on the application server.
 - o To enable scripting for all users, set this parameter on all application servers.
 - o To enable scripting for a specific group of users, set the parameter only on application servers with the appropriate access restriction settings.

For more details, see ["How to Enable Scripting on the SAP Application \(Server-Side\)" on page 266](#).

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.

3. Enable scripting on the SAP application (client-side)

You can do this on your SAP client only if the **SAP GUI Scripting** option is installed. If this option is not installed, reinstall your SAP GUI for Windows application and be sure to select the **SAP GUI Scripting** check box. For details, see your SAP GUI for Windows documentation.

4. Eliminate warning messages

By default, you regularly receive two warning messages when using UFT with an SAP GUI for Windows application:

- When UFT connects to the Scripting API, the following warning message is displayed: A script is trying to attach to the GUI.
- When UFT 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 UFT.

5. Check the connection speed on the SAP server

Confirm that the **Low speed connection** option is NOT selected for the server to which you are connecting before recording and running GUI tests.

This is because 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 UFT to properly record and run tests. (UFT displays an error message if the **Low speed connection** option is selected.)

For details, see SAP OSS note #587202.

6. Set F1 Help to use the modal dialog box mode

Confirm that the modal dialog box option is selected. This enables UFT to record the display of **F1** Help in your tests. (The **F1** Help in your SAP GUI for Windows application can be displayed using either the Performance Assistant or as a modal dialog box.)

7. Set F4 Help to use the dialog display mode

Confirm that your client is set to load **F4** Help screens in Dialog mode. (The SAP GUI for Windows application cannot load **F4** Help screens in Control mode when using the SAP GUI Scripting API (Enable Scripting option).)

Note: This is a per-user setting. You must set this option on each client that you want to test using the UFT Add-in for SAP Solutions. Alternatively, the SAP system administrator can change the system default for you.

How to Enable Scripting on the SAP Application (Server-Side)

UFT records and runs steps based directly on the API events that are sent from the client to the SAP server because UFT communicates directly with the SAP GUI Scripting API. Therefore, to record and run tests and business components on your SAP GUI for Windows application, you must enable scripting on both the server and client computers.

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.

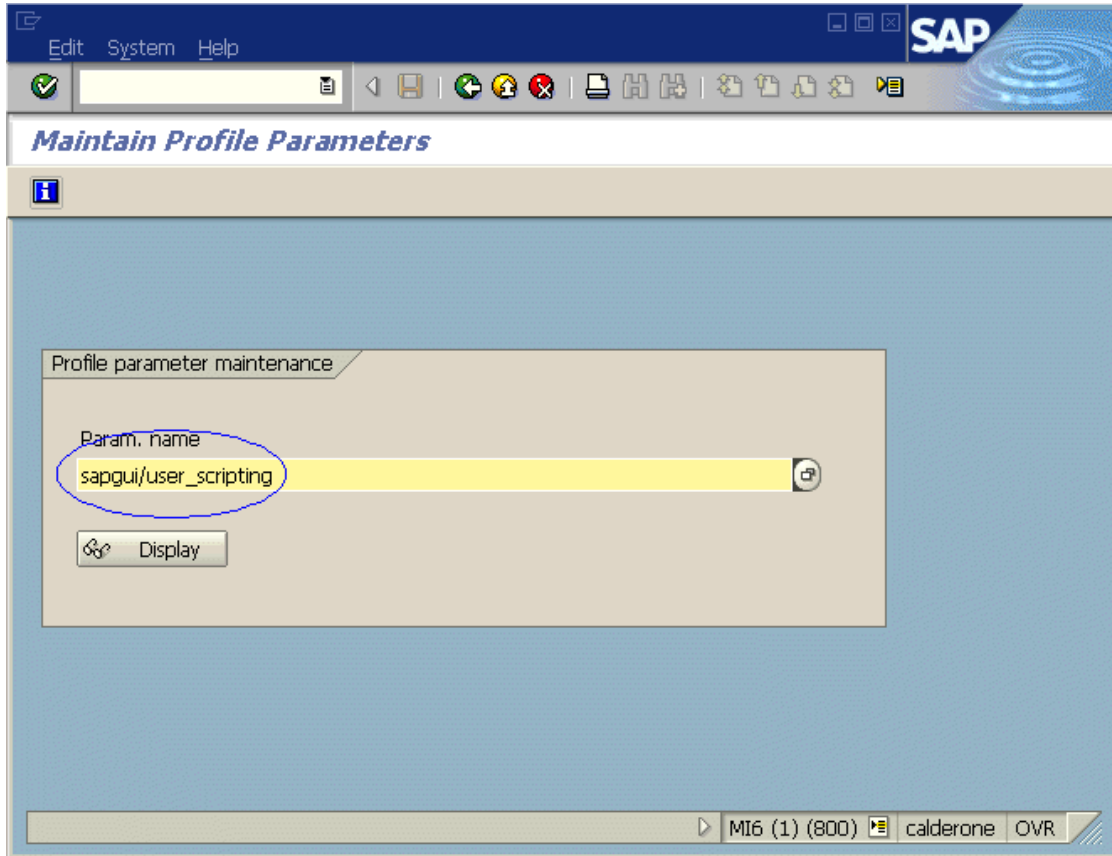
You enable scripting by entering the Maintain Profile Parameters window with administrative permissions and setting the `sapgui/user_scripting` profile parameter to `TRUE` on the application server.

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.

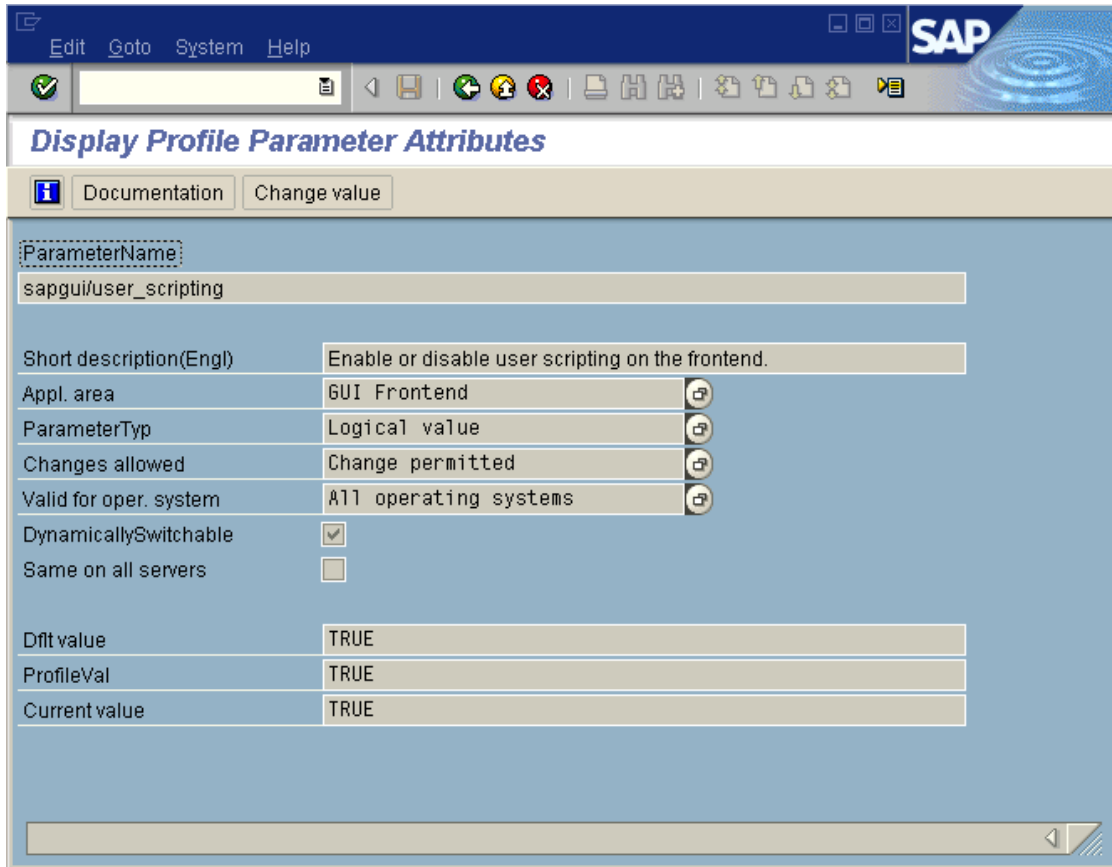
To change the profile parameter:

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.

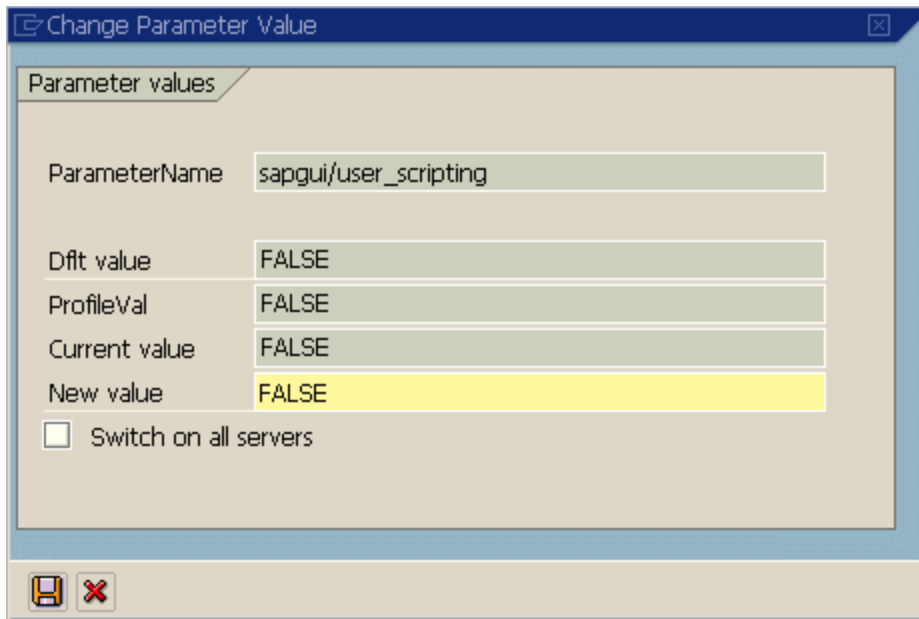



Note: If the message **Parameter name is unknown** is displayed in the status bar, your client lacks the required support package (see "[Package and Patch Versions Requirements - SAP Application Server and SAP GUI for Windows](#)" on page 270). Download and install the support package that corresponds to the SAP release you are using and then begin this task again.

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 capital letters) in the **New value** box and click **Save**  .

Note: The change takes effect only when you log on to the system. Therefore, before beginning to work with UFT, 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 details, see SAP OSS note # 480149.

Considerations - Windows-based SAP Add-in for SAP Solutions

When recording and running tests or business components on SAP GUI for Windows applications, consider the following:

- When working in tests, the Record and Run Settings dialog box in UFT 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 business 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 business 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 details, see ["How to Record on Standard Windows Controls During an SAP GUI for Windows Recording Session"](#) on page 289.
- As you record a test or business component on your SAP GUI for Windows application, UFT records the operations you perform. UFT works directly with the SAP GUI Scripting API to record your operations. Therefore, although UFT records a step for each operation you perform, it adds the steps to your test only when API events are sent to UFT (when information is sent to the SAP server).

For details on the SAP GUI Scripting API events, see your SAP documentation.

- When you select a test step in UFT, 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 details on Active Screen capture levels, see the section on the Custom Active Screen Capture Settings Dialog Box in the *HP Unified Functional Testing User Guide*.

Environment Variables for Windows-based SAP Applications

You can use environment variables to specify details for the applications you want to use during a recording or run session. These variables can also be used in external library files for automation scripts.

If you define any of these environment variables, they override the values in the **Server description**, **User**, **Password**, **Client**, and **Language** boxes in the "SAP Tab (Record and Run Settings Dialog Box)" (described on page 271).

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 details on defining and working with environment variables, see the *HP Unified Functional Testing User Guide*.

Package and Patch Versions Requirements - SAP Application Server and SAP GUI for Windows

To test your application using UFT, you must confirm that you have the correct support package and kernel patch levels for your software component release. The following tables show the **minimum** required versions and levels. You must have these versions and levels or higher:

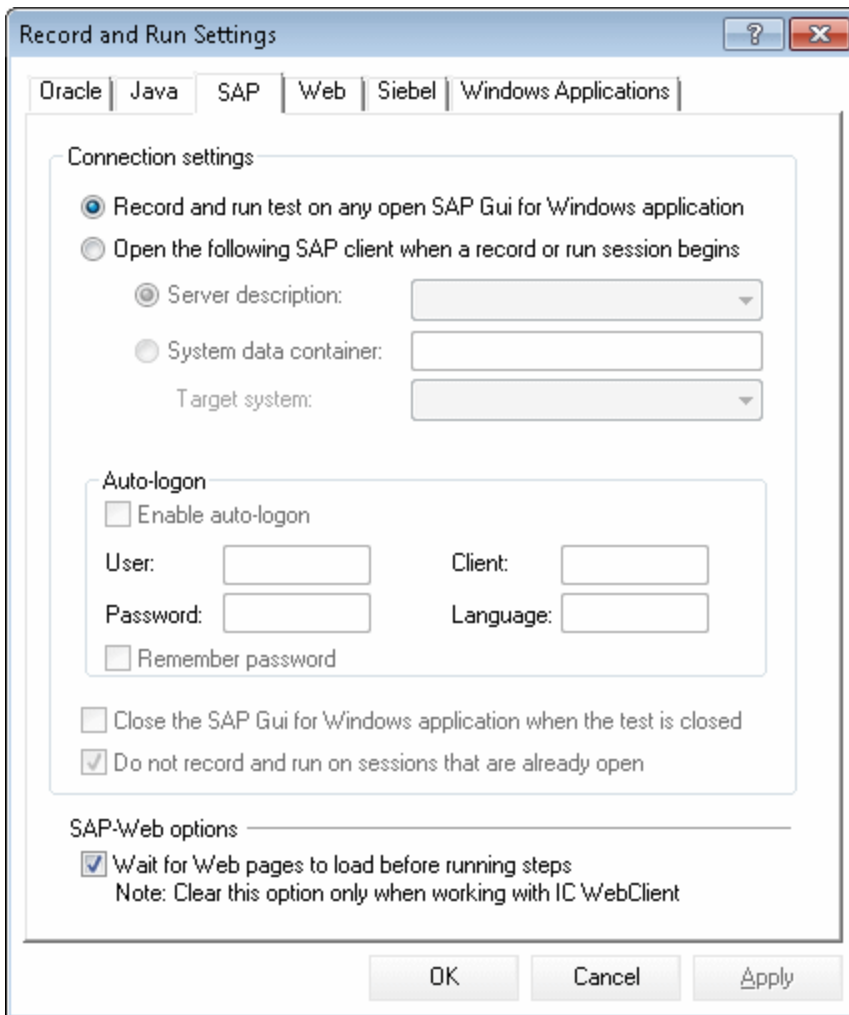
Software Component	Release	Support Package	Kernel Patch Level
SAP_APPL	311	SAPKH31196	Kernel 3.11 level 650
SAP_APPL	40B	SAPKH40B71	Kernel 4.0B level 903
SAP_APPL	45B	SAPKH45B49	Kernel 4.5B level 753

Software Component	Release	Support Package	Kernel Patch Level
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

For details, see SAP OSS note # 480149.

SAP Tab (Record and Run Settings Dialog Box)

This dialog box enables you to specify how UFT connects to your SAP GUI for Windows application during a recording or run session.



To access Select **Record > Record and Run Settings** and choose the **SAP** tab.

	<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>
See also	<ul style="list-style-type: none"> You can also use application details environment variables to specify these parameters. For details, see "Environment Variables for Windows-based SAP Applications" on page 270. "SAP GUI Scripting API and UFT" on page 262. "Using the Auto-Parameterize Option to Parameterize Table and Grid Cell Values" on page 282. "Low-Level or Analog Mode Recording on SAP GUI for Windows" on page 288. "How to Record on Standard Windows Controls During an SAP GUI for Windows Recording Session" on page 289. "SAP > General Pane (Options Dialog Box > GUI Testing Tab)" on page 274.

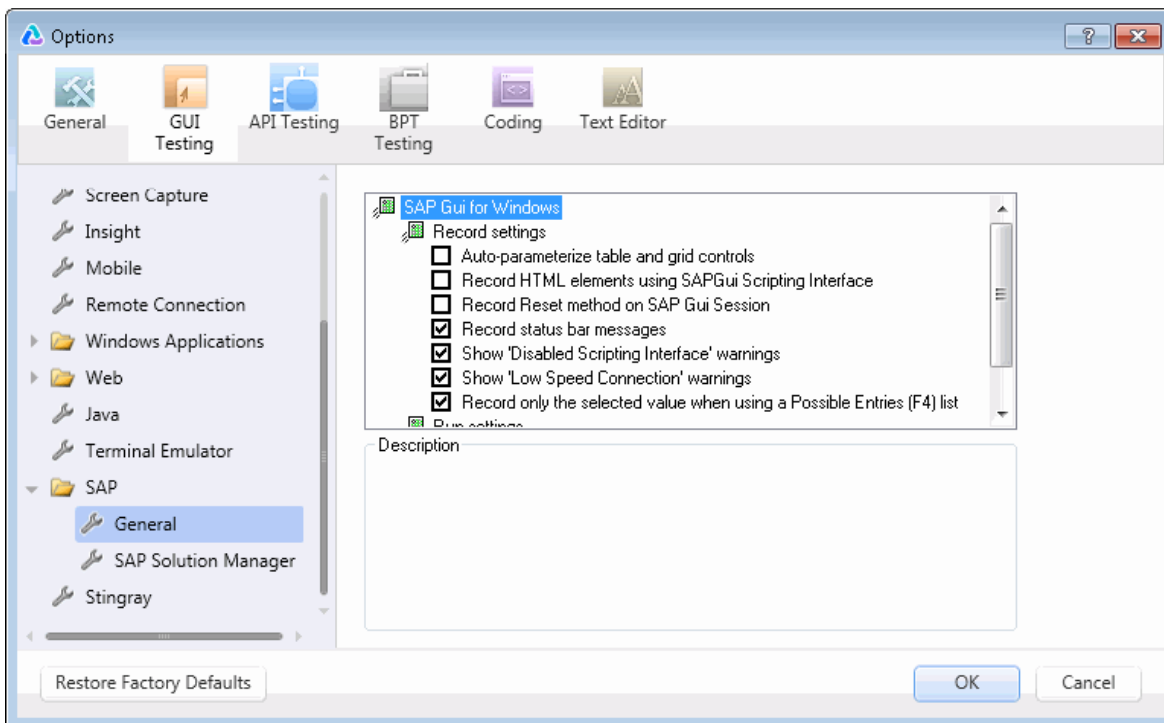
User interface elements are described below:

UI Elements	Description
Record and run tests on any open SAP GUI for Windows application	<p>Instructs UFT 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>
Open the following SAP client when a record or run session begins	<p>Instructs UFT to connect to the specified SAP server and open your SAP GUI for Windows application using specified user settings.</p> <p>Note: If you select to connect to a specific server, you can specify details that will enable UFT to automatically log on to the server each time a record or run session begins (instead of recording the logon steps).</p>
Server description	<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> <p>Note: The values you enter for the User, Client, Password, and Language in the Auto-logon area of the dialog box are saved with the selected server (in addition to being saved with the test). Therefore, the saved values are automatically displayed in the auto-logon area when you specify a previously selected server.</p>
System data container	<p>Specifies the system data container that contains the available target systems for the current test.</p> <p>Enabled only if you open a GUI test from Solution Manager in integrated mode.</p>
Target system	<p>A drop-down list of available SAP environments that can be tested.</p> <p>Enabled only if the System data container option is selected.</p>
Enable auto-logon	<p>Instructs UFT 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>

UI Elements	Description
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>
Close the SAP GUI for Windows application when the test is closed	<p>Instructs UFT 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 > GUI Testing tab > SAP node) overrides this option. For details, see "SAP > General Pane (Options Dialog Box > GUI Testing Tab) " on the next page.</p>
Do not record and run on sessions that are already open	<p>Instructs UFT 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>
Wait for Web pages to load before running steps	<p>Instructs UFT to wait for Web-based SAP pages to synchronize completely before starting the test run.</p> <p>Note: This option is selected by default. Clear this option only when working with IC WebClient.</p>

SAP > General Pane (Options Dialog Box > GUI Testing Tab)

This pane enables you to configure how UFT records and runs tests and business components on SAP applications.



<p>To access</p>	<ol style="list-style-type: none"> Do one of the following: <ul style="list-style-type: none"> Ensure that a GUI test, action, or component is in focus in the document pane. In the Solution Explorer, select a GUI test or component node or one of its child nodes. Select Tools > Options > GUI Testing tab > SAP > General node.
<p>Important information</p>	<ul style="list-style-type: none"> This pane is available only when the UFT Add-in for SAP Solutions is installed and loaded. The Restore Factory Defaults button resets all Options dialog box options to their defaults.
<p>See also</p>	<ul style="list-style-type: none"> "SAP GUI Scripting API and UFT" on page 262. "Using the Auto-Parameterize Option to Parameterize Table and Grid Cell Values" on page 282. "How to Record on Standard Windows Controls During an SAP GUI for Windows Recording Session" on page 289. "Environment Variables for Windows-based SAP Applications" on page 270/ "SAP Tab (Record and Run Settings Dialog Box)" on page 271.

User interface elements are described below:

UI Elements	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 pane. UFT inserts an <code>Input</code> 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 details, see "Using the Auto-Parameterize Option to Parameterize Table and Grid Cell Values" on page 282.
Record HTML elements using SAPGui Scripting interface	Specifies whether UFT 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 business component.
Record Reset Method on SAP Gui Session	Records a Reset method as the first step when recording a test or business component. When the test or business 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 business 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-logout options are both selected in the "SAP Tab (Record and Run Settings Dialog Box)" (described on page 271).
Record status bar messages	Records a <code>SAPGuiStatusBar.Sync</code> step each time a status bar message is displayed in the SAP application. Note: This option is selected by default.
Show 'Disabled Scripting Interface' warnings	Instructs UFT 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 details on enabling the SAP GUI Scripting API, see "How to Enable Support for SAP GUI for Windows" on page 264. Note: This option is selected by default.
Show 'Low Speed Connection' Warnings	Instructs UFT to display warnings if the connection speed to the server is set to Low speed connection . If this option is selected, one of the following occurs if the connection speed is low: <ul style="list-style-type: none"> • If the session was opened by UFT 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 details on Standard Windows recording mode, see "How to Record on Standard Windows Controls During an SAP GUI for Windows Recording Session" on page 289. If this option is not selected, one of the following occurs if the connection speed is low: <ul style="list-style-type: none"> • If the session was opened by UFT 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 details on Standard Windows recording mode, see "How to Record on Standard Windows Controls During an SAP GUI for Windows Recording Session" on page 289.

UI Elements	Description
	<p>The connection speed can be checked using the SAP client. For details, see "Check the connection speed on the SAP server" on page 265.</p> <p>Note: This option is selected by default.</p>
Record only the selected value when using a Possible Entries (F4) list	<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>
Session cleanup	<p>Instructs UFT to close all SAP GUI for Windows sessions opened by UFT 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 (Record and Run Settings Dialog Box)" (described on page 271).</p>

Troubleshooting and Limitations - Windows-based SAP

This section contains general troubleshooting and limitation information about the Windows-based SAP add-in, and includes the following sections:

- ["Creating and Running Testing Documents" below](#)
- ["Working with Windows-based SAP Controls" on the next page](#)
- ["Checkpoints, Output Values, and the Object Spy" on page 278](#)
- ["Test Objects, Methods, and Properties" on page 279](#)
- ["SAP Scripting API" on page 279](#)
- ["SAP Scripting API" on page 279](#)
- ["SAP Structured Parameters" on page 280](#)

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 UFT 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 (**Tools > Options > GUI Testing tab > SAP > General** node). Then close and reopen the test and then begin recording again.
- The SAP Gantt chart (SAP Bar Chart) and Image/Picture controls are supported by the SAP GUI for Windows alternative recording mechanism. The current support for these controls is limited. You can override the default recording behavior for SAP Windows test objects, or add limited recording support for other SAP GUI for Windows objects.
- If you insert a call to an external action or a copy of an action, and that action includes an `SAPGuiTable.Input`, `SAPGuiGrid.Input`, or `SAPGuiAPOGrid.Input` statement, the corresponding input data sheet is not copied to the Data pane 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.
- UFT can connect 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 computer, UFT connects to the latest process that was launched.
- Use the SAP Tab of the Record and Run Settings Dialog Box to instruct UFT to open your SAP GUI for Windows application. Do not use the Windows Applications tab of the dialog box for this purpose.

Working with Windows-based SAP 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 window. 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.
- UFT fails to run steps on SAP tree nodes that contain the ";" character.
- UFT 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: Do one of the following:

- Change to Standard Windows Recording mode (select Standard Windows Recording from the **Recording Modes** drop-down in the Record toolbar) to record on these objects. (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.)
- Use low-level recording to record on these objects.
- Use programmatic descriptions to run steps on these objects.

Checkpoints, Output Values, and the Object Spy

- To ensure that the correct object properties are captured with your checkpoint, always record a step that results in communication with the server (such as pressing **ENTER**) *before* inserting a checkpoint or output value.
- You cannot use the Object Spy or create checkpoints for the controls listed below. However, you can successfully record and run steps on them.
 - Toolbar buttons in grid controls.
 - Internal controls in tree or table objects.
(For example, a radio button in a table cell or a check box in a tree.)
- Creating checkpoints or using the Object Spy on an object that is located in a currently inactive SAP screen (for example, if the screen is behind an invoked dialog box) is not supported. However, you can create checkpoints on status bar messages (displayed in an inactive window) using the **Record status bar messages** option (**Tools > Options > GUI Testing** tab > **SAP** node > **Record status bar messages**).
- When running old 6.20 tests on a 6.40 client, checkpoints on radio buttons, check boxes, edit boxes, or regular buttons may fail due to changes to tooltip property values for these objects in the 6.40 client.
- UFT can estimate the number of rows in a table control, but it cannot retrieve the exact number because only the table content that is visible on the client is actually available. Data from non-visible rows are stored only on the back-end server. Therefore, when inserting or modifying checkpoints for a table control object, the number of rows specified in the Define/Modify Row Range dialog box may not be accurate.

- Do not perform any operations on the SAP GUI window (such as changing the transaction state or navigating to another window) while UFT retrieves the data for a table checkpoint even if it seems to take a long time, as this may cause severe problems.
- When inserting a checkpoint on a table or grid from the Active Screen, the actual table must be open in your SAP GUI for Windows application to extract the correct information from the table or grid.

Test Objects, Methods, and Properties

- When using the `SAPGuiTableInput` method, check the scrolling mode of the current table. If you parameterize a table with a Data pane sheet that contains more rows in the sheet than are displayed in the table's current view, UFT tries to scroll down the table while running the test, to insert more rows from the data sheet. UFT 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 details, see the *HP Unified Functional Testing 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 UFT 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, UFT captures one Active Screen image for several steps. UFT 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, UFT records a `Set` statement using asterisks (`****`) as the method argument value.

Workaround: Do one of the following:

- Configure and enable the **Auto-logon** settings in the SAP Tab of the Record and Run Settings Dialog Box.
- Insert a step using one of the SAPGuiUtil object's AutoLogon methods.
- 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 details, see the **SAP Windows** section of the *HP UFT Object Model Reference for GUI Testing*(**Help > HP Unified Functional Testing Help > Object Model Reference for GUI Testing > SAP Windows**).

SAP Structured Parameters

When you launch UFT by clicking the **Edit Test Script** button directly in the SAP Solution Manager Test Automation:Initial Screen transaction(Transaction Code: stce) or in the SAP Solution Manager Configuration transaction (Transaction Code :Solar02), clicking the **Maintain SAP Parameter** button in UFT might not return you to the correct page in SAP Solution Manager.

Workaround: Use the **External Test** button in the Change Test Configuration transaction to launch UFT.

When you create a test in the Change Test Script transaction of SAP Solution Manager and then click the **Back** button, UFT may not show the test.

Workaround: Save and close UFT and click the **External Test** button to call UFT.

When you click the **Maintain SAP Parameter** button or the **Back/External Test** button to switch from UFT to SAP, you may receive an error in SAP Solution Manager: OBJECT_OBJREF_NOT_ASSIGNED.

Windows-Based SAP Support - Test Objects

This chapter includes:

- [Checkpoints and Output Values in SAP GUI for Windows](#) 282
- [Using the Auto-Parameterize Option to Parameterize Table and Grid Cell Values](#) 282
- [Low-Level or Analog Mode Recording on SAP GUI for Windows](#) 288
- [Spooling Data from a Table](#) 288
- [How to Record on Standard Windows Controls During an SAP GUI for Windows Recording Session](#) . 289

Checkpoints and Output Values in SAP GUI for Windows

- 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.

- To insert a new table checkpoint while editing your test or business component, the actual table or grid must be open to the appropriate level or view. 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 ActiveX grid controls (these generally have toolbars), UFT 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.
- The **Visible Rows** option in the Define/Modify Row Range Dialog Box is not available for checkpoints created on grid controls.

Using the Auto-Parameterize Option to Parameterize Table and Grid Cell Values

When working with tests, UFT 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, UFT 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 pane. UFT 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.

For details, see:

- [How UFT Records in Auto-Parameterize Mode](#)283
- [Parameterized Cell Values in the Input Data Sheet](#)285
- [Considerations for Auto-Parameterization](#) 286
- [Data in Rows that Require Scrolling](#) 287

How UFT 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, UFT:

- 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, UFT 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.`

	Connection_Number	Airline
1		AA
2		17 AB
3		64
4	.END.	

- Inserts an Input `<DataSheetName>` statement (followed by a `SelectCell` statement) into your test.

<ul style="list-style-type: none"> □ SAP_2 └─ □ Table control tc spfli └─ □ Fixed_cols └─ □ SAP_2 	<table border="0"> <tr><td>Resize</td><td>141,29</td></tr> <tr><td>Input</td><td>"Action1.Table control tc spfli"</td></tr> <tr><td>SetFocus</td><td></td></tr> <tr><td>SendKey</td><td>ENTER</td></tr> </table>	Resize	141,29	Input	"Action1.Table control tc spfli"	SetFocus		SendKey	ENTER	<table border="0"> <tr><td>Resize the "SAP_2" window to 141 by 29 characters.</td></tr> <tr><td>Add a sheet named "Action1.Table control tc spfli" to the Data</td></tr> <tr><td>Set the focus on the "Fixed_cols" edit box.</td></tr> <tr><td>Press the ENTER keyboard key.</td></tr> </table>	Resize the "SAP_2" window to 141 by 29 characters.	Add a sheet named "Action1.Table control tc spfli" to the Data	Set the focus on the "Fixed_cols" edit box.	Press the ENTER keyboard key.
Resize	141,29													
Input	"Action1.Table control tc spfli"													
SetFocus														
SendKey	ENTER													
Resize the "SAP_2" window to 141 by 29 characters.														
Add a sheet named "Action1.Table control tc spfli" to the Data														
Set the focus on the "Fixed_cols" edit box.														
Press the ENTER keyboard key.														

UFT records these steps in the Editor as follows:

```
SAPGuiSession("Session").SAPGuiWindow("SAP_2").Resize 141,29  
SAPGuiSession("Session").SAPGuiWindow("SAP_2").SAPGuiTable("Table control tc  
spfli").Input "Action1.Table control tc spfli"  
SAPGuiSession("Session").SAPGuiWindow("SAP_2").SAPGuiEdit("Fixed_  
cols").SetFocus  
SAPGuiSession("Session").SAPGuiWindow("SAP_2").SendKey ENTER
```

The Input statement instructs UFT 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 pane.

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:

ID	No.	Flight Date	Airfare	Curr.	Plane Type	Capacity	Occupied
A	17	11.02.2009	422,94	USD	747-400	385	370
AA	17	11.03.2009	422,94	USD	747-400	385	365
AA	17	08.04.2009	422,94	USD	747-400	385	374
AA	17	06.05.2009	422,94	USD	747-400	385	368
AA	17	03.06.2009	422,94	USD	747-400	385	373
AA	17	01.07.2009	422,94	USD	747-400	385	372
AA	17	29.07.2009	422,94	USD	747-400	385	372
AA	17	26.08.2009	422,94	USD	747-400	385	371
AA	17	23.09.2009	422,94	USD	747-400	385	370
AA	17	21.10.2009	422,94	USD	747-400	385	210
AA	17	18.11.2009	422,94	USD	747-400	385	73
AA	17	16.12.2009	422,94	USD	747-400	385	76
AA	17	13.01.2010	422,94	USD	747-400	385	55
AA	17	10.02.2010	422,94	USD	747-400	385	16
AA	17	10.03.2010	422,94	USD	747-400	385	0
AA	64	13.02.2009	422,94	USD	A310-300	280	265
AA	64	13.03.2009	422,94	USD	A310-300	280	266
AA	64	10.04.2009	422,94	USD	A310-300	280	269
AA	64	08.05.2009	422,94	USD	A310-300	280	271
AA	64	05.06.2009	422,94	USD	A310-300	280	262
AA	64	03.07.2009	422,94	USD	A310-300	280	271

UFT 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 details, see ["Data in Rows that Require Scrolling" on page 287](#).

The corresponding data sheet in your Data pane looks like this:

Data					
H5					
	Flight_Number	Depart.city	Dep._airport	Arrival_city	Dest._airport
1	56		JFK	SAN FRANCISCO	SFO
2	64	NEW YORK			
3	77	NEW YORK	JFK		FRA
4	END				
5					
6					
7					

< Global \ Action1 \ Action1.GridViewCtrl \ Action1.Table control tc spfli

There are three rows in the data sheet, because data was modified in the first three rows of the table or grid in the application. Note that the data sheet does not contain columns for the **Airfare** and **Plane Type** columns, because no values were modified in those columns during the recording session.

Parameterized 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, UFT 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

Data					
B15					
	Flight_Number	Depart.city	Dep._airport	Arrival_city	Dest_airport
1	56		JFK	SAN FRANCISCO	SFO
2	64	NEW YORK			
3	77	NEW YORK	JFK		FRA
4	END				
5	56				SFO
6		NEW YORK			
7	77		ORD		FRA
8	END				
9	6665				FRA
10					
11			SFO		JFK
12	END				
13					
14					
15					
16					
17					
18					
19					
20					
21					
22					
23					
24					

Global / Action1 / Action1.GridViewCtrl / Action1.Table control tc spfli

Considerations for Auto-Parameterization

Consider the following when using the **Auto-parameterize table and grid controls** option:

- UFT 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. For best results:
 - 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.

- 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 details, see ["Data in Rows that Require Scrolling" below](#).

- When recording, UFT 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, UFT 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 pane 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 Call 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.

Data in Rows that Require Scrolling

When working in tests, UFT 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 UFT 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, UFT must send a command to add the row. By default, UFT sends a `PageDown` command if the rows in the data sheet exceed those currently displayed in the application. If UFT 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  
("SAPMV45ATCTRL_V_ERF_").Input "Action1.All items", ENTER
```

Low-Level or Analog Mode Recording on SAP GUI for Windows

When working in tests, if you are unable to record steps 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 steps on those objects using low-level or analog recording (select **Low-Level Recording** or **Analog Recording** from the **Recording Mode** drop-down in the Record toolbar 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 details on low-level and analog recording, see the *HP Unified Functional Testing User Guide*.

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
```


next

For details on the **GetCellData** method, see the **SAP GUI for Windows** section of the *HP UFT Object Model Reference for GUI Testing*.

How to Record on Standard Windows Controls During an SAP GUI for Windows Recording Session

To enable UFT to record steps on standard Windows controls during an SAP GUI for Windows recording session, you must switch to **Standard Windows Recording** mode *prior* to performing steps on these controls. (If you switch to **Standard Windows Recording** mode *after* performing an operation on a standard Windows control, both UFT and the SAP application may sometimes become unresponsive.)

This task describes how to toggle between the relevant recording options.

To switch to Standard Windows recording mode while recording a test in an SAP GUI for Windows application:

On the Record toolbar, select **Standard Windows Recording** from the **Recording Modes** drop-down, or press **ALT+SHIFT+F3**.

To record steps as SAP GUI for Windows objects again:

Do one of the following:

- On the Record toolbar, select **Default** from the **Recording Modes** drop-down.
- Stop the recording session.

This restores the normal recording mode for SAP GUI for Windows.

UFT-SAP Solution Manager Integration

This section includes

- [UFT-SAP Solution Manager Integration - Overview](#) 291
 - [Test Management in SAP Solution Manager](#)291
 - [Resource Files in Solution Manager](#) 292
- [Solution Manager Testing Modes: Standalone or Integrated](#) 292
 - [Standalone Mode](#)293
 - [Integrated Mode](#) 294
- [SAP Structured Parameters](#) 294
- [How to Configure Solution Manager to Work with UFT](#) 295
- [How to Work with Tests in Solution Manager in Standalone Mode](#) 297
- [How to Run a Test Stored in Solution Manager](#) 301
- [How to Display or Edit a GUI Test from Solution Manager in Integrated Mode](#) 303
- [How to Transfer Data To and From GUI Tests in Integrated Mode Using Test Parameters](#) 304
- [How to Work with SAP Structured Parameters](#) 305
- [Solution Manager Connection Dialog Box](#)309
- [Save GUI Test to Solution Manager Dialog Box](#)310
- [Open GUI Test from Solution Manager Dialog Box](#) 311
- [Upload File to Solution Manager Dialog Box](#)312
- [Save External File to Solution Manager Dialog Box](#) 313
- [Download File from Solution Manager](#) 314
- [SAP > SAP Solution Manager Pane \(Options Dialog Box > GUI Testing Tab\)](#) 315
 - [Solution Manager Trace Options Dialog Box](#)316
- [Value Map Dialog Box](#)317

UFT-SAP Solution Manager Integration - Overview

Note: Unless otherwise specified, references to Solution Manager in this Help file apply to all currently supported versions of SAP eCATT (SAP Extended Computer Aided Test Tool) and SAP Solution Manager. Note that some features and options may not be supported in the specific edition of Solution Manager or eCATT that you are using.

For a list of the supported versions of Solution Manager or eCATT, see the *HP Unified Functional Testing Product Availability Matrix*, available from the [HP Support Matrix page](#) (requires an HP passport).

In addition to ALM, HP's Web-based test management tool, you can also store and manage GUI tests in SAP Solution Manager.

UFT Add-in for SAP Solutions integrates with SAP Solution Manager. This means that you can use Solution Manager with UFT to run quality tests in environments that span beyond Windows and SAP environments including complex, multi-platform, highly-integrated composite, legacy, and proprietary enterprise applications.

Note: UFT cannot connect to both Solution Manager and ALM in the same session. Therefore, you cannot use Solution Manager to manage business components and application areas.

You can work with GUI tests stored in Solution Manager in standalone or integrated mode. The current mode is indicated in the UFT title bar.

- **"Standalone Mode"** (described on page 293). The test is opened from UFT.
- **"Integrated Mode"** (described on page 294). The test is opened from Solution Manager.

Solution Manager support is available only when:

- **SAP Frontend software** is installed on your computer (including support for Unicode).
- **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.
- The **UFT Add-in for SAP Solutions** is installed and loaded. For details, see ["How to Manage UFT Add-ins"](#) on page 28.
- **Solution Manager integration components** are installed.

Test Management in SAP Solution Manager

You can store and manage GUI tests for your SAP application in the file system or in a test management tool, such as HP ALM or SAP Solution Manager, depending on your testing needs.

To work with SAP Solution Manager, you must configure your Solution Manager server to work with UFT. After the server is configured, you can connect to Solution Manager from UFT in standalone mode, or

you can connect to UFT from Solution Manager in integrated mode. For details, see ["How to Configure Solution Manager to Work with UFT" on page 295](#).

You can create tests from UFT or from Solution Manager, store tests and associated resource files in the Solution Manager database, edit tests, run tests, and review run results. For details see, ["How to Work with Tests in Solution Manager in Standalone Mode" on page 297](#).

You can also call and pass values from a Solution Manager test script to a GUI test. For details, see ["How to Transfer Data To and From GUI Tests in Integrated Mode Using Test Parameters" on page 304](#).

For details on performing basic Solution Manager test management operations, see:

- ["How to Configure Solution Manager to Work with UFT" on page 295](#)
- ["How to Work with Tests in Solution Manager in Standalone Mode" on page 297](#)
- ["How to Run a Test Stored in Solution Manager" on page 301](#)
- ["How to Display or Edit a GUI Test from Solution Manager in Integrated Mode" on page 303](#)
- ["How to Transfer Data To and From GUI Tests in Integrated Mode Using Test Parameters" on page 304](#)

Resource Files in Solution Manager

When you save a GUI test in Solution Manager, make sure you store all associated resource files in Solution Manager so that any user who opens the test from Solution Manager will have access to all of the test's resource files.

Like test names, all test resource files stored in Solution Manager must begin with a valid prefix according to the server settings. For example, if your Solution Manager server requires all file names to begin with z, you would use the following naming convention: z<filename> (for example: zSOR_dwdm). You can set the default prefix for files in the Solution Manager pane of the Options dialog box. For details, see ["SAP > SAP Solution Manager Pane \(Options Dialog Box > GUI Testing Tab\)" on page 315](#).

When you create a file in UFT, such as a new shared object repository or recovery file, you can create the file as you normally would in UFT and then save the file directly to Solution Manager.

You can also upload existing files that are stored in the file system (such as external data table files, function library files, shared object repository files, recovery files, and environment variable files).

For details on uploading resource files to Solution Manager, see ["How to Work with Tests in Solution Manager in Standalone Mode" on page 297](#).

Solution Manager Testing Modes: Standalone or Integrated

The table below describes the basic differences between the [standalone](#) and [integrated](#) testing modes.

	Standalone Mode	Integrated Mode
Open test from	UFT	Solution Manager

	Standalone Mode	Integrated Mode
Solution Manager - UFT connection	Connect to Solution Manager from UFT using the "Solution Manager Connection Dialog Box" (described on page 309).	Solution Manager automatically establishes the Solution Manager - UFT connection.
Available UFT features	All UFT features are available. You can open and work with any test in Solution Manager 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 , UFT warns you that it will disconnect from Solution Manager and switch UFT 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 Solution Manager.	When you open the test, test resources stored in Solution Manager are opened in read-only mode.
Save location	Tests and uploaded files are automatically saved to the local package (\$TMP) in Solution Manager.	You can save tests to any package (including non-local packages).
Solution Manager dependence	Although UFT is connected to Solution Manager, you can work and navigate in Solution Manager independently.	Solution Manager is locked while the test is open in UFT. To release Solution Manager, close UFT.
Run results	All run results are stored in the file system. They cannot be accessed from your Solution Manager log list.	Run results are stored to the network drive you specify in the Solution Manager pane of the Options dialog box and in the Solution Manager server. You can access the run results from the Solution Manager log.

Standalone Mode

You can connect to Solution Manager from UFT. This is known as **standalone mode**. After you connect to Solution Manager in standalone mode, you can:

- Store tests in the Solution Manager database.
- Open existing tests from the Solution Manager database.
- Upload files to or download files from Solution Manager.
- Store a test's external resource files in Solution Manager. For example, you can store shared object repository files, data table files, function library files, environment variable files, and recovery files in your Solution Manager database. UFT provides a special set of Solution Manager-specific options that enable you to control certain elements of the Solution Manager-UFT integration.
- Pass values from a Solution Manager test script to a GUI test, or vice versa, using GUI test parameters. For example, 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** node of the GUI Testing tab in the Options dialog box (**Tools > Options > GUI Testing** tab > **General** node)). This option instructs UFT to automatically parameterize all the operation arguments in the steps of one

or more actions in your test, at the end of a UFT recording session. You can then supply the values for these test parameters from Solution Manager. For more details on parameters, see the *HP Unified Functional Testing User Guide*.

Note:

You can also perform many of these operations from Solution Manager in integrated mode. For details, see ["Integrated Mode" below](#). For a comparison of Solution Manager testing modes, see ["Solution Manager Testing Modes: Standalone or Integrated" on page 292](#).

You can also download resource files already stored in Solution Manager and save them in the file system.

Integrated Mode

You can connect to UFT from Solution Manager. This is called **integrated mode**. When you work in integrated mode, only UFT features related to the Solution Manager test are available in UFT. When you run tests in integrated mode, your run session results are accessible in the Solution Manager log.

When you log on to a Solution Manager server that is configured to integrate with UFT, you can view, edit, and run GUI tests that are stored in Solution Manager. You can also use the standard Solution Manager commands to copy, rename, and delete GUI tests, just as you would with any other file stored in Solution Manager.

When you open a GUI test from Solution Manager, UFT opens in integrated mode. In this mode, you can use all UFT features that are associated with the open test. However, you cannot save the open test with another name.

You can run a test in integrated mode by using the **Run** option in UFT or using the **Execute Test Script (F8)** option for a selected GUI test in Solution Manager. You can also execute a Solution Manager test script (or *blob*—Binary Large Object) that calls a GUI test. Creating Solution Manager scripts that call GUI tests is useful if you want to pass or retrieve values to or from a GUI test.

For details on configuring Solution Manager to work with UFT, see ["How to Configure Solution Manager to Work with UFT" on the next page](#).

Note: You can also work with tests stored in Solution Manager from the UFT interface in standalone mode. For details, see ["Standalone Mode" on the previous page](#). For a comparison of Solution Manager testing modes, see ["Solution Manager Testing Modes: Standalone or Integrated" on page 292](#).

SAP Structured Parameters

When you work in integrated mode with SAP Solution Manager, you can use **structure**-type test parameters to pass complex values such as XML values or arrays from a Solution Manager test script to a GUI test, or vice versa,

You create and maintain the structured parameters in SAP Solution Manager. Then you can map action parameters to the structured parameters in your test. When you run your test, UFT receives parameter values from SAP Solution Manager; and resolves the mapped local parameter with the actual value from SAP Solution Manager.

For more details, see ["How to Work with SAP Structured Parameters" on page 305](#).

How to Configure Solution Manager to Work with UFT

This task describes how to configure Solution Manager to work with UFT so that you can use the Solution Manager-UFT integration features available with the UFT Add-in for SAP Solutions.

This task includes the following steps:

- ["Prerequisites" below](#)
- ["Set external tool parameters in the ECCUST_ET table" below](#)
- ["Apply necessary roles or profiles to Solution Manager-UFT Users" on the next page](#)
- ["Register UFT to work with Solution Manager" on the next page](#)

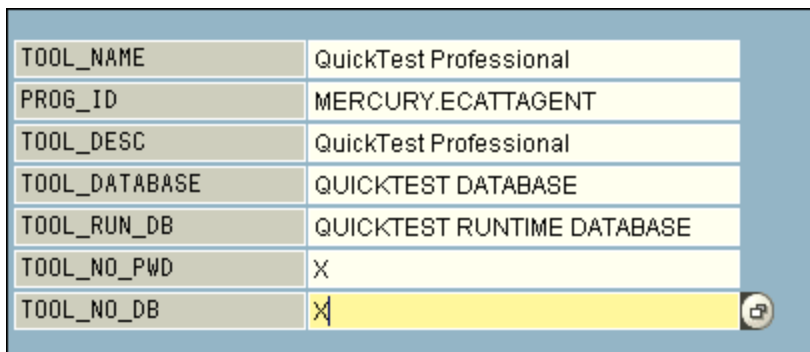
Prerequisites

You (or a Solution Manager system administrator) must install the appropriate support package and configure the Solution Manager server to work with UFT.

Set external tool parameters in the ECCUST_ET table

This step enables Solution Manager to communicate with UFT. (You perform this procedure only once in the system.)

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 details, see your Solution Manager documentation.

Apply necessary roles or profiles to Solution Manager-UFT Users

1. Make sure you have permission to:
 - Run Solution Manager scripts
 - Edit Solution Manager scripts
 - Work with an external tool (UFT) in integrated mode
 - Connect to Solution Manager from an external tool (UFT) in standalone mode
2. Confirm with your system administrator that the user name you use is assigned the necessary roles or profiles to perform the above tasks *before* you begin working with the UFT-Solution Manager integration. For example, to work with UFT in standalone mode, you must be assigned the role **S_ECET** or the profile **SAP_ECET** in the Solution Manager system. This is because each of these tasks requires special roles or profiles.

For details, contact your system administrator or see your SAP and Solution Manager documentation.

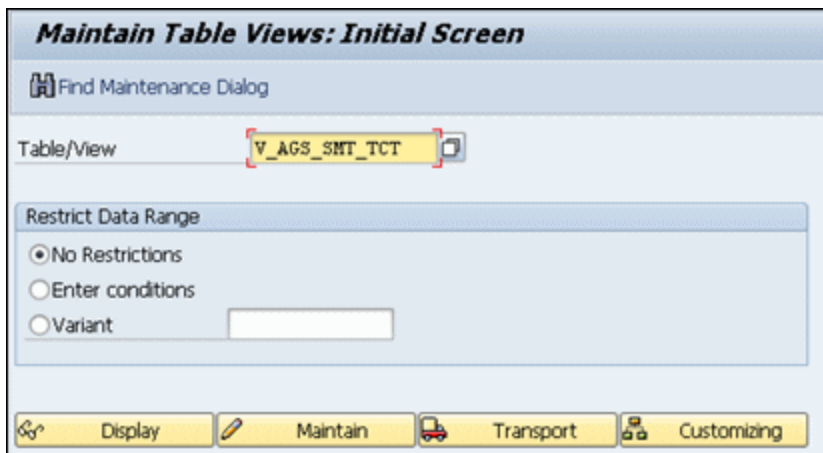
Register UFT to work with Solution Manager

To enable UFT to communicate with Solution Manager, you must register UFT and then verify the registration. You perform this procedure only once in the system.

1. Navigate to transaction **SPRO**, click **SAP Reference IMG**, and browse to **SAP Solution Manager > Capabilities (Optional) > Test Management > External Integration > External Test Tool with eCATT > Register Test Tool**.
2. Click **New Entries**.
3. Enter the values exactly as shown below:

Tool name	QUICKTEST PROFESSIONAL
Customizing Table for External Test Tools	
COM Program ID	MERCURY.ECATTAGENT
Tool Desc.	QUICKTEST PROFESSIONAL
Maintain DB	QUICKTEST DATABASE
TestExecutionDB	QUICKTEST RUNTIME DATABASE
<input checked="" type="checkbox"/> W/o Password	
<input checked="" type="checkbox"/> W/o Database	
<input checked="" type="checkbox"/> Transfer Log	
<input type="checkbox"/> Supp. BPCA Trace	

4. Navigate to transaction **SM30**, and enter **V_AGS_SMT_TCT**, as shown in the example below.



5. Click **Display** and verify that a row exists with the following information:

Test Case Types Settings			
Test Case Type	Test Tool	Active	Description
Test Configuration	3rd Party Test Tool	<input checked="" type="checkbox"/>	Test Configuration

UFT is now registered to work with your SAP application.

How to Work with Tests in Solution Manager in Standalone Mode

This task describes how to:

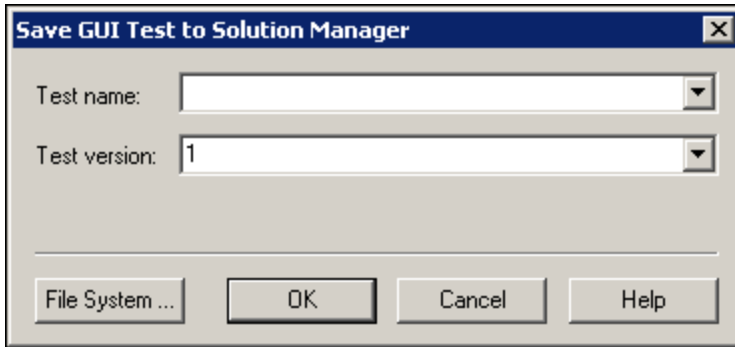
- ["Save a test in standalone mode" below](#)
- ["Open a test from Solution Manager in standalone mode" on the next page](#)
- ["Upload external resource files from the file system to Solution Manager" on page 299](#)
- ["Create a new shared object repository file and store it in Solution Manager" on page 299](#)
- ["Copy or export an object repository to Solution Manager" on page 300](#)
- ["Create a new recovery file in Solution Manager" on page 301](#)

Note: You can also open, edit, and save existing tests that are stored in Solution Manager, and you can save existing tests with a new name in the Solution Manager database or in the file system.

Save a test in standalone mode

1. In UFT, create or open a test.
2. From UFT, connect to a Solution Manager server. UFT connects to Solution Manager in standalone mode. For user interface details, see ["Solution Manager Connection Dialog Box" on page 309](#).

3. In UFT, select **File > Save As**. The Save GUI Test to Solution Manager dialog box opens.



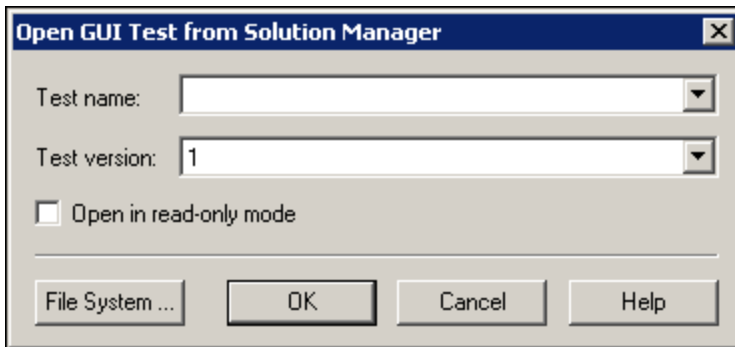
Note: If you defined a [New test prefix](#) in the SAP Solution Manager pane of the Options dialog box, the dialog box displays the defined prefix.

4. Do one of the following
 - To save a test directory to the file system, click **File System..** The Save Test dialog box opens. For user interface details, see the *HP Unified Functional Testing User Guide*.
 - To save the test to Solution Manager, enter the required information and click **OK**. For details on the user interface, see "[Save GUI Test to Solution Manager Dialog Box](#)" on page 310. When the save process finishes, the status bar displays the word **Ready**, and the test is saved to the local package (**\$TMP**) in Solution Manager. When the save process is complete, the UFT title bar displays the test information in the following format:

```
[Solution Manager] TestName: Version Number (Mode)
```

Open a test from Solution Manager in standalone mode

1. Connect to a Solution Manager server. For details, see "[Solution Manager Connection Dialog Box](#)" on page 309.
2. In UFT, select **File > Open > Test** to open the test. The "[Open GUI Test from Solution Manager Dialog Box](#)" on page 311 (described on page 311) opens.



Note: The Open GUI Test from Solution Manager dialog box opens when UFT is connected to a

Solution Manager server. To open a test directly from the file system while you are connected to Solution Manager, click the **File System** button to open the Open Test dialog box.

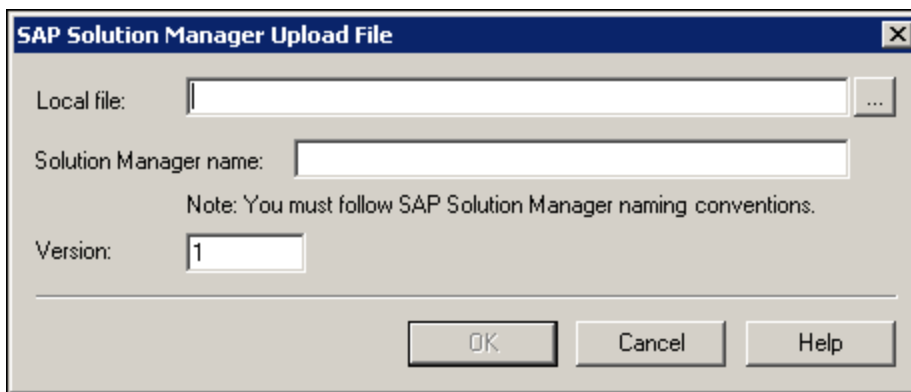
3. Enter the required information and click **OK**. For details on the user interface, see ["Open GUI Test from Solution Manager Dialog Box"](#) on page 311.

When the test opens, the UFT title bar displays the test information in the following format:

```
[Solution Manager] TestName: VersionNumber (Mode)
```

Upload external resource files from the file system to Solution Manager

1. Create and save the resource file in the file system.
2. Connect to Solution Manager. For details, see ["Solution Manager Connection Dialog Box"](#) on page 309.
3. Select **File > Upload File to Solution Manager** option. The ["Upload File to Solution Manager Dialog Box"](#) (described on page 312) opens.

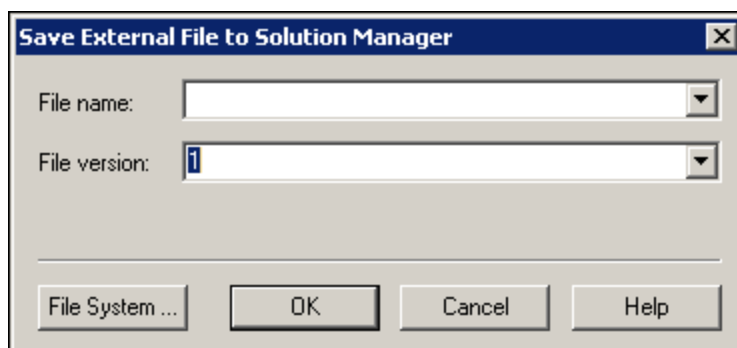


4. Browse or enter the file path of the **Local file** you want to upload.
5. Specify the **Solution** name and **Version** number you want to assign to the uploaded file.
6. Associate the uploaded file with your test in the appropriate UFT dialog box. For more details on associating Data pane files, library files, environment variable files, and shared object repositories with your test, see the *HP Unified Functional Testing User Guide*.

Create a new shared object repository file and store it in Solution Manager

1. Open a blank test.
2. Select **Resources > Object Repository Manager** and add test objects as needed.

3. Select **File > Save**. The "Save External File to Solution Manager Dialog Box" (described on page 313) 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 Solution Manager server. For example, if your Solution Manager server requires all file names 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 Solution Manager.
- For more details on creating object repository files, see the *HP Unified Functional Testing User Guide*.

Copy or export an object repository to Solution Manager

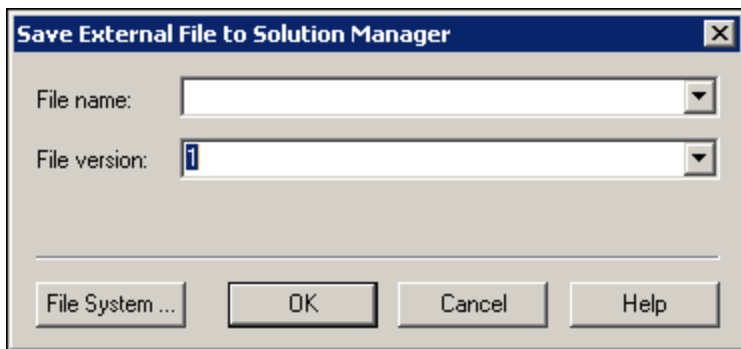
1. Open the test whose object repository you want to copy or export.

Note: If you are 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 Solution Manager.
 - In the Object Repository Window, select **File > Export Local Objects** to export the object repository to a shared object repository file in Solution Manager.

The "Save External File to Solution Manager Dialog Box" (described on page 313) opens.



4. Enter the required information and click **OK**. For details on the user interface, see "Save External File to Solution Manager Dialog Box" on page 313.

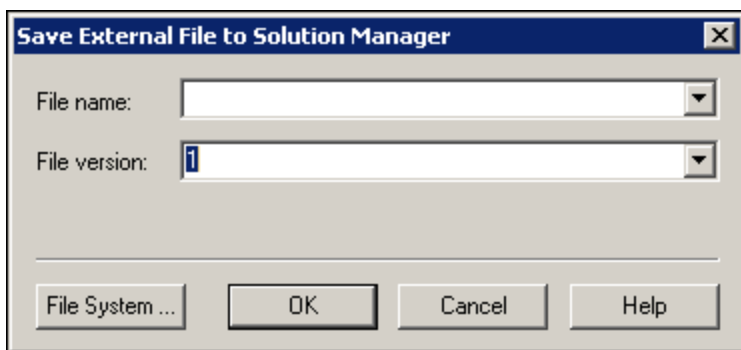
For details on exporting and saving object repository files, see the *HP Unified Functional Testing User Guide*.

Create a new recovery file in Solution Manager

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

To add more scenarios to the new scenario file, repeat this step.

When you are ready to save the scenario file, click **Save**. The "Save External File to Solution Manager Dialog Box" (described on page 313) opens.



3. Enter the required information and click **OK**. For details on the user interface, see "Save External File to Solution Manager Dialog Box" on page 313.

For details on creating and saving recovery files, see the *HP Unified Functional Testing User Guide*.

How to Run a Test Stored in Solution Manager

This task describes how to run tests from UFT. The run results are stored in the location you specify in the file system. You cannot access these results from Solution Manager.

When working with UFT in standalone mode, you run a test stored in a Solution Manager database just like any other UFT test.

This task describes how to:

- ["Run a test in standalone" below](#)
- ["Run a UFT test from Solution Manager in Integrated Mode using the Execute Test Script option" below](#)
- ["View results of a GUI test run in integrated mode" on the next page](#)

Run a test in standalone

1. Open UFT in standalone mode. For details, see ["Solution Manager Connection Dialog Box" on page 309](#).
2. In UFT, open the test you want to run. For user interface details, see ["Open GUI Test from Solution Manager Dialog Box" on page 311](#).
3. Click the **Run** button or select **Run > Run**. The Run Dialog Box opens.
4. Accept the default results folder or browse to select another one.

Note:

- 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 **Solution Manager test cache folder** in the **Solution Manager** pane of the Options dialog box.
- When running tests in standalone mode, no Solution Manager log is created.
- To run the test and overwrite the previous run session results, select the **Temporary run results folder (overwriting older temporary results)** option.
- UFT 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.

5. Click **OK**. The Run dialog box closes and UFT begins running the test.

When the run session ends, the run results open (unless the **View results when run session ends** check box is cleared in the **Run Sessions** pane of the Options dialog box (**Tools > Options > GUI Testing tab > General tab > Run Sessions** node)).

Run a UFT test from Solution Manager in Integrated Mode using the Execute Test Script option

For details, see your SAP documentation.

View results of a GUI test run in integrated mode

You can view the results of a GUI test that was run from Solution Manager in the following ways:

- **In the Solution Manager Log Display.** You can view the results of the test or the results of a specific event, such as a checkpoint, in the Solution Manager log. You can view the folder in which the results are saved in the **UNCPathToLocalLog** line of the Solution Manager log.
- **In UFT.** You can view the run results in the run results, which you can either access from UFT (as separate tab in the document pane) or open directly from your browser.

If a test includes steps that log on to Solution Manager using the "[SAP Tab \(Record and Run Settings Dialog Box\)](#)" (described on page 271), the logon steps are displayed in the run results tree.

You can set the Run Results Viewer to open automatically after a test runs from Solution Manager. To do this, in Solution Manager, select the **Log Display** check box in the **Shared** tab in **Start Options** window.

- **Via the generated XML Report.** Each time you run a GUI test from Solution Manager, 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 Solution Manager log.

How to Display or Edit a GUI Test from Solution Manager in Integrated Mode

This task describes how to display or edit any existing GUI test that is stored in Solution Manager. When you open the test, UFT opens in integrated and read-only mode. When you display or open a GUI 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.

Despite this, resource files that are saved with the test (for example, a local repository or the test's local data table) are editable. To edit external resource files, open the test in standalone mode.

Note: If you select the UFT **File > Save As** menu command when working in integrated mode, UFT displays a warning message indicating that you can save a test with a new name in the file system, but doing so disconnects UFT from Solution Manager and switches UFT to standalone mode.

For more details on integrated and standalone modes, see "[Standalone Mode](#)" on page 293 and "[Integrated Mode](#)" on page 294.

This task describes how to:

- "[Display or open a GUI test from Solution Manager](#)" below
- "[Create a new GUI test from Solution Manager](#)" on the next page

Display or open a GUI test from Solution Manager

1. Log on to Solution Manager and open a test script. Make sure to specify Unified Functional Testing

as the external tool. For details on how to open test scripts in Solution Manager, see your SAP documentation.

You can open the test in:

- **Read-only (Display) mode.** (If the test uses external resource files, the test and its resources open in read-only mode by default.)
 - **Edit mode.** This enables you to use most UFT options. For details, see the *HP Unified Functional Testing User Guide*.
2. To return to Solution Manager, close UFT.

Create a new GUI test from Solution Manager

1. Log on to Solution Manager. Then create and save a test script. Make sure to specify Unified Functional Testing as the external tool. For details on how to open test scripts in Solution Manager, see your SAP documentation.

When you create the script, UFT opens with a blank test.

2. Create the test in UFT. For details on creating GUI tests in UFT, see the other chapters in this guide and the *HP Unified Functional Testing User Guide*.
3. In UFT, select **File > Save As**. A dialog box opens in Solution Manager.
4. In Solution Manager, specify the package in which you want to store the test. Confirm that the other edit boxes contain correct values.

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.

5. In Solution Manager, save the test. UFT is restored in integrated mode and displays the saved test for additional editing.
6. To return to Solution Manager, close UFT.

How to Transfer Data To and From GUI Tests in Integrated Mode Using Test Parameters

You can pass values from a Solution Manager test script to a GUI test, or vice versa, using GUI test parameters.

To send values to your input arguments, you must run your test via a call from a Solution Manager test script. After you define input and output arguments for your GUI test, you can insert a call to that test from a Solution Manager test script and specify argument values for the input arguments.

Prerequisites

Define test parameters and use them in your GUI test. For details, see the section on the "Parameters

Tab (Properties Pane - Testing)" in the *HP Unified Functional Testing User Guide*.

If you are working with SAP structured parameters, see also "[How to Work with SAP Structured Parameters](#)" below.

Call a GUI test and specify arguments from Solution Manager

To send values to your input arguments from Solution Manager, you must run your test via a call from a Solution Manager test script.

Tip: You can enter the name of a Solution Manager parameter from the Solution Manager script as the value of a GUI input parameter.


After you define input and output arguments for your GUI test, you can insert a call to that test from a Solution Manager script and specify argument values for the input arguments.

How to Work with SAP Structured Parameters

When you work in integrated mode with SAP Solution manager, you can pass values from a Solution Manager test script to a GUI test, or vice versa, using the structure value type for your test parameters.

You create and maintain the structured parameters in SAP Solution Manager. After you have defined your test parameters via SAP Solution Manager you can map action parameters to the structured parameters in the test. When you run a test, UFT receives the defined structure from SAP Solution Manager, and resolves the mapped local parameter with the actual structured parameter value from SAP Solution Manager.

Create or modify the structured parameters of a test

1. From SAP Solution Manager, Launch your test as an external test.
UFT opens with your test displayed.
2. In the Parameters tab of the Properties pane, click the Maintain SAP Parameters icon . If the Properties Pane is not already open, select **View > Properties** to open it.
SAP Solutions Manager opens and UFT is hidden.

Note: SAP Structured Parameters can be maintained only in SAP Solution Manager.

3. In SAP Solutions Manager, create or modify the structure parameters you want to use for your test, save your changes and click **Back**.
UFT re-opens with the changes you made now available.

Assign or modify the structured parameters for an action

To use the SAP structured parameter to run the test, you must define parameters of type structure, associate the test and action parameters, and then map the action's structured parameter to the test's structured parameter. You can also map a simple type action parameter to a single element in a

structured parameter defined for the test.

1. Select the relevant action.
2. In the Parameters tab of the Properties pane, you can add or remove parameters, as described in *Add/Edit Input/Output Parameter Dialog Box (Properties Pane - GUI Testing)* in the *HP Unified Functional Testing User Guide*. If the Properties Pane is not already open, select **View > Properties** to open it.

Note: If the test contains structured parameters, you can add parameters with the structure type to the action.

3. Map action parameters to the test structured parameters
 - a. Right-click the relevant actions.
 - b. In the context menu, select **Action Call Properties**.
 - c. In the Action Call Properties Dialog Box > Parameter Values tab, in the **Value** cell/column of an input parameter or the **Store In** cell/column of an output parameter, click the **Configure Value** button.

- d. In either the Value Configuration Options Dialog Box (for input parameters) or the Storage Location Options Dialog Box (for output parameters), select the **Parameter** radio button and click **Browse**.

The Value Map dialog box opens.

- e. Define the mapping, as described in ["Value Map Dialog Box"](#) on page 317.

You can select the root node to map the entire structure, or a sub tree node to map to an embedded structure, or you can select a leaf node to map to a specific value in the structure.

- f. Click **OK**.

The parameters are mapped

Note: If an action is called inside another action, you can map a parameter to an input parameter of the parent action. If the action is called after another action, you can map a parameter to an output parameter of any previous action.

For more details on test and action parameters, see the *HP Unified Functional Testing User Guide*.

Use structured parameters in a script

The examples below show how you can use structured parameters directly from a script.

```
<?xml version="1.0" encoding="utf-16"?>  
<ZMOVIE>  
  <TITLE>Avatar</TITLE>  
  <DIRECTOR>
```

```
<FIRST_NAME>James</FIRST_NAME>  
<LAST_NAME>Cameron</LAST_NAME>  
<BIRTHDAY>16-8-1954</BIRTHDAY>  
</DIRECTOR>  
<REL_DATE>10-12-2009</REL_DATE>  
<GENRE>SF</GENRE>  
<STARRING>  
  <item>  
    <FIRST_NAME>Michelle</FIRST_NAME>  
    <LAST_NAME>Rodriguez</LAST_NAME>  
  </item>  
  <item>  
    <FIRST_NAME>Stephen</FIRST_NAME>  
    <LAST_NAME>Lang</LAST_NAME>  
  </item>  
  <item>  
    <FIRST_NAME>Zoe</FIRST_NAME>  
    <LAST_NAME>Saldana</LAST_NAME>  
  </item>  
</STARRING>  
</ZMOVIE>
```

- To access an element in a structured parameter, type the parameter name followed by a colon (:)
and then the element path. Use a period (.) between elements and their sub-elements. For example:

```
Print Parameter("Param1:ZMOVIE.DIRECTOR.FIRST_NAME")
```

Output:

```
James
```

```
Print Parameter("Param1:ZMOVIE.STARRING.item[1].FIRST_NAME")
```

Output:

```
Michelle
```

Note: UFT provides statement completion for structured parameters, displaying the elements available for the relevant structure type.

- If the path represents an element that contains additional sub-elements, the returned value will be a
XML string. The path follows the XPath expression rule

```
Print Parameter("Param1:ZMOVIE.DIRECTOR")
```

Output:

```
<DIRECTOR>
```

```
<FIRST_NAME>James</FIRST_NAME>  
<LAST_NAME>Cameron</LAST_NAME>  
<BIRTHDAY>16-8-1954</BIRTHDAY>  
</DIRECTOR>
```

- If the structure is an array or a table, you can use it in a loop:

```
rowCount = Parameter("Param1:ZMOVIE.STARRING.item.count()")  
For Iterator = 1 To rowCount Step 1  
    first_name = "Param1:ZMOVIE.STARRING.item[" & Iterator & "].FIRST_NAME"  
    last_name = "Param1:ZMOVIE.STARRING.item[" & Iterator & "].LAST_NAME"  
    print Parameter(first_name) & " " & Parameter(last_name)  
Next
```

Output:

```
Michelle Rodriguez  
Stephen Lang  
Zoe Saldana
```

Note: You can omit the root element from the path. For example,

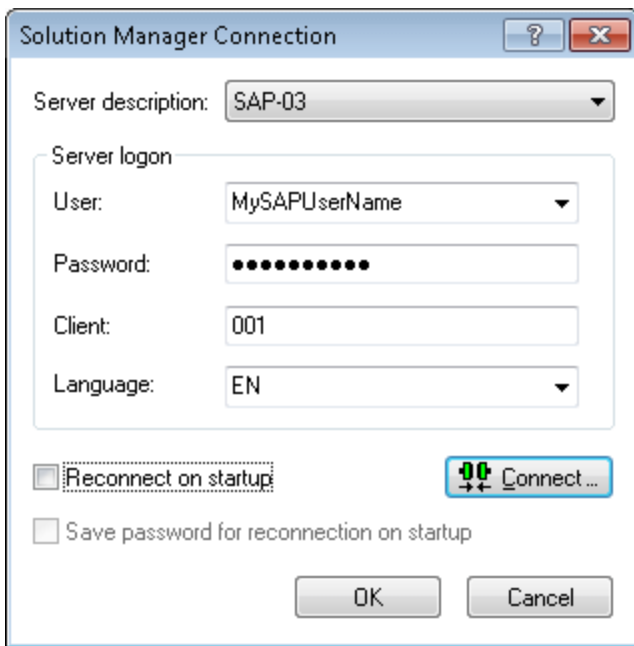
```
Print Parameter("Param1:ZMOVIE.DIRECTOR.FIRST_NAME")
```

Can also be written as

```
Print Parameter("Param1:DIRECTOR.FIRST_NAME")
```

Solution Manager Connection Dialog Box

This dialog box enables you to connect to or disconnect from Solution Manager at any time during the testing process.



To access	<ol style="list-style-type: none"> 1. Make sure that the computer is connected to Solution Manager. 2. Make sure that UFT is open with the Add-in for SAP Solutions loaded. 3. Select Tools > SAP Solution Manager Connection.
Important information	Do not disconnect UFT from Solution Manager while a GUI test that is stored in Solution Manager is open or while UFT is using a shared resource stored in Solution Manager (such as a shared object repository or data table file).
Relevant tasks	<ul style="list-style-type: none"> • "How to Configure Solution Manager to Work with UFT" on page 295 • "How to Work with Tests in Solution Manager in Standalone Mode" on page 297 • "How to Run a Test Stored in Solution Manager" on page 301 • "How to Display or Edit a GUI Test from Solution Manager in Integrated Mode" on page 303 • "How to Transfer Data To and From GUI Tests in Integrated Mode Using Test Parameters" on page 304

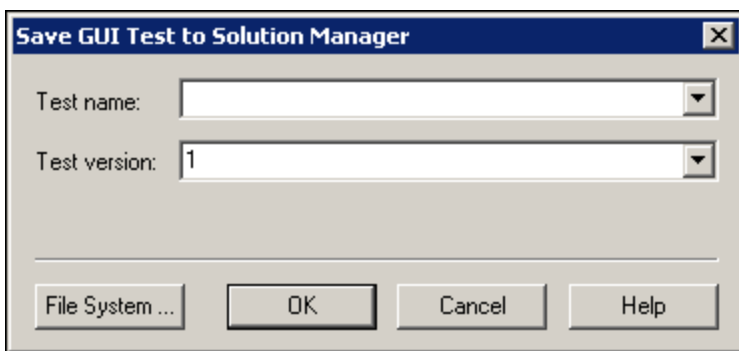
User interface elements are described below:

UI Elements	Description
Server description	<p>The Solution Manager server to which you want to connect.</p> <p>The Server description box lists the servers available in the SAP Logon Pad or the SAP Logon dialog box.</p>

UI Elements	Description
	To add a server to the list in the Solution Manager Connection dialog box, close the dialog box, define an appropriate entry using your SAP Logon dialog box, and then reopen the Solution Manager 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 UFT to automatically reconnect to the Solution Manager server the next time you open UFT.
Save password for reconnection on startup	Instructs UFT 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 UFT opens. Enabled only when Reconnect on startup is selected.
Connect	Connects UFT to Solution Manager.

Save GUI Test to Solution Manager Dialog Box

This dialog box enables you to save a GUI test in the Solution Manager database.



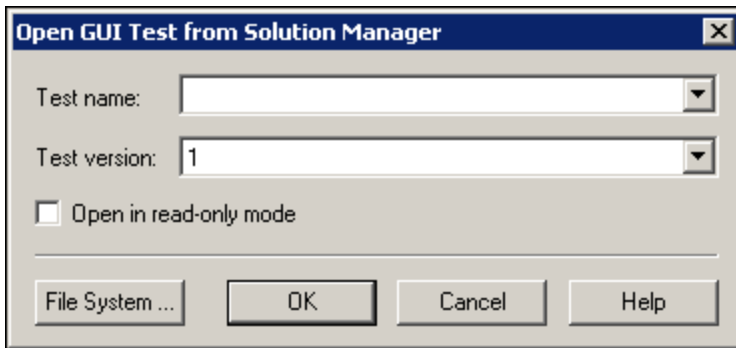
To access	<ol style="list-style-type: none"> 1. Make sure that the computer is connected to Solution Manager. 2. Make sure that UFT is open with the Add-in for SAP Solutions loaded. 3. In UFT, display a GUI test and select File > Save As.
Important information	<ul style="list-style-type: none"> • This dialog box is similar to the Save External File to Solution Manager dialog box, and displays similar user interface elements. • When saving a test to Solution Manager in standalone mode, the test is automatically saved to the local package (\$TMP) in Solution Manager.

User interface elements are described below:

UI Elements	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 Solution Manager server naming conventions. For example, your Solution Manager server may require all file names to begin with the letter z.</p> <p>When the Save Test in Solution Manager dialog box opens, it displays the default test prefix in the Test name box. You can define or modify this prefix in the Solution Manager pane of the Options dialog box. For details, see "SAP > SAP Solution Manager Pane (Options Dialog Box > GUI Testing Tab)" on page 315.</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>Enables you to save the currently open test anywhere in the file system.</p>

Open GUI Test from Solution Manager Dialog Box

This dialog box enables you to open a GUI test from the Solution Manager database.



To access	<ol style="list-style-type: none"> 1. Make sure that the computer is connected to Solution Manager. 2. Make sure that UFT is open with the Add-in for SAP Solutions loaded. 3. In UFT, select File > Open > Test.
Important information	<p>This dialog box is similar to the Open External File from Solution Manager dialog box, except that this dialog box contains the Open in read-only mode option.</p>
Relevant tasks	<p>"How to Work with Tests in Solution Manager in Standalone Mode" on page 297</p>

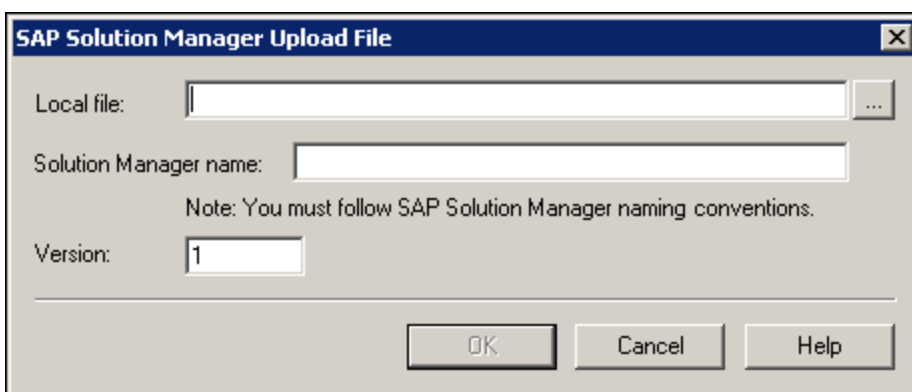
User interface elements are described below:

UI Elements	Description
Test name	<p>The name of the test you want to open.</p> <p>When the Open GUI Test from Solution Manager dialog box opens, it displays the most recently opened</p>

UI Elements	Description
	Solution Manager test in the Test name box. You can enter a valid test name or select one from the list of recently opened Solution Manager tests. Do not specify a folder path or other location.
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	Enables you to open a test from anywhere in the file system.

Upload File to Solution Manager Dialog Box

This dialog box enables you to store a test's external resource files in Solution Manager. You use this dialog box to upload files that are stored in the file system, such as external data table files, external function library files, and environment variable files, shared object repository files, and recovery files.



To access	<ol style="list-style-type: none"> 1. Make sure that the computer is connected to Solution Manager. 2. Make sure that UFT is open with the Add-in for SAP Solutions loaded. <p style="text-align: center;">Note: This step is optional for standalone mode.</p> <ol style="list-style-type: none"> 3. In UFT, select File > Upload File to Solution Manager.
Important information	When uploading a file to Solution Manager in standalone mode, it is automatically saved to the local package (\$TMP) in Solution Manager.
Relevant tasks	"How to Work with Tests in Solution Manager in Standalone Mode" on page 297

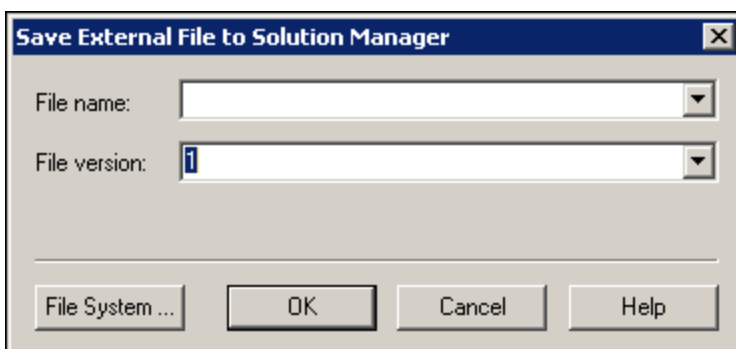
User interface elements are described below:

UI Elements	Description
Local file	The complete path of the file you want to upload. You can enter a file path or browse to the file.

UI Elements	Description
Solution	<p>The name under which to store the file in Solution Manager. Ensure that the file name begins with a prefix that matches your Solution Manager server naming conventions. For example, you may have to prefix all file names with the letter z.</p> <p>When the Upload File to Solution Manager dialog box opens, it displays the default file prefix in the Solution Manager name box. You can define or modify this prefix in the Solution Manager pane of the Options dialog box. For details, see "SAP > SAP Solution Manager Pane (Options Dialog Box > GUI Testing Tab)" on page 315.</p>
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>

Save External File to Solution Manager Dialog Box

This dialog box enables you to save a GUI test's resource files directly to Solution Manager.



To access	<ol style="list-style-type: none"> 1. Make sure that the computer is connected to Solution Manager. 2. Make sure that UFT is open with the Add-in for SAP Solutions loaded. <p>Note: This step is optional for standalone mode.</p> <ol style="list-style-type: none"> 3. In UFT, display the file and select Tools > Save As.
Important information	<p>This dialog box is similar to the Open External File from Solution Manager dialog box, except that this dialog box does not contain the Open in read-only mode option.</p>
Relevant tasks	<p>"How to Work with Tests in Solution Manager in Standalone Mode" on page 297</p>

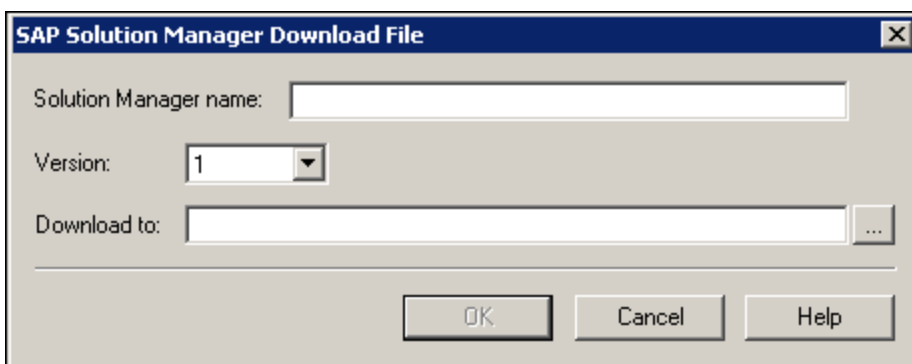
User interface elements are described below:

UI Elements	Description
File	<p>The name under which to store the file in Solution Manager. Ensure that the file name begins with a prefix that</p>

UI Elements	Description
name	<p>matches your Solution Manager server naming conventions. For example, you may have to prefix all file names with the letter z.</p> <p>When the Save External File to Solution Manager dialog box opens, it displays the default file prefix in the Solution Manager name box. You can define or modify this prefix in the SAP Solution Manager pane of the Options dialog box. For details, see "SAP > SAP Solution Manager Pane (Options Dialog Box > GUI Testing Tab)" on the next page.</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>

Download File from Solution Manager

This dialog box enables you to download files stored in Solution Manager and save them in the file system.



To access	<ol style="list-style-type: none"> 1. Make sure that the computer is connected to Solution Manager. 2. Make sure that UFT is open with the Add-in for SAP Solutions loaded. 3. In UFT, select File > Download File from Solution Manager.
Important information	<p>If you upload a file to Solution Manager and then associate that file with a test as a resource file, the resource file is automatically downloaded each time you open the test, and you do not need to use this dialog box.</p>

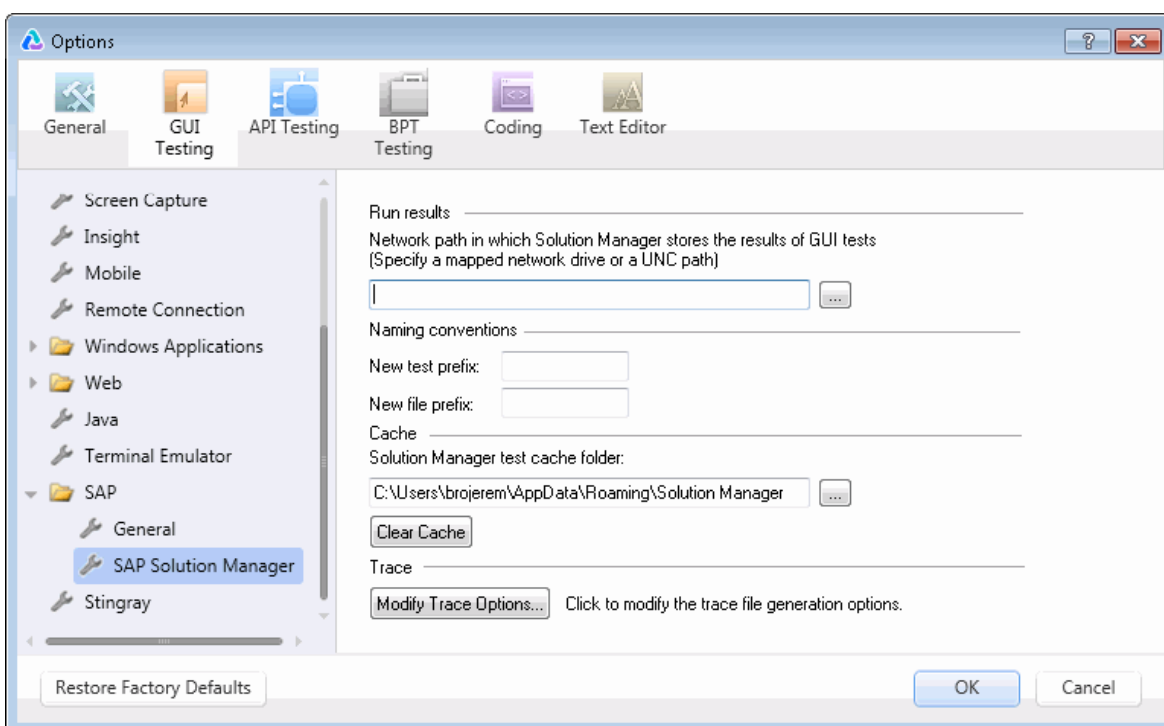
User interface elements are described below:

UI Elements	Description
Solution	<p>The name of the file stored in Solution Manager.</p> <p>When the Download File from Solution Manager dialog box opens, it displays the default file prefix in the Solution Manager name box. You can define or modify this prefix in the Solution Manager pane of the Options dialog box. For details, see "SAP > SAP Solution Manager Pane (Options Dialog Box > GUI Testing Tab)" on the next page.</p>

UI Elements	Description
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.

SAP > SAP Solution Manager Pane (Options Dialog Box > GUI Testing Tab)

This pane enables you to configure how UFT behaves when you are connected to SAP Solution Manager.



To access	<ol style="list-style-type: none"> 1. Make sure that UFT is open with the Add-in for SAP Solutions loaded. 2. Make sure that a GUI test is open. 3. In UFT, select Tools > Options > GUI Testing tab > SAP > SAP Solution Manager node.
------------------	--

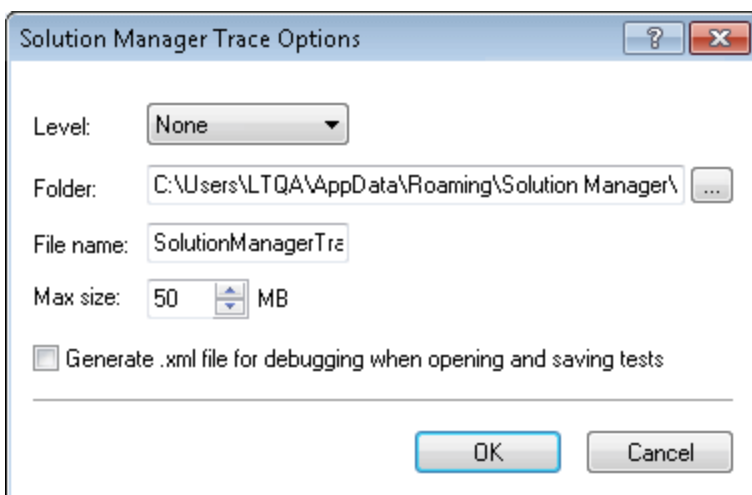
User interface elements are described below:

UI Elements	Description
Run results	The location in which run results are stored when the test is run from Solution Manager. 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 GUI Test in Solution Manager dialog box.

UI Elements	Description
New file prefix	The prefix that is displayed by default when specifying a file to store in Solution Manager.
Solution Manager test cache folder	<p>The location in which a test from Solution Manager is temporarily stored when it is open for editing or running in UFT.</p> <p>Default location = %APPDATA%\Solution Manager</p> <p>For example = C:\Documents and Settings\<username>\Application Data\Solution Manager</username></p> <p>Note: RunCache and EditCache folders are automatically created under the specified folder.</p>
Clear Cache	Deletes all files from the Solution Manager test cache folder.
Modify Trace Options	Opens the " Solution Manager Trace Options Dialog Box ", which enables you to configure if and how UFT generates the UFT–Solution Manager communication trace log (used for troubleshooting communication errors). For details, see " Solution Manager Trace Options Dialog Box " below.

Solution Manager Trace Options Dialog Box

This dialog box enables you to configure whether and how UFT generates a trace file. For example, you can instruct UFT to generate a UFT–Solution Manager communication trace file each time Solution Manager runs a UFT test to troubleshoot communication errors.



To access	<ol style="list-style-type: none"> 1. Connect the computer to Solution Manager in "Standalone Mode". 2. Do one of the following: <ul style="list-style-type: none"> • Ensure that a GUI test, action, or component is in focus in the document pane. • In the Solution Explorer, select a GUI test or component node or one of its child nodes. 3. In UFT, select Tools > Options > GUI Testing tab > SAP > SAP Solution Manager node. 4. In the SAP Solution Manager pane, click Modify Trace Options.
------------------	---

User interface elements are described below:

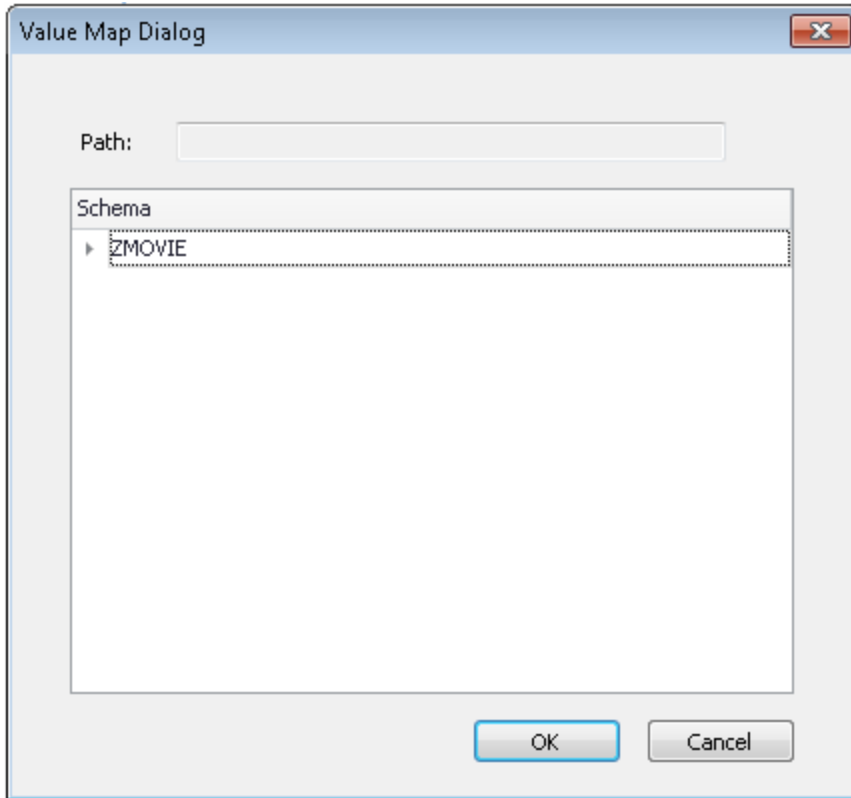
UI Elements	Description
Level	<p>The level of detail to include in the trace file that is created when Solution Manager runs a GUI test.</p> <ul style="list-style-type: none"> • None. (default) No trace file is created. • Low. The trace file lists any Solution Manager–UFT communication errors. • Medium. The trace file includes Solution Manager–UFT communication errors and information on other major operations that result in Solution Manager–UFT communication. • High. The trace file includes all available information related to Solution Manager–UFT communications.
Folder	<p>The folder path for storing the trace file. Required if a trace file level other than None is specified in the Level option.</p> <p>Default location = %APPDATA%\Solution Manager\Trace</p> <p>For example = C:\Documents and Settings\<username>\Application Data\Solution Manager\Trace</username></p>
File name	<p>The file name for the trace file.</p> <p>Default = SolutionManagerTrace.html</p>
Max size	<p>The maximum file size you want to allow for the trace file.</p>
Generate .xml file for debugging when opening and saving tests	<p>Generates an .xml file each time you open or save a test that is stored in Solution Manager. The files are saved in Open and Save folders under the trace folder.</p> <p>Note: Selecting this option results in slower response times for editing and saving tests in UFT. In general, you should select this option only when instructed to do so to debug Solution Manager connectivity issues.</p>

Value Map Dialog Box

Relevant for: GUI tests and components

This dialog box enables you to map a structured parameter to an action parameter.

The image below shows the dialog to map parameters.



To Access	<ul style="list-style-type: none">• For Input Parameters, click Browse in the Value Configuration Options dialog.• For Output Parameters, click Browse in the Storage Location Options dialog.
------------------	---

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

UI Element	Description
Path	The path of the selected node.
Schema	The structure of the parameter in tree format. You can open and close each node.

Part 13: Siebel Add-in

This section includes:

["Siebel Add-in - Quick Reference" on page 320](#)

["Siebel Add-in - Testing and Configuration" on page 323](#)

["Siebel Test Express" on page 339](#)

Siebel Add-in - Quick Reference

You can use the UFT Siebel Add-in to test Siebel user-interface objects (controls).

The following tables summarize basic information about the Siebel Add-in and how it relates to some commonly-used aspects of UFT.

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 "Web-Based Application Support" on page 34 .
Supported Environments	For details on supported Siebel environments, see the Siebel Add-in section of the <i>HP Unified Functional Testing Product Availability Matrix</i> , available from the HP Support Matrix page (requires an HP passport). If you are testing SiebelOpenUI applications, the SiebelOpenUI Add-in is supported as a child add-in of the Web Add-in and appears in the Add-in Manager as a child add-in of the Web Add-in. The SiebelOpenUI Add-in must be installed as one of the Web 2.0 toolkits. You should load the Web Add-in and the SiebelOpenUI Add-in, but do not load the Siebel Add-in. If you load both the Siebel and the SiebelOpenUI add-ins, the add-ins sometimes conflict with each other, and prevent successful object recognition.
Important Information	<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. For details, see "Siebel Test Express" on page 339. For general considerations, see "Considerations - Siebel Add-in" on page 324.
Test Object Methods and Properties	The Siebel Add-in provides test objects, methods, and properties that can be used when testing objects in Siebel applications. For details, see the Siebel section of the <i>HP UFT Object Model Reference for GUI Testing</i> .
Checkpoints and Output Values	<ul style="list-style-type: none"> See the sections describing checkpoints and output values in the <i>HP Unified Functional Testing User Guide</i>. See "Considerations - Siebel Add-in" on page 324. See "GUI Checkpoints and Output Values Per Add-in" on page 512
Troubleshooting and Limitations	"Troubleshooting and Limitations - Siebel Add-in" on page 335
Prerequisites	
Opening Your Application	You must open UFT 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	<ul style="list-style-type: none"> To test a Siebel 7.7.x or later application, you must: <ul style="list-style-type: none"> Modify the Siebel Test Automation module configuration. Instruct your Siebel application to generate test automation information. See "Siebel 7.7.x or Later - Test Automation Module Configuration" on page 330 .

	<ul style="list-style-type: none"> To test your Web application in Mozilla Firefox version 33 and later, you must enable the Functional Testing Extension in one of the following ways: <ul style="list-style-type: none"> If the Select Your Add-ons screen is displayed when you open Firefox, select to enable the Functional Testing Extension. If the Install Add-on tab opens and displays Functional Testing Extension when opening Firefox, select the Allow this installation check box and click Continue. Otherwise: <ol style="list-style-type: none"> In Firefox, open the browser menu. In the menu, click Add-ons. In the Add-ons Manager tab, select the Extensions node. In the Functional Testing Extension row, click the Enable button. <p>To test your Web application in Mozilla Firefox versions earlier than 33, you must enable the legacy Firefox agent:</p> <ol style="list-style-type: none"> Open the <UFT installation folder>\Installations\Firefox folder. From the Firefox folder, drag the Legacy.xpi file into Firefox. In Firefox, open the browser menu. In the menu, click Add-ons. In the Add-ons Manager tab, select the Extensions node. In the Functional Testing Extension row, disable the Functional Testing 12.50 extension and enable the extension you added to Firefox.
--	---

Configuration	
Options Dialog Box	Use the Web pane. (Make sure that a GUI test is open and select Tools > Options > GUI Testing tab > Web > General node.) See " Web > General Pane (Options Dialog Box) " on page 58.
Record and Run Settings Dialog Box (tests only)	Use the Siebel tab. (Record > Record and Run Settings) See " Siebel Tab (Record and Run Settings Dialog Box) " on page 333.
Test Settings Dialog Box (tests only)	Use the Web pane. (File > Settings > Web node) See " Web Pane (Test/Business Component Settings Dialog Box / Application Area - Additional Settings Pane) " on page 68.
Custom Active Screen Capture Settings Dialog Box (tests only)	Use the Web section. (Tools > Options > GUI Testing tab > Active Screen node > Custom Level) See the section on the Custom Active Screen Capture Settings dialog box in the <i>HP Unified Functional Testing User Guide</i> .
Application Area Additional Settings pane	<ul style="list-style-type: none"> Use the Web pane. In the application area, click Additional Settings > Web in the sidebar.

<p>(business components only)</p>	<p>See "Web Pane (Test/Business Component Settings Dialog Box / Application Area - Additional Settings Pane)" on page 68.</p> <ul style="list-style-type: none">• Use the Applications pane. In the application area, select Additional Settings > Applications in the sidebar. <p>In the Siebel version box, specify the Siebel version for the applications on which you want to record your business component. The version that you choose remains selected for all subsequent business components.</p>
-----------------------------------	---

Siebel Add-in - Testing and Configuration

This chapter includes:

- [Siebel Add-in - Overview](#) 324
- [Considerations - Siebel Add-in](#) 324
- [Siebel Test Object Model - Overview](#) 325
 - [Recording Steps on Siebel Objects](#) 326
 - [Native Operations and Properties in Siebel 7.0.x and 7.5.x Applications](#) 327
- [Siebel Add-in - Checkpoints and Output Values](#) 327
 - [Spooling Data from a Siebel Table](#) 329
- [Siebel 7.7.x or Later - Test Automation Module Configuration](#) 330
- [How to Define Environment Variables for Siebel Applications](#) 331
- [Siebel Tab \(Record and Run Settings Dialog Box\)](#) 333
- [Troubleshooting and Limitations - Siebel Add-in](#) 335
 - [Siebel 7.7.x or Later](#) 336
 - [Siebel 7.0.x and 7.5.x](#) 337

Siebel Add-in - Overview

The Siebel eBusiness platform is widely used in many organizations for their business process applications. UFT can create and run tests and business 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 business components on Siebel applications.

UFT 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.

Considerations - Siebel Add-in

General Considerations

- UFT 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 details, see ["Siebel Test Object Model - Overview" on the next page](#).
- 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 UFT behaves when you record on other non-Siebel Web objects. For details, see ["Event Recording Configuration for Web-Based Applications" on page 39](#).
- 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 business component on your Siebel 7.7.x or later application, UFT records the operations you perform. UFT works directly with the Siebel Test Automation API (SiebelAx_Test_Automation_18306.exe) to record your operations. Therefore, although UFT records a step for each operation you perform, it adds the steps to your test or business component only when API events are sent to UFT (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 UFT when recording and running tests or business 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 business component with dynamically created pages.

Siebel Test Object Model - Overview

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, UFT learns only **Sbl** test objects. If you are learning objects on a Siebel 7.7.x or later application, UFT 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 details on each of the Siebel test objects, see the **Siebel** section of the *HP UFT Object Model Reference for GUI Testing*.

When you perform an operation on your Siebel application while recording a test or business component, UFT:

- identifies the object on which you performed the operation and creates the appropriate test object in the test or business 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 Editor.

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**.

UFT 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
RBC	repositoryname	Competitor
RBC	classname	SiebCheckbox

It also records that you performed a SetOn method to select the SiebCheckbox object.

UFT displays your step in the Keyword View like this:

▼ Siebel Call Center		
▼ Accounts		
▼ Account Details		
▼ Account		
Competitor	SetOn	Select the "Competitor" check box.

UFT displays your step in the Editor like this:

```
SiebApplication("Siebel Call Center").SiebScreen("Accounts").SiebView("Account Details").SiebApplet("Account").SiebCheckbox("Competitor").SetOn
```

When you run a test or business component, UFT 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, UFT 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`), UFT searches the application for a SiebCheckbox object named **Competitor**. When it finds the object, UFT performs the **SetOn** method on the object to select the check box.

This section also includes:

- [Recording Steps on Siebel Objects](#)326
- [Native Operations and Properties in Siebel 7.0.x and 7.5.x Applications](#)327

Recording Steps on Siebel Objects

When you record an operation on a Siebel object, UFT inserts a step with the relevant Siebel object in the Keyword View and adds the corresponding statement in the Editor.

For example, if you select an item from a list, the Keyword View may be displayed as follows:

Item	Operation	Value	Documentation
▼ Siebel Call Center			
▼ Accounts			
▼ Account Details			
▼ Account			
Account Type	Select	"Consultant"	Select the "Consultant" item in the "Account Type" pick list.

UFT records this step in the Editor 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, you can use UFT to generate an object repository for your application. For details, see ["Siebel Test Express" on page 339](#)

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 details on using the Object property, see the *HP Unified Functional Testing User Guide*.

Siebel Add-in - Checkpoints and Output Values

After you create your test or business component, you can enhance it by adding checkpoints, retrieving output values, parameterizing values, and inserting Siebel objects, methods and properties.

You check most Siebel objects or output their property values in the same way as you do for other objects supported by UFT, with exceptions for **SblTable** objects and **Sieb** tabular test objects.

Considerations - Siebel Add-in Checkpoints and Output Values

- You check SblTable objects and output their values in the same way as you do for other table objects supported by UFT—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 details, see "[Spooling Data from a Siebel Table](#)" on the next page.

- Specific test objects in Siebel 7.7.x applications (with **Sieb** prefixes) have tabular characteristics. UFT 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 business 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 details, see "[Spooling Data from a Siebel Table](#)" on the next page.

- 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.

This section also includes:

- [Spooling Data from a Siebel Table](#)329

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 details on the **GetCellData**, **RowCount**, and **ColumnsCount** methods, see the **Siebel** section of the *HP UFT Object Model Reference for GUI Testing*.

Siebel 7.7.x or Later - Test Automation Module Configuration

UFT 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 business 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.

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 business components on these Siebel application versions.

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 - Siebel Application

To create and run tests or business 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 (Record and Run Settings Dialog Box)" (described on page 333), UFT automatically appends the Siebel Test Automation information to the URL (you do not need to specify it manually in the URL). For details on the Record and Run Settings dialog box options, see "How to Define Environment Variables for Siebel Applications" on the next page.

If you select the **Open the following application when a record or run session begins** option in the "Siebel Tab (Record and Run Settings Dialog Box)" (described on page 333), UFT automatically appends the Siebel Test Automation information to the URL (you do not need to specify it manually in the URL). For details on the Record and Run Settings dialog box options, see "How to Define Environment Variables for Siebel Applications" on the next page.

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 - 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 `SWECommand=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 UFT 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 details 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 (Record and Run Settings Dialog Box)"** (described on page 333), 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, UFT automatically appends the password information to the URL (you do not need to specify it manually in the URL). For details on the Record and Run Settings dialog box options, see ["How to Define Environment Variables for Siebel Applications"](#) below.

How to Define Environment Variables for Siebel Applications

Note:

- If you define any of these environment variables, it overrides the corresponding values in the **"Siebel Tab (Record and Run Settings Dialog Box)"** (described on page 333) (for components), or the Applications pane in the application area's Additional Settings pane (for application areas).
- For details on defining and working with environment variables, see ["Environment Variables in Record and Run Settings"](#) on page 26.

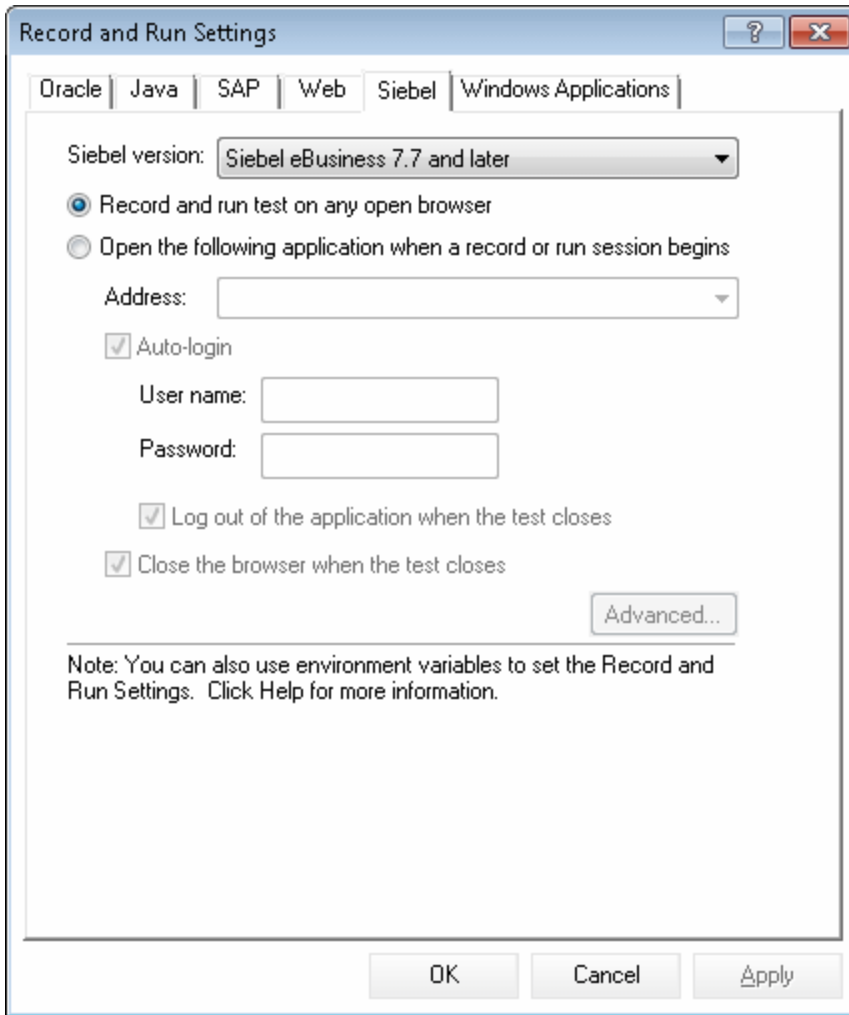
Use the variable names listed in the following table to define Siebel application details:

Option	Variable Name	Description
Siebel version	APPLICATION_ENV	The Siebel version for the applications on which you want to record your test or business component. Possible values: <ul style="list-style-type: none">• 77• 7075 This option is available for tests and business components.
Address	URL_ENV	The URL of the application you want to open. This option is available only for tests.

Option	Variable Name	Description
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: <ul style="list-style-type: none">• 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: <ul style="list-style-type: none">• True• False

Siebel Tab (Record and Run Settings Dialog Box)

This tab enables you to define how UFT starts recording and running tests on Siebel objects.



<p>To access</p>	<p>Select Record > Record and Run Settings and select the Siebel tab.</p> <p>Note: 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).</p>
<p>Important information</p>	<ul style="list-style-type: none"> If you are working with Siebel 7.7.x or later applications, consider the following: <ul style="list-style-type: none"> 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 details, see "Siebel 7.7.x or Later - Test Automation Module Configuration" on page 330. If you select the Open the following application when a record or run session begins option, UFT 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

	<p>required parameter values as part of the application URL when you open the application.</p> <ul style="list-style-type: none"> If you select the Open the following application when a record or run session begins option, and specify the password in the Siebel automation access code box in the Advanced Siebel Record and Run Settings dialog box, UFT 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.
Relevant tasks	"How to Define Environment Variables for Siebel Applications" on page 331
See also	<ul style="list-style-type: none"> "Considerations - Siebel Add-in" on page 324 "How to Define Record and Run Settings for UFT Add-ins" on page 29

User interface elements are described below:

UI Elements	Description
Siebel version	<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 details, see "How to Define Environment Variables for Siebel Applications" on page 331.</p>
Record and run tests on any open browser	<p>Instructs UFT to use any Internet Explorer browser to record and run the test. UFT can record and run only browsers that are opened after UFT is opened. If you are using Siebel 7.7.x or later, make sure you specify the required test automation parameters, as described in "Siebel 7.7.x or Later - Test Automation Module Configuration" on page 330.</p>
Open the following application when a record or run session begins	<p>Instructs UFT to open the specified application when record or run sessions begin.</p>
Address (Enabled only when "Open the following application when a record or run session begins" is selected)	<p>Instructs UFT to open Internet Explorer to the specified URL. Recommended format: <host>/<application name>/start.swe</p> <div style="background-color: #f0f0f0; padding: 5px; margin: 5px 0;"> <p>Example: siebapp/callcenter_enu/start.swe</p> </div> <p>You can use an environment variable to specify the URL. For details, see "How to Define Environment Variables for Siebel Applications" on page 331.</p>
Auto-login (Enabled only when "Open the following application when a record or run session begins" is selected)	<p>Instructs UFT 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 details, see "How to Define Environment Variables for Siebel Applications" on page 331.</p>

UI Elements	Description
User (Enabled only when Auto-login is selected)	The user name used to log in to the specified application. You can use an environment variable to specify the user name. For details, see "How to Define Environment Variables for Siebel Applications" on page 331 .
Password (Enabled only when Auto-login is selected)	The password for the specified user name. You can use an environment variable to specify the password. For details, see "How to Define Environment Variables for Siebel Applications" on page 331 .
Log out of the application when the test closes (Enabled only when Auto-login is selected)	Instructs UFT 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. You can use an environment variable to specify the Log out setting. For details, see "How to Define Environment Variables for Siebel Applications" on page 331 .
Close the browser when the test closes (Enabled only when "Open the following application when a record or run session begins" is selected)	Instructs UFT 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.
Advanced (Enabled only when "Siebel version 7.7 and later" and "Open the following application when a record or run session begins" are selected)	Opens the Advanced Siebel Record and Run Settings dialog box, where you can specify the following options: <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. Default: 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.

Troubleshooting and 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 business 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" below](#)
- ["Siebel 7.0.x and 7.5.x " on the next page](#)

Siebel 7.7.x or Later

- Certain objects, methods, or properties may be available from within UFT even though they are not described in the documentation. This is because UFT 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 UFT 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 business 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 business 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 window cannot be used to add SiebList inner objects to the object repository.

- If a warning message opens while recording your test or business component, for example, if you insert invalid data, UFT 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 UFT 11.50 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.

- When using Siebel version 8.1.1.11, the **SiebCalculator.ClickKeys** method may not work when running the test of the application calculator

Workaround: Enter the value of the calculator directly into the edit field instead of using the **ClickKey/ClickKeys** method to enter the value.

Siebel 7.0.x and 7.5.x

Creating and Running Testing Documents

- UFT does not support recording on Siebel applications using keyboard shortcuts.

Workaround: Use the mouse to record on Siebel applications.

- UFT 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 Editor to manually edit the statement to scroll multiple rows.

- By default, UFT 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 last modified in 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), UFT 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 business 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.

- UFT 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, UFT 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, UFT does record the operations performed within this secondary table. These statements are not required in the test or business 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 business 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, UFT 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, UFT 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.

Siebel Test Express

This chapter includes:

- [Using Siebel Test Express to Generate or Update Shared Object Repositories](#) 340
- [How to Use Siebel Test Express to Generate or Update a Shared Object Repository](#) 340
- [Create Object Repository Wizard](#) 342
 - [Connection Information Page \(Create Object Repository Wizard\)](#) 342
 - [Screen Selection Page \(Create / Update Object Repository Wizard\)](#) 343
 - [Importing Test Objects Page \(Create Object Repository Wizard\)](#) 344
 - [Object Repository Created / Updated Page \(Create / Update Object Repository Wizard\)](#) 345

Using Siebel Test Express to Generate or Update Shared Object Repositories

If the Siebel Add-in is installed on UFT, you can use Siebel Test Express to automatically generate a new shared object repository, or to 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 an ALM 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 update 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 details on working with object repositories in general, see the *HP Unified Functional Testing User Guide*.

For details on creating and updating object repositories using Siebel Test Express, see ["How to Use Siebel Test Express to Generate or Update a Shared Object Repository"](#) below.

For details on the Create / Update Object Repository wizard, see ["Create Object Repository Wizard"](#) on [page 342](#).

How to Use Siebel Test Express to Generate or Update a Shared Object Repository

This task describes how to use Siebel Test Express to generate a new shared object repository for a Siebel application or to update an existing shared object repository.

This task includes the following steps:

- ["Prerequisites" on the next page](#)
- ["Create or update a shared object repository" on the next page](#)
- ["\(Optional\) Use the Object Repository Merge Tool to merge the updated Siebel object repository" on the next page](#)
- ["Save the shared object repository" on the next page](#)

1. Prerequisites

- 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.
- To work with Siebel Test Express in UFT, ensure that the Siebel Test Automation API version installed on your server is one that supports Siebel Test Express.

2. Create or update a shared object repository

- a. Select **Resources > Object Repository Manager**. The Object Repository Manager opens.
- b. (Optional) To update an existing object repository, open the object repository file you want to update in editable format.

Note: By default, the object repository file opens in read-only mode. To open it in editable format, either clear the **Open in read-only mode** check box in the Open Shared Object Repository window, or enable editing by selecting **File > Enable Editing** after you open the repository.

- c. Open the Create Object Repository Wizard, as described in "[Create Object Repository Wizard](#)" on the next page.
- d. Follow the steps of the wizard to create the new shared object repository. The wizard contains the following pages:
 - "[Connection Information Page \(Create Object Repository Wizard\)](#)" on the next page
 - "[Screen Selection Page \(Create / Update Object Repository Wizard\)](#)" on page 343
 - "[Importing Test Objects Page \(Create Object Repository Wizard\)](#)" on page 344
 - "[Object Repository Created / Updated Page \(Create / Update Object Repository Wizard\)](#)" on page 345

After the import process ends, the Object Repository Merge Tool opens. This may take a few minutes.

3. (Optional) Use the Object Repository Merge Tool to merge the updated Siebel object repository

Conflicts between objects in the primary and secondary repository files are resolved automatically by the Merge Tool according to the default resolution settings. After the merge, 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.

For details on working with the Object Repository Merge Tool, see the *HP Unified Functional Testing User Guide*.

4. Save the shared object repository

Save the shared object repository to the file system or to an ALM project. For details on working

with or saving shared object repositories, see the section on the Object Repository Manager in the *HP Unified Functional Testing User Guide*.

Create Object Repository Wizard

This wizard enables you to create or update a shared object repository using Siebel Test Express.

To access	<p>In the Object Repository Manager (Resources > Object Repository Manager), do one of the following:</p> <p>To create a new object repository:</p> <ul style="list-style-type: none"> Click the Create Object Repository button  on the Object Repository Manager toolbar. Select Tools > Siebel Test Express > Create Object Repository. <p>To update an existing object repository:</p> <ul style="list-style-type: none"> Click the Update Object Repository button  on the Object Repository Manager toolbar. Select Tools > Siebel Test Express > Update Object Repository.
Important information	<p>You can run only one instance of the Create / Update Object Repository Wizard on a computer at any given time.</p>
Relevant tasks	<p>"How to Use Siebel Test Express to Generate or Update a Shared Object Repository" on page 340</p>
Wizard map	<p>This wizard contains:</p> <p>"Connection Information Page (Create Object Repository Wizard)" (page 342 > "Screen Selection Page (Create / Update Object Repository Wizard)" (page 343 > "Importing Test Objects Page (Create Object Repository Wizard)" (page 344 > "Object Repository Created / Updated Page (Create / Update Object Repository Wizard)" (page 345</p>
See also	<p>"Using Siebel Test Express to Generate or Update Shared Object Repositories" on page 340</p>

Connection Information Page (Create Object Repository Wizard)

This wizard page enables you to enter the connection information for logging in to the Siebel server.

Important information	<ul style="list-style-type: none"> General information about the wizard is available here: "Create Object Repository Wizard" above The information you enter on this page is saved as metadata in the generated object repository file. If you are creating a new repository, the information that was entered in this page the last time you used the wizard is automatically entered in this page. If you are updating the repository, the information that was saved as metadata with the repository file is automatically entered in this page. The data required in this page is not necessarily the same as the data you use to log into the Siebel application as a user. Contact your Siebel server administrator for details. While the Connection Information page is open, you cannot make the Object Repository Manager or UFT window active.
------------------------------	---

Wizard map	The " Create Object Repository Wizard " contains: Connection Information Page (Create Object Repository Wizard) > Screen Selection Page (page 343 > "Importing Test Objects Page (Create Object Repository Wizard)" (page 344) > "Object Repository Created / Updated Page (Create / Update Object Repository Wizard)" (page 345)
See also	<ul style="list-style-type: none"> • "How to Use Siebel Test Express to Generate or Update a Shared Object Repository" on page 340 • "Using Siebel Test Express to Generate or Update Shared Object Repositories" on page 340

User interface elements are described belows:

UI Element	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.

Screen Selection Page (Create / Update Object Repository Wizard)

This wizard page enables you to select the test objects whose children you want to import. 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.

Important information	<ul style="list-style-type: none"> • General information about the wizard is available here: "Create Object Repository Wizard" on the previous page. • While the Screen Selection page is open, you cannot make the Object Repository Manager or UFT window active. • When Siebel Test Express creates the object repository, it imports the metadata and creates test objects for all children (descendants) of the applications or top-level objects you select in this page. Therefore, it is recommended to select only the top-level objects that you need. Importing an entire application may take a very long time.
Wizard map	The " Create Object Repository Wizard " contains: "Connection Information Page (Create Object Repository Wizard)" (page 342) > Screen Selection Page (Create / Update Object Repository Wizard) > Importing Test Objects Page (page 344) > "Object Repository Created / Updated Page (Create / Update Object Repository Wizard)" (page 345)
See also	<ul style="list-style-type: none"> • "How to Use Siebel Test Express to Generate or Update a Shared Object Repository" on page 340 • "Using Siebel Test Express to Generate or Update Shared Object Repositories" on page 340

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

UI Elements	Description
<test object tree>	<p>The list of all the available applications and their objects, according to the connection information entered in the "Connection Information Page (Create Object Repository Wizard)", displayed as nodes in a tree.</p> <div style="background-color: #f0f0f0; padding: 10px; margin-top: 10px;"> <p>Note: (for updating an object repository)</p> <ul style="list-style-type: none"> 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. 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. </div>
Import only objects modified since (Update Object Repository Wizard only)	<p>The date in which the last import was performed. You can keep the displayed date or select a new date by clicking the drop-down arrow, 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.</p>

Importing Test Objects Page (Create Object Repository Wizard)

This wizard page 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.

Important information	<ul style="list-style-type: none"> General information about the wizard is available here: "Create Object Repository Wizard" on page 342. Importing into the object repository can take up to several hours, depending on the size of the repository. 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. You can work in either the Object Repository Manager or UFT while the wizard is generating the object repository. However, do not close either window. If you do try to close either window, a message is displayed, warning that the object repository generation process will be stopped and all data will be lost. If you are using the wizard to update an object repository, that object repository file is locked and you cannot modify it in the Object Repository Manager. 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. You can then select whether to keep, discard, or continue importing test objects into the partial object repository.
Wizard map	<p>The "Create Object Repository Wizard" contains:</p> <p>"Connection Information Page (Create Object Repository Wizard)" (342) > Screen Selection Page"Screen Selection Page (Create / Update Object Repository Wizard)" (343) > Importing Test Objects Page (Create Object Repository Wizard > Object Repository Created / Updated Page"Object Repository Created / Updated Page (Create / Update Object Repository Wizard)" (page 345)</p>

See also	<ul style="list-style-type: none"> • "How to Use Siebel Test Express to Generate or Update a Shared Object Repository" on page 340 • "Using Siebel Test Express to Generate or Update Shared Object Repositories" on page 340
-----------------	---

Object Repository Created / Updated Page (Create / Update Object Repository Wizard)

This wizard page opens after all the objects have been imported, displaying the total number of objects added or modified in the object repository.

Important information	<ul style="list-style-type: none"> • General information about the wizard is available here: "Create Object Repository Wizard" on page 342. • Displaying the new or updated object repository in the Object Repository Manager may take a few minutes. • If any errors occur during the import process, this page displays a warning and an Error Log button. The log contains error and exception data from the Siebel server, listing the failed calls and the object that caused the error. You can 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 <Unified Functional Testing>\Tests folder.
Wizard map	<p>The "Create Object Repository Wizard" contains:</p> <p>"Connection Information Page (Create Object Repository Wizard)" (342) > "Screen Selection Page (Create / Update Object Repository Wizard)" (page 343) > Importing Test Objects Page Importing Test Objects Page (Create Object Repository Wizard)" (page 344 > Object Repository Created / Updated Page (Create / Update Object Repository Wizard)</p>
See also	<ul style="list-style-type: none"> • "How to Use Siebel Test Express to Generate or Update a Shared Object Repository" on page 340 • "Using Siebel Test Express to Generate or Update Shared Object Repositories" on page 340

Part 14: Standard Windows Testing Support

This section includes:

["Standard Windows Support -Quick Reference" on page 347](#)

["Troubleshooting and Limitations - Standard Windows" on page 349](#)

Standard Windows Support -Quick Reference

You can use the standard Windows testing support provided by UFT to test user-interface objects (controls) developed using the Win32 API or MFC platforms. UFT standard Windows testing support is built-in and does not require you to load any UFT add-in.

The following tables summarize basic information about standard Windows testing support and how it relates to some commonly-used aspects of UFT.

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 "Windows-Based Application Support" on page 81
Important Information	UFT uses built-in standard Windows testing support and standard Windows test objects to identify the following: <ul style="list-style-type: none"> • Objects from other environments if the relevant add-in is not installed and loaded. • Stingray, VisualAge Smalltalk, and Qt (widget toolkit) controls when the relevant add-in is installed and loaded. For details, see the relevant add-in documentation. • Many windowless objects, if they were developed using the MSAA (Microsoft Active Accessibility) API. For example, the controls within the Microsoft Office ribbons are identified as independent objects.
Test Object Methods and Properties	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 <i>HP UFT Object Model Reference for GUI Testing</i> .
Checkpoints and Output Values	<ul style="list-style-type: none"> • See the sections describing checkpoints and output values in the <i>HP Unified Functional Testing User Guide</i>. • See "Supported Checkpoints " on page 513 and "Supported Output Values " on page 515 Per Add-In. • See "GUI Checkpoints and Output Values Per Add-in" on page 512
Troubleshooting and Limitations	"Troubleshooting and Limitations - Standard Windows" on page 349

Prerequisites	
Opening Your Application	You can open your standard Windows application before or after opening UFT. Standard Windows testing support is always loaded in UFT. It is therefore not an available option in the Add-in Manager.
Add-in Dependencies	None

Configuration	
Options Dialog Box	Use the Windows Applications pane. (Tools > Options > GUI Testing tab > Windows Applications node)

	See " Windows Applications > General Pane (Options Dialog Box > GUI Testing Tab) " on page 85.
Record and Run Settings Dialog Box (tests only)	Use the Windows Applications tab. (Record > Record and Run Settings) See " Windows Applications Tab (Record and Run Settings Dialog Box) " on page 93. Note: UFT 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 > GUI Testing tab > Active Screen node > Custom Level) See the section on the Custom Active Screen Capture Settings dialog box in the <i>HP Unified Functional Testing User Guide</i> .
Application Area Additional Settings pane (business components only)	Use the Applications pane. In the application area, select Additional Settings > Applications in the sidebar. See the section on the Applications pane in the <i>HP Unified Functional Testing User Guide</i> .

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, UFT 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.
- Windowless objects developed using an API other than the MSAA API are not identified.
- The description properties of a windowless control must include the **acc_name** property. By default, this property is not available in the list properties when you add a new test object.

Workaround: Add the **acc_name** property to the list of properties. To do this from the Define New Test Object Dialog Box, in the Test object details area, click the **Add description properties** button . In the Add Properties dialog box, click the **Define new property** button  and add the **acc_name** property.

Part 15: Stingray Add-in

This section includes:

["Stingray Add-in - Quick Reference" on page 351](#)

["Stingray Add-in - Testing and Configuration" on page 353](#)

Stingray Add-in - Quick Reference

The following tables summarize basic information about the Stingray Add-in and how it relates to some commonly-used aspects of UFT.

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 " Windows-Based Application Support " on page 81.
Supported Environments	The UFT 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 <i>HP Unified Functional Testing Product Availability Matrix</i> , available from the HP Support Matrix page (requires an HP passport).
Important Information	" Considerations for Working with the Stingray Add-in " on page 354
Test Object Methods and Properties	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 details, see the Stingray section of the <i>HP UFT Object Model Reference for GUI Testing</i> .
Checkpoints and Output Values	<ul style="list-style-type: none"> See the sections describing checkpoints and output values in the <i>HP Unified Functional Testing User Guide</i>. See "GUI Checkpoints and Output Values Per Add-in" on page 512
Troubleshooting and Limitations	" Troubleshooting and Limitations - Stingray Add-in " on page 374

Prerequisites	
Opening Your Application	You can open your Stingray application before or after opening UFT.
Add-in Dependencies	None
Other	You must configure the Stingray Add-in to work with your application. See " Setting Up Stingray Object Support " on page 354.

Configuration	
Wizard	" Stingray Support Configuration Wizard " on page 358
Options Dialog Box	<ul style="list-style-type: none"> Use the Stingray pane. (Make sure that a GUI test is open and select Tools > Options > GUI Testing tab > Stingray node.) See "Stingray Pane (Options Dialog Box > GUI Testing Tab)" on page 371. Use the Windows Applications pane. (Tools > Options > GUI Testing tab > Windows Applications node)

	See " Windows Applications > General Pane (Options Dialog Box > GUI Testing Tab) " on page 85.
Record and Run Settings Dialog Box (tests only)	Use the Windows Applications tab. (Record > Record and Run Settings) See " Windows Applications Tab (Record and Run Settings Dialog Box) " on page 93. <div style="background-color: #f0f0f0; padding: 10px;"> <p>Note:</p> <ul style="list-style-type: none"> • In addition to the settings in the Record and Run Settings dialog box, you must also configure UFT to recognize your Stingray applications in the Stingray pane of the Options dialog box (Tools > Options > GUI Testing tab > Stingray node). For details, see "Stingray Pane (Options Dialog Box > GUI Testing Tab)" on page 371. • 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. </div>
Custom Active Screen Capture Settings Dialog Box (tests only)	Use the Windows applications section. (Tools > Options > GUI Testing tab > Active Screen node > Custom Level) See the section on the Custom Active Screen Capture Settings dialog box in the <i>HP Unified Functional Testing User Guide</i> .
Application Area Additional Settings pane (business components only)	Use the Applications pane. In the application area, click Additional Settings > Applications in the sidebar. See the section on the Applications pane in the <i>HP Unified Functional Testing User Guide</i> .

Stingray Add-in - Testing and Configuration

This chapter includes:

- [Setting Up Stingray Object Support](#) 354
- [Considerations for Working with the Stingray Add-in](#) 354
- [Stingray Run-time Agent \(Agent DLL\)](#) 354
- [Stingray Precompiled Agent Mode](#) 355
- [How to Set Up Your Stingray Project Using the Precompiled Agent Mode](#) 356
- [Stingray Support Configuration Wizard](#) 358
 - [Support Mode Selection Page \(Stingray Support Configuration Wizard\)](#) 359
 - [Add Support Code Page \(Stingray Support Configuration Wizard\)](#) 360
 - [Select Configuration Mode Page \(Stingray Support Configuration Wizard\)](#) 362
 - [Manual Configuration Mode Page \(Stingray Support Configuration Wizard\)](#) 364
 - [Automatic Configuration Mode Page \(Stingray Support Configuration Wizard\)](#) 365
 - [Detected Stingray Components Page \(Stingray Support Configuration Wizard\)](#) 367
 - [Finish Page \(Stingray Support Configuration Wizard\)](#) 369
 - [Detection Failure Page \(Stingray Support Configuration Wizard\)](#) 370
- [Stingray Pane \(Options Dialog Box > GUI Testing Tab\)](#) 371
 - [Record Cell Editing Options - Example](#) 374
- [Troubleshooting and Limitations - Stingray Add-in](#) 374

Setting Up Stingray Object Support

Before you begin working, you need to configure the Stingray Add-in to work with your application. UFT support for Stingray objects is based on an agent entity that exists in the Stingray application. This agent interacts with UFT to enable record and run operations. There are two different modes for establishing the agent entity:

- **Run-time Agent Mode** UFT injects an agent .dll into the application's process during run-time. This is the recommended mode. For details, see "[Stingray Run-time Agent \(Agent DLL\)](#)" below.
- **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 details, see "[Stingray Precompiled Agent Mode](#)" on the next page.

You choose your preferred mode and configure support for the Stingray Add-in using the Stingray Support Configuration Wizard. For details, see "[Stingray Support Configuration Wizard](#)" on page 358.

After you configure support for the Stingray Add-in, you can fine-tune the configuration options, if needed. For details, see "[Stingray Pane \(Options Dialog Box > GUI Testing Tab\)](#)" on page 371.

Considerations for Working with the Stingray Add-in

UFT stores Stingray support configuration for each configured Stingray application separately. By default, UFT 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 UFT to support both applications as follows:

1. Run the Stingray Support Configuration Wizard and configure support for the `grid1.exe` application. UFT saves the configuration for this application.
2. Run the Stingray Support Configuration Wizard again and configure support for the `tree1.exe` application. UFT saves the configuration for this application.

After performing these steps, UFT will support the `grid1.exe` application and support all Stingray applications that have Stingray TreeView controls version 11.00, including the `tree1.exe` application.

Stingray Run-time Agent (Agent DLL)

When you choose the run-time agent mode, UFT 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 Unified Functional Testing Product Availability Matrix*, available from the [HP Support Matrix page](#) (requires an HP passport). 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 details, see "[Stingray Precompiled Agent Mode](#)" below, or contact HP Software Support.

Stingray 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 UFT 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.

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.

How to Set Up Your Stingray Project Using the Precompiled Agent Mode

Note:

- Use the precompiled agent mode only if the run-time agent mode is unsuitable or cannot be used.
- For a conceptual overview, see ["Stingray Precompiled Agent Mode" on the previous page.](#)

This task includes the following steps:

- ["Prerequisites" below](#)
- ["Copy StgAgentLib.h and StgAgentLib.lib files" below](#)
- ["Add #include "StgAgentLib.h" to a .cpp file" on the next page](#)
- ["Add the ReleaseWRVC\(\); function call" on the next page](#)
- ["Make sure that the Precompiled Agent option is selected in the Stingray Support Configuration Wizard" on the next page](#)
- ["Results" on the next page](#)

1. Prerequisites

- 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 Unified Functional Testing Product Availability Matrix*, available from the [HP Support Matrix page](#) (requires an HP passport).

Note: If you do not have the required Stingray Objective Grid and Stingray Objective Toolkit version combination, contact HP Software Support for assistance.

- If your Stingray application was previously compiled with agent files from an earlier version of the Stingray Add-in, remove the existing agent files from your project.

Caution: 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 UFT Stingray Add-in, you will not be able to take advantage of the latest functionality.

2. Copy StgAgentLib.h and StgAgentLib.lib files

- a. Copy the StgAgentLib.h header file from <UFT 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.)

- b. 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:

<UFT 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: <UFT

InstallationFolder>\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 Unified Functional Testing Product Availability Matrix*, available from the [HP Support Matrix page](#) (requires an HP passport).

- c. Copy the StgAgentLib.lib support library file to your Visual C++ project directory.

3. **Add #include "StgAgentLib.h" to a .cpp file**

Add the #include "StgAgentLib.h" statement to one of your cpp files, such as, MainFrm.cpp.

4. **Add the ReleaseWRVC(); function call**

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.

5. **Make sure that the Precompiled Agent option is selected in the Stingray Support Configuration Wizard**

For details, see "Stingray Support Configuration Wizard" on the next page.

6. **Results**

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.

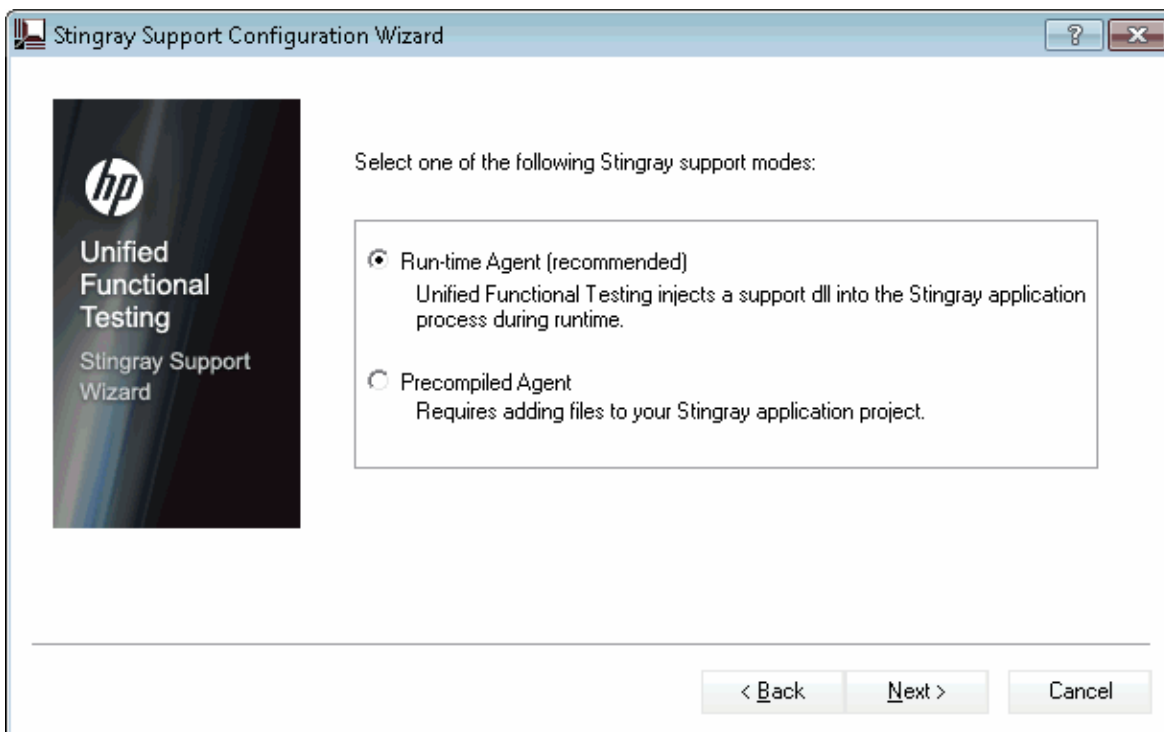
Stingray Support Configuration Wizard

The wizard guides you through the steps that are necessary to configure UFT to work according to the agent mode that you select.

To access	<p>Use one of the following:</p> <ul style="list-style-type: none"> • Select Tools > Options > GUI Testing tab > Stingray node > Select Version button. • From the Start menu, select All Programs > HP Software > HP Unified Functional Testing > Tools > Stingray Support Configuration Wizard. • The <UFT installation>\bin\StgAgentConfig.exe file <p>Note: For details on accessing UFT and UFT tools and files in Windows 8 and Windows Server 2012, see "Accessing UFT in Windows 8.X or Higher Operating Systems" on page 517.</p>
Important information	<p>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 UFT program group or by activating it from the Stingray pane of the Options dialog box (Tools > Options > GUI Testing tab > Stingray node). For details, see "Stingray Pane (Options Dialog Box > GUI Testing Tab)" on page 371.</p>
Relevant tasks	<p>"How to Set Up Your Stingray Project Using the Precompiled Agent Mode" on page 356</p>
Wizard map	<p>The Stingray Support Configuration Wizard contains:</p> <p>Welcome Page > Support Mode Selection Page (page 359 > Add Support Code Page (page 360 > (Select Configuration Mode Page (page 362) > (Manual Configuration Mode Page (page 364) > (Automatic Configuration Mode Page (page 365) > (Detected Stingray Components Page (page 367) > (Finish Page (page 369) > (Detection Failure Page (page 370)</p> <p>Note: Pages that are in parentheses open according to the option selected in the previous page. Therefore, not all pages are displayed for every option.</p>
See also	<ul style="list-style-type: none"> • "Setting Up Stingray Object Support" on page 354 • "Stingray Run-time Agent (Agent DLL)" on page 354 • "Stingray Precompiled Agent Mode" on page 355

Support Mode Selection Page (Stingray Support Configuration Wizard)

This wizard page enables you to select a Stingray support mode.



Important information	General information about this wizard is available here: "Stingray Support Configuration Wizard" on the previous page.
Wizard map	<p>The "Stingray Support Configuration Wizard" contains:</p> <p>Welcome Page > Support Mode Selection Page > Add Support Code Page (page 360) > (Select Configuration Mode Page (page 362) > (Manual Configuration Mode Page (page 364) > (Automatic Configuration Mode Page (page 365) > (Detected Stingray Components Page (page 367) > (Finish Page (page 369) > (Detection Failure Page (page 370)</p> <p>Note: Pages that are in parentheses open according to the option selected in the previous page. Therefore, not all pages are displayed for every option.</p>
See also	<ul style="list-style-type: none"> • "Setting Up Stingray Object Support" on page 354 • "Stingray Run-time Agent (Agent DLL)" on page 354 • "Stingray Precompiled Agent Mode" on page 355

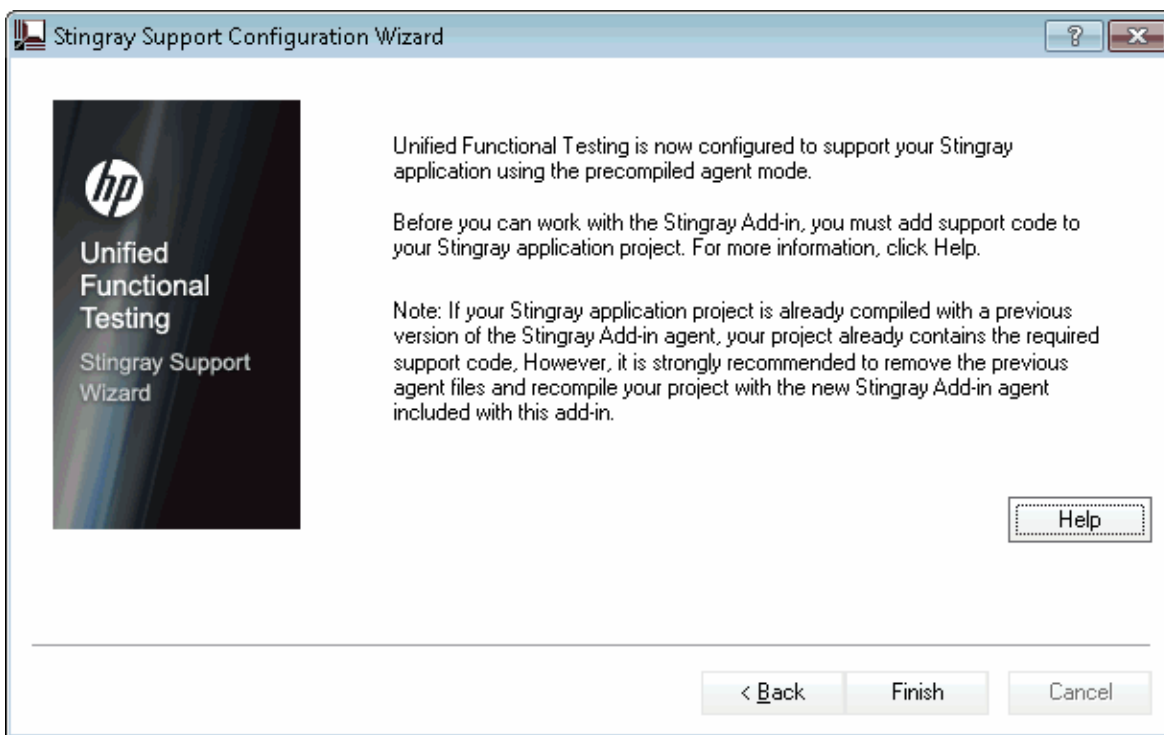
User interface elements are described below:

UI Elements	Description
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.
Precompiled Agent	A mode that requires you to make slight modifications to your Stingray application project so that UFT can support your Stingray application. If you select this option and click Next , the Finish page opens.

Add Support Code Page (Stingray Support Configuration Wizard)

This wizard page enables you to access information describing how to add support code to your Stingray application project.

This wizard page opens if you selected the **Precompiled Agent** option in the [Support Mode Selection Page](#) (described on page 362).



Important information	<ul style="list-style-type: none"> General information about this wizard is available here: "Stingray Support Configuration Wizard" on page 358. If your Stingray application project is already compiled with an earlier version of the Stingray Add-in agent, your project already contains the required support code. However, it is strongly recommended to remove the previous agent files and to recompile your project with the new Stingray Add-in agent included with this add-in.
Wizard map	The "Stingray Support Configuration Wizard" contains:

	<p>Welcome Page > Support Mode Selection Page (page 359 > Add Support Code Page > Select Configuration Mode Page (page 362)) > (Manual Configuration Mode Page (364) > (Automatic Configuration Mode Page (365)) > (Detected Stingray Components Page (page 367) > (Finish Page (page 369) > (Detection Failure Page (page 370))</p> <p>Note: Pages that are in parentheses open according to the option selected in the previous page. Therefore, not all pages are displayed for every option.</p>
See also	"Stingray Precompiled Agent Mode" on page 355

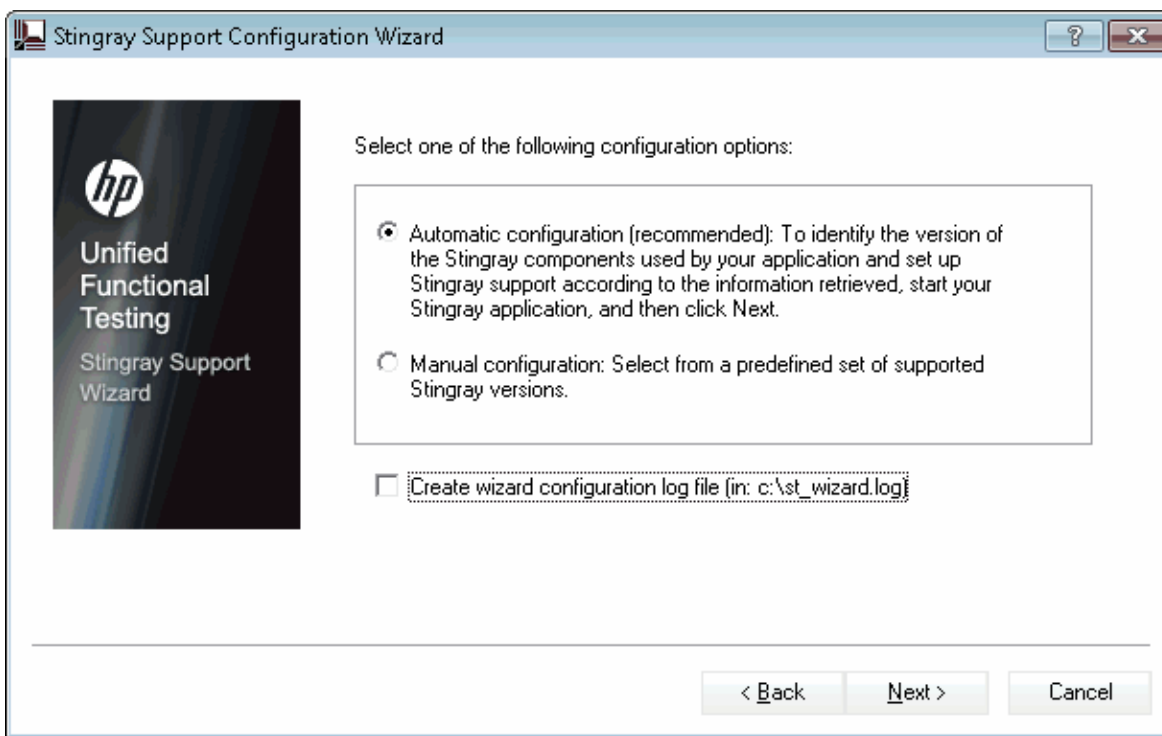
User interface elements are described below:

UI Elements	Description
Help	Displays information describing how to add support code to your Stingray application project.
Finish	Closes 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 UFT Stingray Add-in.

Select Configuration Mode Page (Stingray Support Configuration Wizard)

This wizard page enables you to select a configuration mode—either automatic or manual—and to create a diagnostic log file, when needed.

This wizard page opens if you selected the **Run-time Agent** option in the [Support Mode Selection Page](#) (described on page 359).



Important information	<ul style="list-style-type: none">• General information about this wizard is available here: "Stingray Support Configuration Wizard" on page 358.• If your Stingray application project is already compiled with an earlier version of the Stingray Add-in agent, your project already contains the required support code. However, it is strongly recommended to remove the previous agent files and to recompile your project with the new Stingray Add-in agent included with this add-in.
Wizard map	<p>The "Stingray Support Configuration Wizard" contains:</p> <p>Welcome Page > Support Mode Selection Page (page 359) > Add Support Code Page (page 360) > (Select Configuration Mode Page) > (Manual Configuration Mode Page (page 364)) > (Automatic Configuration Mode Page (page 365)) > (Detected Stingray Components Page (page 367)) > (Finish Page (page 369)) > (Detection Failure Page (page 370))</p> <p>Note: Pages that are in parentheses open according to the option selected in the previous page. Therefore, not all pages are displayed for every option.</p>

See also	"Setting Up Stingray Object Support" on page 354
-----------------	--

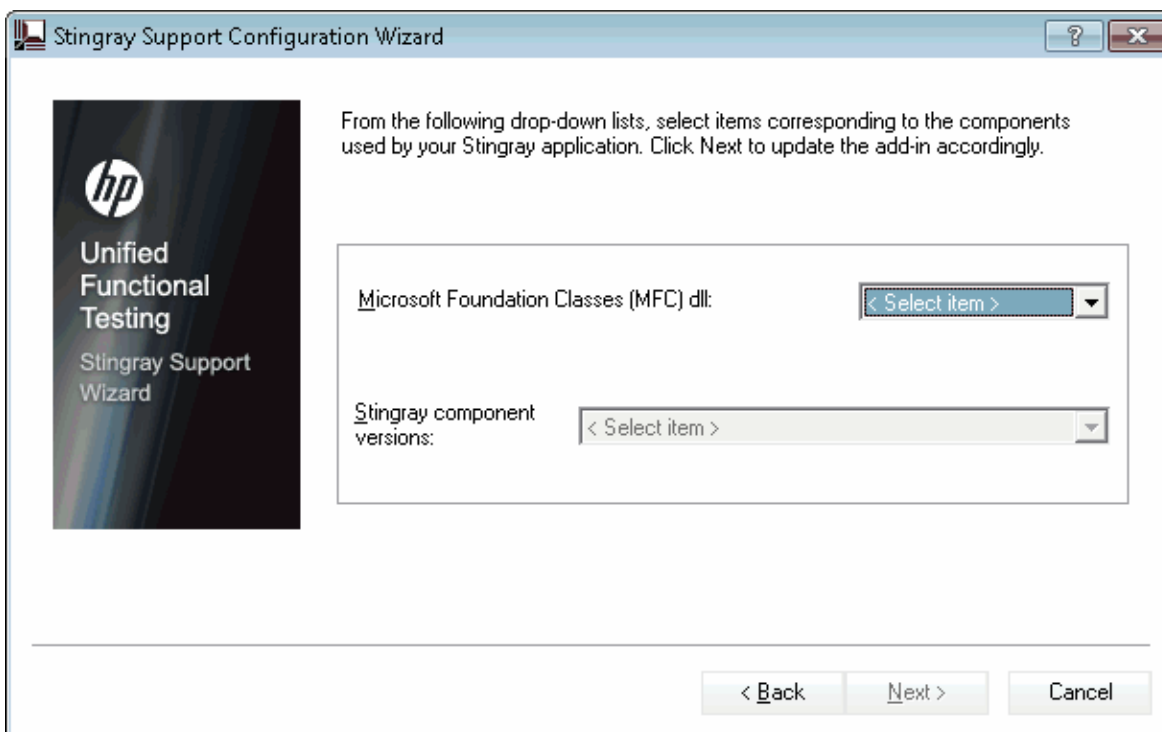
User interface elements are described below:

UI Elements	Description
Automatic configuration	Instructs the wizard to configure Stingray support automatically according to the detected MFC DLL and the version of Stingray business components used in your application.
Manual configuration	Enables you to configure Stingray support manually by specifying the MFC DLL and Stingray business component version used in your application. This is useful, for example, if your application is statically linked to the Stingray libraries.
Create wizard configuration log file	<p>Creates a diagnostic log file.</p> <p>Select this check box if, at any time, you encounter problems with the Stingray Add-in.</p> <p>Note: If you contact HP Software Support for assistance, you may be asked to provide this log file for diagnostic purposes.</p>

Manual Configuration Mode Page (Stingray Support Configuration Wizard)

This wizard page enables you to configure Stingray support manually by specifying the MFC DLL and Stingray business component version used in your application. This is useful, for example, if your application is statically linked to the Stingray libraries.

This wizard page opens if you selected the **Manual configuration** option in the [Select Configuration Mode Page](#) (described on page 362).



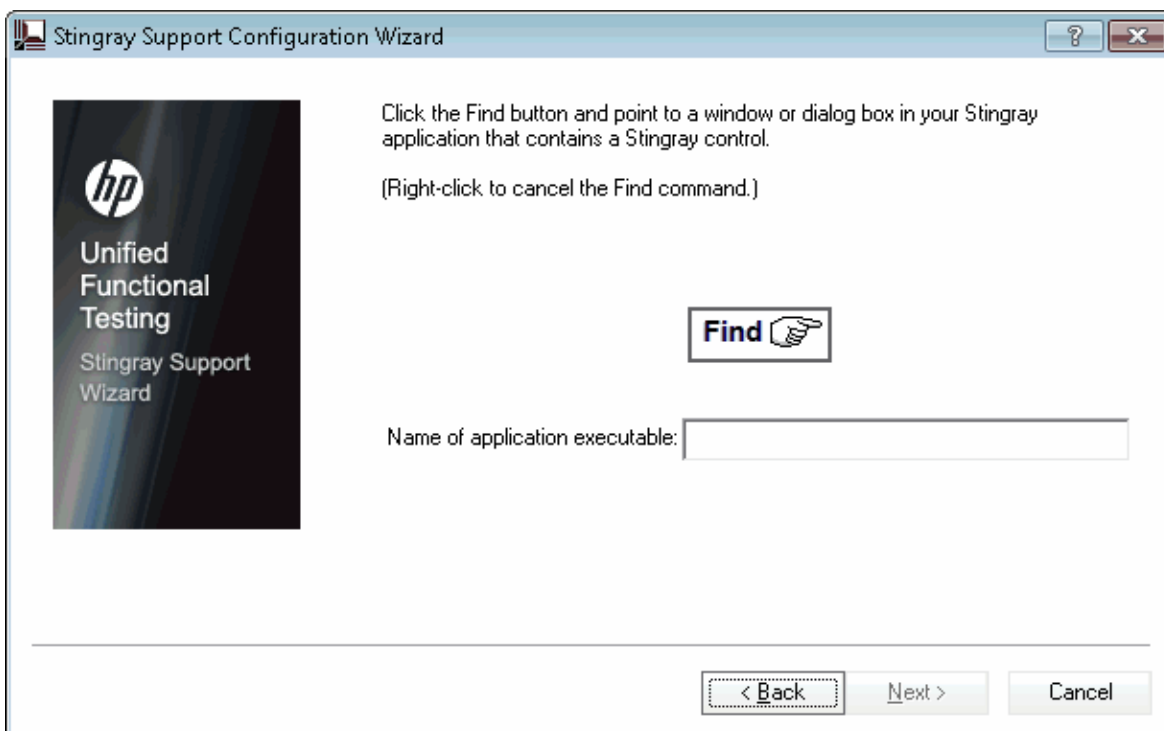
<p>Important information</p>	<ul style="list-style-type: none"> • General information about this wizard is available here: "Stingray Support Configuration Wizard" on page 358. • If your Stingray application project is already compiled with an earlier version of the Stingray Add-in agent, your project already contains the required support code. However, it is strongly recommended to remove the previous agent files and to recompile your project with the new Stingray Add-in agent included with this add-in.
<p>Wizard map</p>	<p>The "Stingray Support Configuration Wizard" contains:</p> <p>Welcome Page > Support Mode Selection Page (page 359 > Add Support Code Page (page 360 > (Select Configuration Mode Page (page 362) > (Manual Configuration Mode Page) > (Automatic Configuration Mode Page (page 365) > (Detected Stingray Components Page (page 367) > (Finish Page (page 369) > (Detection Failure Page (page 370))</p> <p>Note: Pages that are in parentheses open according to the option selected in the previous page.</p>

	Therefore, not all pages are displayed for every option.
See also	"Setting Up Stingray Object Support" on page 354

Automatic Configuration Mode Page (Stingray Support Configuration Wizard)

This wizard page enables you to instruct to identify the version of Stingray business components used by your application so that the wizard can set up Stingray support accordingly.

This wizard page opens if you selected the **Automatic configuration** option in the [Select Configuration Mode Page](#) (described on page 364).



Important information	General information about this wizard is available here: "Stingray Support Configuration Wizard" on page 358.
Wizard map	<p>The "Stingray Support Configuration Wizard" contains:</p> <p>Welcome Page > Support Mode Selection Page (359) > Add Support Code Page (page 360) > (Select Configuration Mode Page (page 362) > (Manual Configuration Mode Page (page 364)) > (Automatic Configuration Mode Page) > (Detected Stingray Components Page (page 367)) > (Finish Page (369)) > (Detection Failure Page (page 370))</p> <p>Note: Pages that are in parentheses open according to the option selected in the previous page. Therefore, not all pages are displayed for every option.</p>

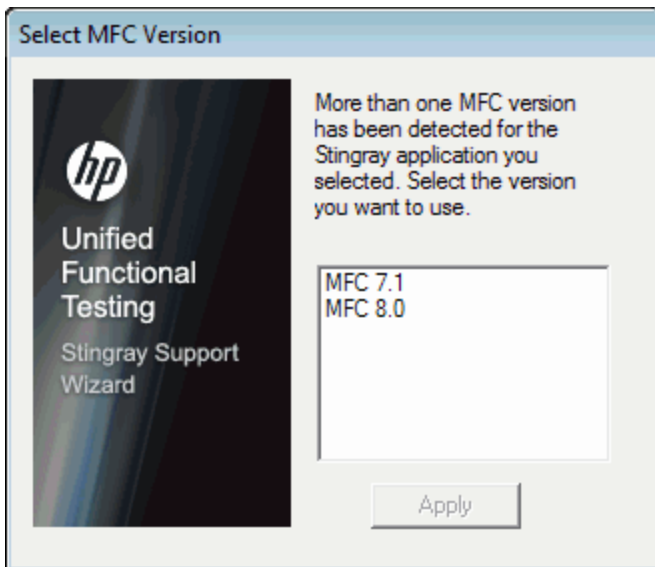
See also ["Setting Up Stingray Object Support" on page 354](#)

User interface elements are described below:

UI Elements	Description
Find	<p>Enables you to specify the application executable. Click the Find button and point to a window or dialog box in your application that contains a Stingray control. UFT automatically detects and displays the executable in the Name of application executable box.</p> <p>Note:</p> <ul style="list-style-type: none">• If UFT detects more than one MFC version, you must select the relevant version in the dialog box that opens. For details, see "Multiple MFC Versions", below.• You can right-click at any time to cancel the Find command. <p>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.</p>

Multiple MFC Versions

If UFT detects more than one MFC version for the Stingray application, the following dialog box opens:

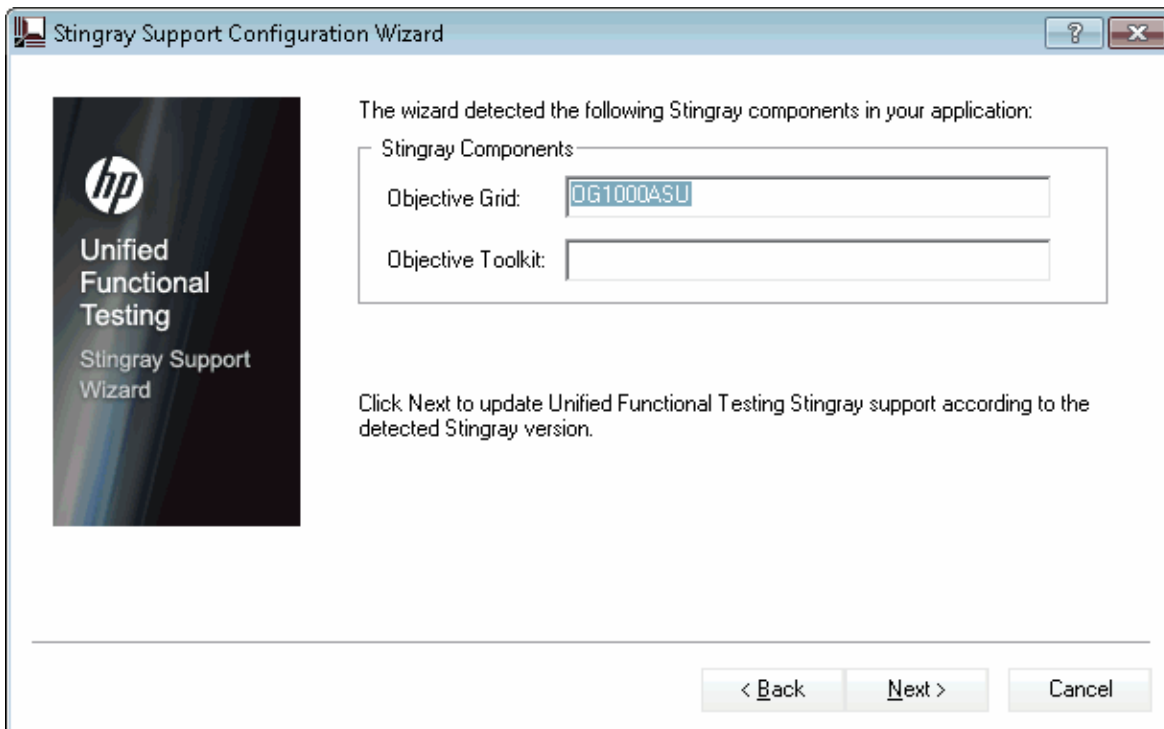


Select the relevant version and click **Apply**.

Detected Stingray Components Page (Stingray Support Configuration Wizard)

This wizard page displays the Stingray business components it detected in your application process.

This wizard page opens if the wizard successfully identified the application executable in the [Automatic Configuration Mode Page](#) (described on page 365).



Important information	General information about this wizard is available here: "Stingray Support Configuration Wizard" on page 358.
Wizard map	<p>The "Stingray Support Configuration Wizard" contains:</p> <p>Welcome Page > Support Mode Selection Page (page 359) > Add Support Code Page (page 360) > (Select Configuration Mode Page (362)) > (Manual Configuration Mode Page (page 364)) > (Automatic Configuration Mode Page (365)) > (Detected Stingray Components Page) > (Finish Page (page 369)) > (Detection Failure Page (page 370))</p> <p>Note: Pages that are in parentheses open according to the option selected in the previous page. Therefore, not all pages are displayed for every option.</p>
See also	"Setting Up Stingray Object Support" on page 354

User interface elements are described below:

UI Elements	Description
<p><Stingray business components></p>	<p>The Objective Grid and Objective Toolkit that the wizard detected in your application.</p> <p>Note: If, in the previous screen, you pointed to a non-Stingray application, or to a Stingray application whose business components UFT could not detect, a warning message displays stating that UFT failed to detect the Stingray business components in your application.</p> <p>UFT may fail to detect business components of a Stingray application for several reasons, for example:</p> <ul style="list-style-type: none"> • 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 details, see "Manual Configuration Mode Page (Stingray Support Configuration Wizard) " on page 364. • The application may be statically linked to the Microsoft Foundation Class (MFC) libraries. In this case, click Back three times and select Precompiled Agent. For details, see "Add Support Code Page (Stingray Support Configuration Wizard) " on page 360. • The selected Stingray version may not be supported by the Stingray Add-in, or may be slightly different from the officially supported versions. <p>If you need to work 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.</p>

Finish Page (Stingray Support Configuration Wizard)

This wizard page indicates that UFT configures Stingray support according to the detected Stingray version.

This wizard page opens if the wizard successfully identified the Stingray business components in the [Detected Stingray Components Page](#) (described on page 367).



Important information	General information about this wizard is available here: " Stingray Support Configuration Wizard " on page 358 .
Wizard map	The " Stingray Support Configuration Wizard " contains: Welcome Page > Support Mode Selection Page (page 359 > Add Support Code Page (page 360 > (Select Configuration Mode Page (page 362) > (Manual Configuration Mode Page (page 364)) > (Automatic Configuration Mode Page (page 365) > (Detected Stingray Components Page (page 367) > (Finish Page) > (Detection Failure Page (page 370)) <div style="background-color: #f0f0f0; padding: 5px; margin-top: 10px;"> <p>Note: Pages that are in parentheses open according to the option selected in the previous page. Therefore, not all pages are displayed for every option.</p> </div>
See also	" Setting Up Stingray Object Support " on page 354

User interface elements are described below:

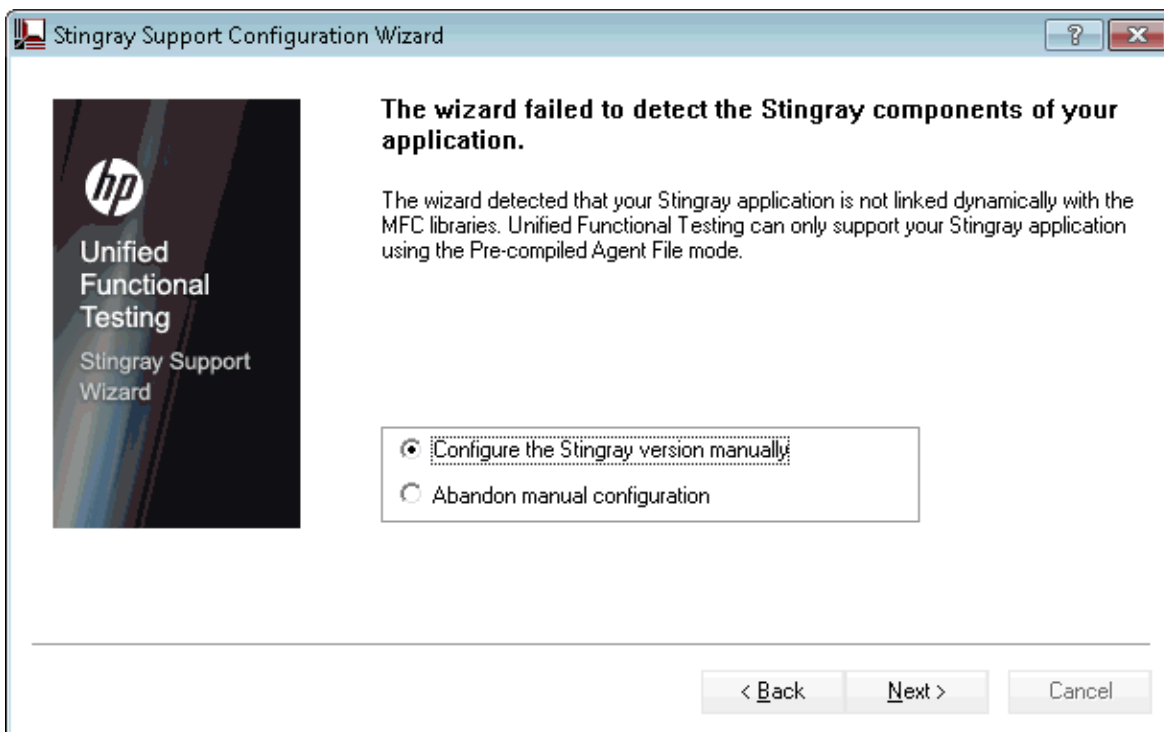
UI Elements	Description
<detected Stingray	Read-only display of the Stingray version detected using the Stingray Support Configuration Wizard.

UI Elements	Description
version>	
Apply to all users on this computer	Applies the identical settings you configured in this wizard for 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, this option is disabled.

Detection Failure Page (Stingray Support Configuration Wizard)

This wizard page indicates that UFT configures Stingray support according to the detected Stingray version.

This wizard page opens if the wizard cannot identify the Stingray business components in the [Detected Stingray Components Page](#) (described on page 367).



Important information	General information about this wizard is available here: "Stingray Support Configuration Wizard" on page 358.
Wizard map	The "Stingray Support Configuration Wizard" contains: Welcome Page > Support Mode Selection Page (page 359) > Add Support Code Page (360) > (Select Configuration Mode Page (page 362)) > (Manual Configuration Mode Page (page 364)) > (Automatic Configuration Mode Page (page 365)) > (Detected Stingray Components Page (page 367) > (Finish Page (page 369) > (Detection Failure Page)

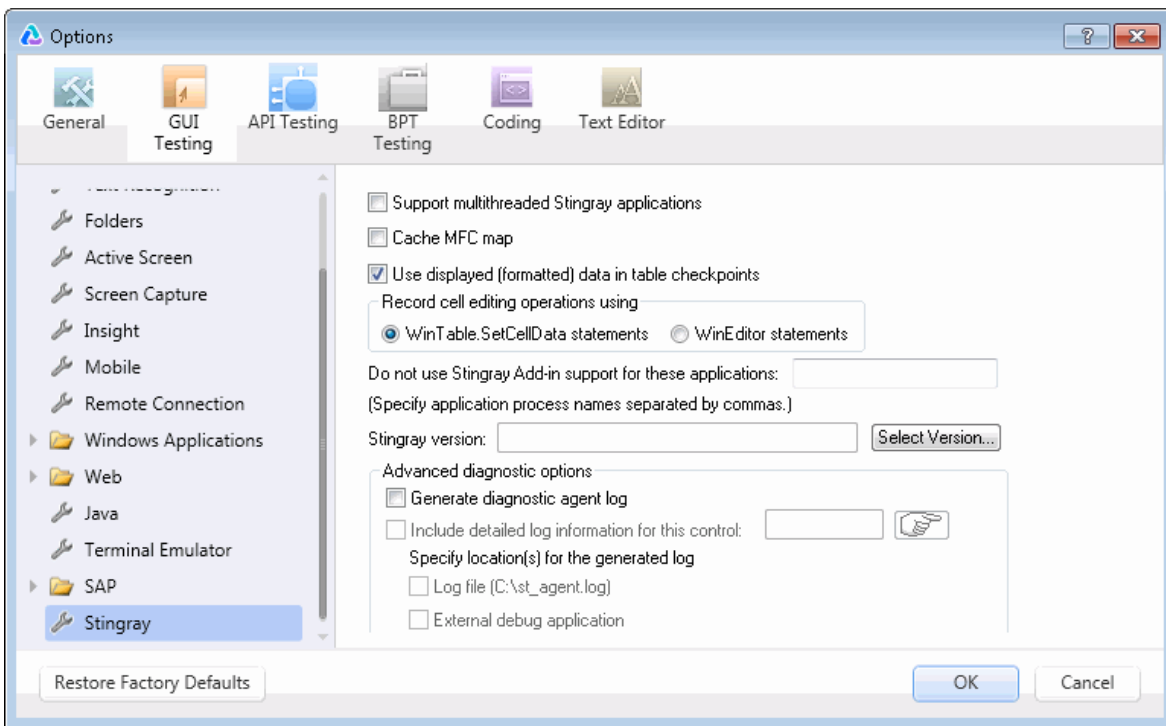
	Note: Pages that are in parentheses open according to the option selected in the previous page. Therefore, not all pages are displayed for every option.
See also	"Setting Up Stingray Object Support" on page 354

User interface elements are described below:

UI Elements	Description
Configure the Stingray version manually	Opens the "Manual Configuration Mode Page (Stingray Support Configuration Wizard)" (described on page 364) when you select this option and click Next .
Abandon manual configuration	Closes the Stingray Support Configuration Wizard when you select this option and click Finish .

Stingray Pane (Options Dialog Box > GUI Testing Tab)

This pane enables you to configure how UFT records and runs tests and business components on Stingray Objective Grid and Objective Toolkit objects.



To access	<ol style="list-style-type: none"> Do one of the following: <ul style="list-style-type: none"> Ensure that a GUI test, action, or component is in focus in the document pane. Ensure that a GUI test, action, or component is in focus in the document pane. Select Tools > Options > GUI Testing tab > Stingray node.
------------------	---

Important information	<ul style="list-style-type: none"> 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 business 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 modifying this pane, you must restart UFT before you continue working with this add-in. The Restore Factory Defaults button resets all Options dialog box options to their defaults.
Relevant tasks	"How to Set Up Your Stingray Project Using the Precompiled Agent Mode" on page 356
See also	"Stingray Support Configuration Wizard" on page 358

User interface elements are described below:

UI Elements	Description
Support multithreaded Stingray applications	<p>Instructs UFT 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.</p> <p>Select this check box only if you are working with a multithreaded application.</p> <p>By default, this check box is cleared.</p>
Cache MFC map	<p>Instructs UFT to use auxiliary caching as a backup for MFC's internal mapping of window handles to Visual C++ objects. If UFT cannot identify one or more Stingray controls while recording or running a test or business component, you can select this check box to instruct UFT to use a cached map instead of using the Stingray application for identification.</p> <p>By default, this check box is cleared.</p>
Use displayed (formatted) data in table checkpoints	<p>Instructs UFT to use the formatted data value in the Stingray grid control. You can use this option when working with table checkpoints (not supported for business 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, UFT will use that rounded number instead of the actual number when checking the value during the run session.</p> <p>By default, this check box is selected.</p>
Record cell editing options using	<p>Instructs UFT to record typing operations in a Stingray grid (edit) cell using one of the following options:</p> <ul style="list-style-type: none"> 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 business component. In most cases, this option makes the step more readable and easier to modify manually WinEditor statements. UFT records each operation that you perform in a Stingray grid edit cell as a separate <code>WinEditor</code> 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 business 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. <p>For a use-case example, see "Record Cell Editing Options - Example" on page 374</p>
Do not use Stingray Add-in support for these	<p>Instructs UFT to treat the applications you specify as non-Stingray applications.</p> <p>Some open, non-Stingray processes (such as <code>explorer.exe</code>) can cause unexpected behavior when recording and running tests and business components on Stingray applications. By adding the process names to this edit box, you can help prevent this unexpected behavior.</p>

UI Elements	Description
<p>applications: (Specify application process names separated by commas)</p>	<p>Note:</p> <ul style="list-style-type: none"> 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 (Record > Record and Run Settings). For details on the options available, see "Windows Applications Tab (Record and Run Settings Dialog Box)" on page 93.
<p>Stingray version</p>	<p>Indicates the versions of the Stingray Objective Grid and Stingray Objective Toolkit libraries used for identifying Stingray objects in your application (read-only).</p>
<p>Select Version</p>	<p>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.</p> <p>For details, see "Stingray Support Configuration Wizard" on page 358.</p>
<p>Generate diagnostic agent log</p>	<p>Instructs UFT to generate a diagnostic agent log file. You can use this option if you encounter problems with the Stingray Add-in, for example, if UFT 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.</p> <p>When you select this check box, the following options are enabled:</p> <ul style="list-style-type: none"> Include detailed log information for this control Log file (C:\st_agent.log) External debug application <p>Note: If you select this check box, you must specify the location for the generated log. For details, see "Specify location(s) for the generated log" below.</p>
<p>Include detailed log information for this control</p>	<p>Instructs UFT to include detailed information in the generated log for a specific Stingray control, in addition to the general UFT/agent communication log information. For example, you may want to generate additional log details for a specific Stingray grid.</p> <p>To select the object for which you want to generate detailed log information:</p> <p>Click the pointing hand and then click the relevant Stingray control. The selected object's window handle is displayed in the edit box.</p> <p>Note: This option is available only when the Generate diagnostic agent log check box is selected.</p>
<p>Specify location(s) for the generated log</p>	<p>Instructs UFT to generate the log to the selected locations. You can select one or both of the following options:</p> <ul style="list-style-type: none"> 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. <p>Note: These options are available only when the Generate diagnostic agent log check box is selected.</p>



Record Cell Editing Options - 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, UFT records the following in the Editor:

```
Window("GRIDAPP").Window("GridAp1").WinTable("StingrayGrid").SelectCell "#2",  
"#3"  
Window("GRIDAPP").Window("GridAp1").WinTable("StingrayGrid").SelectCell "#2",  
"#3", "abcde"
```






UFT 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	SelectCell	"#2", "#3", "abcde"	Use the "abcde" mouse button to select the cell in row "#2", colum...

If you are using the **WinEditor statements** option, UFT records the following in the Editor:

```
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"
```

UFT 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	SetCaret...	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.

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 UFT at least once.

Workaround: Apply Stingray Support Configuration settings separately for each user that has opened UFT at least once.

- UFT 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 single-threaded Stingray applications are supported.

To provide support for multithreaded applications, in UFT, select **Tools > Options > GUI Testing tab > Stingray** node. Select the **Support multithreaded Stingray applications** check box and click **OK**. Close and restart UFT.

For details, see "[Stingray Pane \(Options Dialog Box > GUI Testing Tab\)](#)" on page 371.

- The Stingray Add-in does not support Objective Edit or Objective Chart controls.
- The **ExpandAll** method is not supported for Stingray tree controls.
- 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.
- Sometimes, the MFC internal map that correlates a window handle of a control with the 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 UFT, select **Tools > Options > GUI Testing tab > Stingray** node. Select the **Cache MFC map** check box and click **OK**. Close and restart UFT.

Note: This mechanism is not activated by default because it imposes some performance overhead.

- 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 business 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 16: Terminal Emulator Add-in

This section includes:

["Terminal Emulator Add-in - Quick Reference" on page 378](#)

["Terminal Emulator Add-in - Testing and Configuration" on page 380](#)

Terminal Emulator Add-in - Quick Reference

The following tables summarize basic information about the Terminal Emulator Add-in and how it relates to some commonly-used aspects of UFT.

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 "Windows-Based Application Support" on page 81 .
Supported Environments	For details on supported emulators, see the Terminal Emulator Add-in section of the <i>HP Unified Functional Testing Product Availability Matrix</i> , available from the HP Support Matrix page (requires an HP passport).
Important Information	<ul style="list-style-type: none"> Before using the Terminal Emulator Add-in for the first time, you must enable UFT to identify your terminal emulator. See "Terminal Emulator Configuration Wizard Overview" on page 405. You must configure your terminal emulator settings to work with UFT. See "How to Set Your HLLAPI Terminal Emulator to Work with UFT" on page 389.
Test Object Methods and Properties	The Terminal Emulator Add-in provides test objects, methods, and properties, which can be used when testing objects (controls) in Terminal Emulator applications. For details, see the Terminal Emulators section of the <i>HP UFT Object Model Reference for GUI Testing</i> .
Checkpoints and Output Values	<ul style="list-style-type: none"> See the sections describing checkpoints and output values in the <i>HP Unified Functional Testing User Guide</i>. See "Checkpoints and Output Values - Terminal Emulators" on page 404. See "GUI Checkpoints and Output Values Per Add-in" on page 512
Troubleshooting and Limitations	"Troubleshooting and Limitations - Terminal Emulator" on page 419

Prerequisites	
Opening Your Application	You can open your Terminal Emulator application before or after opening UFT and creating a test.
Add-in Dependencies	None

Configuration	
Wizard	"Terminal Emulator Configuration Wizard Overview" on page 405
Options Dialog Box	<ul style="list-style-type: none"> Use the Terminal Emulator pane. (Make sure that a GUI test is open and select Tools > Options > GUI Testing tab >Terminal Emulator node.) See "Terminal Emulator Pane (Options Dialog Box > GUI Testing Tab)" on page 397.

	<ul style="list-style-type: none">• Use the Windows Applications pane. (Tools > Options > GUI Testing tab > Windows Applications node) See "Windows Applications > General Pane (Options Dialog Box > GUI Testing Tab)" on page 85.
Custom Active Screen Capture Settings Dialog Box (tests only)	Use the Terminal Emulator section in the dialog box. (Tools > Options > GUI Testing tab > Active Screen node > Custom Level) See the section on the Custom Active Screen Capture Settings dialog box in the <i>HP Unified Functional Testing User Guide</i> .
Application Area Additional Settings pane (business components only)	Use the Applications pane. In the application area, click Additional Settings > Applications in the sidebar. See the section on the Applications pane in the <i>HP Unified Functional Testing User Guide</i> .

Terminal Emulator Add-in - Testing and Configuration

This chapter includes:

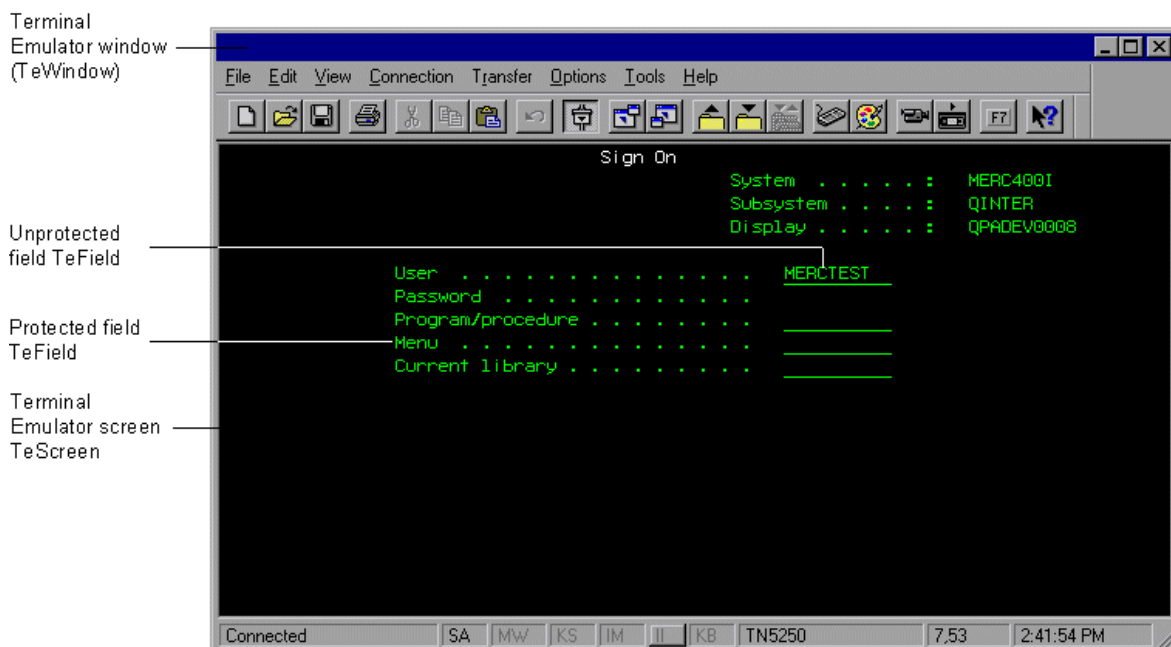
- [Terminal Emulator Add-in - Overview](#)381
- [Recording Tests and Components on Terminal Emulator Applications](#)382
 - [Considerations for Recording and Running Tests and Components on Terminal Emulators](#) 383
- [Run Session Synchronization](#)384
- [Terminal Emulator Recovery Scenarios](#) 384
- [How to Configure an Emulator to Work with the Terminal Emulator Add-in](#) 385
- [How to Set Your HLLAPI Terminal Emulator to Work with UFT](#) 389
- [How to Manage Terminal Emulator Configuration Settings](#) 389
- [How to Copy Existing Terminal Emulator Configurations](#) 390
- [How to Check the Validity of a Terminal Emulator Configuration](#) 392
 - [Validating a Terminal Emulator - Possible Error Responses](#) 392
- [How to Synchronize Steps on Terminal Emulators](#) 395
- [Terminal Emulator Pane \(Options Dialog Box > GUI Testing Tab\)](#) 397
- [Terminal Emulator Configuration Adjustments Dialog Box](#) 399
- [Checkpoints and Output Values - Terminal Emulators](#) 404
- [Terminal Emulator Configuration Wizard Overview](#) 405
 - [Terminal Emulator Configuration Wizard Welcome Page](#) 406
 - [Specify Wizard Operation Page](#) 407
 - [Emulator Setup Page](#) 408
 - [Configure HLLAPI Properties Page](#) 409
 - [HLLAPI Configuration Test Page](#) 410
 - [Configure Emulator Classes Page](#) 413
 - [Configure Emulator Screen Settings Page](#) 414
 - [Completing the Terminal Emulator Configuration Wizard Page](#) 417
- [Troubleshooting and Limitations - Terminal Emulator](#) 419

Terminal Emulator Add-in - Overview

You can use UFT 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.

UFT 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.



If your emulator supports HLLAPI, UFT recognizes the screen and field objects in your emulator screen. If your emulator does not support HLLAPI, or you have configured UFT in **Text-only** mode, UFT records operations in terms of the text as it appears in the rows and columns of your emulator screen.

The UFT 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 Overview](#)" on page 405.

To configure your HLLAPI emulator to work with UFT, see "[How to Set Your HLLAPI Terminal Emulator to Work with UFT](#)" on page 389.

Recording Tests and Components on Terminal Emulator Applications

As you record, the test or business 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. UFT 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 business component. For more details, see the section in the *HP Unified Functional Testing User Guide* on running and closing applications programmatically, and the **Standard Windows** section of the *HP UFT Object Model Reference for GUI Testing*.

By default, when you record a test or business component, UFT 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 details, see ["How to Synchronize Steps on Terminal Emulators" on page 395](#).

The following is a sample of a UFT 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"
```

UFT displays this test in the Keyword View like this:

Item	Operation	Value	Documentation
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 UFT 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 "1"
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
```

UFT displays this test in the Keyword View like this:

Item	Operation	Value	Documentation
TeWindow			
TeTextScreen	ClickPosition	24,2	Click row 24, column 2 of the "TeTextScreen" screen.
TeTextScreen	Type	"1"	Type "1" in the "TeTextScreen" screen.
TeTextScreen	Type	micReturn	Type micReturn in the "TeTextScreen" screen.
TeTextScreen	WaitString	"FRSMAIN",...	Wait 2000 milliseconds for the "FRSMAIN" string to appear.
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.

This section also includes ["Considerations for Recording and Running Tests and Components on Terminal Emulators"](#) below.

Considerations for Recording and Running Tests and Components on Terminal Emulators

- Connect your emulator to the host and ensure that the emulator is configured properly. For details, see ["How to Set Your HLLAPI Terminal Emulator to Work with UFT"](#) on page 389.
- Make sure that only one terminal emulator session is open. (Multiple open sessions may cause problems with recording and running tests or business components.)
- If your test or business component includes calls to WinRunner tests, make sure that these tests do not use the WinRunner Terminal Emulator Add-in. Similarly, when running WinRunner tests with the WinRunner Terminal Emulator Add-in, make sure that these tests do not use the UFT Terminal Emulator Add-in if they include calls to UFT tests.
- 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), you can change the recording mode for your emulator by adjusting the configuration. For details, see ["How to Manage Terminal Emulator Configuration Settings"](#) on page 389.
- UFT 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 details, see ["Checkpoints and Output Values - Terminal Emulators" on page 404](#).

Run Session Synchronization

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 helps to ensure that UFT performs the next step in the test or business 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.

The following table illustrates the synchronization options available for various terminal emulator applications:

Emulator type	Synchronization options
All emulator types	You can instruct UFT to delay the run session: <ul style="list-style-type: none">• For a specified period of time• Until a specific string appears in a defined area• Until a specified property achieves a defined value
Emulators that fully support HLLAPI	You can synchronize the run session with the response time of the host. By default, during a record session, UFT automatically generates a <code>Sync</code> statement for the <code>TeScreen</code> object each time the emulator waits for a response from the host.
Emulators that do not support HLLAPI	When you record using a terminal emulator that does not support HLLAPI, or that has been configured as supporting text-only HLLAPI operations, UFT automatically generates a <code>Sync</code> statement for the <code>TeTextScreen</code> object each time a specified key is pressed. The default is the ENTER key. UFT waits a specified period of time, to allow the host sufficient response time.

For a list of tasks describing how to synchronize, see ["How to Synchronize Steps on Terminal Emulators" on page 395](#).

Terminal Emulator Recovery Scenarios

UFT allows you to define recovery scenarios for your tests or business components, to cater for various unexpected events, such as crashes and error situations, which can disrupt your tests or business 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 business 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 more details on defining recovery scenarios, see the *HP Unified Functional Testing User Guide*.

How to Configure an Emulator to Work with the Terminal Emulator Add-in

This section describes how to configure an emulator to work with the Terminal Emulator Add-in:

"Attachmate EXTRA!" on the next page
"Attachmate myEXTRA! Terminal Viewer" on the next page
"Attachmate INFOConnect" on the next page
"Hummingbird HostExplorer" on page 387
"IBM Personal Communications (PCOM)" on page 387
"IBM WebSphere Host On-Demand" on page 387
"NetManange RUMBA" on page 388
"NetManage RUMBA Web-to-Host" on page 388
"Seagull BlueZone" on page 388
"WRQ Reflection" on page 389
"Zephyr Passport" on page 389

Attachmate EXTRA!

To connect your EXTRA! terminal emulator to UFT:

1. Open 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 before you start testing with UFT. 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 UFT:

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.
8. If this is the first time that you are connecting to a myEXTRA! terminal viewer, install the HLLAPI DLL, as follows:
 - a. Click **Preferences**.
 - b. Click the **Install HLLAPI Client Components** link.

Attachmate INFOConnect

To connect your INFOConnect terminal emulator to UFT:

1. Open Attachmate INFOConnect.
2. Select **Options > Global Preferences** from the main menu.
3. Select the **Advanced** tab.
4. Select **A** as the session short name.
5. To associate the session short name (A), with your session, click **Browse** and locate your session profile in the file system.
6. Click **OK**.

Hummingbird HostExplorer

To connect your HostExplorer terminal emulator to UFT:

1. Open HostExplorer.
2. From the HostExplorer main menu, select **File > Save Session Profile**.
3. In the Save Profile dialog box, set the **HLLAPI Short Name** to the uppercase letter **A**.
4. From the main menu, select **Options > API Settings**.
5. In the API Global Settings dialog box, select the **Update screen after PS update** and **Auto sync** options.
6. Click **OK**.

Alternatively:

1. Open 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 UFT to work with IBM PCOM terminal emulators.

IBM WebSphere Host On-Demand

To connect your WebSphere Host On-Demand terminal emulator to UFT:

1. Open the WebSphere Host On-Demand EHLLAPI Enablement Tool. (If you do not have this tool, contact IBM for details on how to acquire and install it.)
2. To enable UFT to record on the IBM WebSphere Host On-Demand terminal emulator, define the session options as follows:
 - a. Click **Configure** and select **Properties** from the list. Then select **Preferences > Start Options** and set **Auto-Start HLLAPI Enabler** to **Yes**.
 - b. Set the **Start In Separate Window** option to **Yes**.
 - c. Set the **Alternate Terminal** option to **Disable**.

Make sure that the server and client are not installed on a computer on which another terminal emulator is installed.

NetManange RUMBA

To connect your RUMBA terminal emulator to UFT:

1. Open 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 UFT. 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 UFT:

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.

Note:

- For versions 5.x: Only Mainframe Display is supported for the Java client Only Replay is supported for both Java client and Pro client.
- For version 6.x: Java Client is not supported. Only Replay is supported for Pro client.

For more details on supported versions RUMBA Web-to-Host, see the *HP Unified Functional Testing Product Availability Matrix*, available from the [HP Support Matrix page](#) (requires an HP passport).

Seagull BlueZone

To connect your BlueZone terminal emulator to UFT:

1. Open BlueZone.
2. In BlueZone, select **Options > API**. The API Properties dialog box opens.
3. Click the **Options** tab.
4. In the **Short NameSession Identifier** field, type the uppercase letter **A**.
5. Click **OK**.
6. Save the session.

WRQ Reflection

To connect your Reflection terminal emulator to UFT:

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 UFT:

1. Open a new or existing session.
2. Check that the session shortname **(A) Passport.zws** appears in the window title bar.

How to Set Your HLLAPI Terminal Emulator to Work with UFT

The steps below provide a general overview of how to enable testing on your terminal emulator application when working with an emulator that supports HLLAPI. For details on how to configure a specific emulator, see ["How to Configure an Emulator to Work with the Terminal Emulator Add-in" on page 385](#).

1. Connect your emulator to the host before running the Terminal Emulator Configuration Wizard and before recording each test or business component.
2. Assign the uppercase letter **A** as the short name for the current emulator session.
3. (Optional) Restart the emulator after changing these settings.

Note: For details on supported emulator versions and protocols, see the *HP Unified Functional Testing Product Availability Matrix*, available from the [HP Support Matrix page](#) (requires an HP passport)..

How to Manage Terminal Emulator Configuration Settings

The following steps describe how to manage your terminal emulator configuration settings.

- ["Change configuration settings" on the next page](#)
- ["Restore default settings for the selected preconfigured emulator" on the next page](#)
- ["Restore settings for a user-defined configuration" on the next page](#)

Change configuration settings

The Terminal Emulator Configuration Adjustments dialog box contains check boxes, radio buttons, and options that require a numeric or text value.

1. Open the "[Terminal Emulator Configuration Adjustments Dialog Box](#)", as described on page 397.
2. Enter a numeric or text value for an option:
 - a. Click the option once to highlight it.
 - b. Click the option again or press **F2** to access the value to be changed.
 - c. Change the value as necessary.
 - d. Click another location in the dialog box to set the value.
3. Click **OK** to update the current terminal emulator configuration and close the dialog box.

Restore default settings for the selected preconfigured emulator

1. Open the "[Terminal Emulator Configuration Adjustments Dialog Box](#)", as described on page 397.
2. Click the **Reset** button. (This button is enabled only if a preconfigured emulator is selected.)

Restore settings for a user-defined configuration

1. Locate the saved registry file that contains the configuration settings in the <UFT installation folder>\dat folder on your computer. The file has a .reg extension. (The path for the dat folder in a typical installation is: %ProgramFiles%\HP\Unified Functional Testing\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 was copied into the registry.
4. Click **OK**. The settings in the saved file are restored.

Tip: You can also restore the settings for a user-defined terminal emulator, if these settings were saved previously using the wizard. For details, see "[Completing the Terminal Emulator Configuration Wizard Page](#)" on page 417.

How to Copy Existing Terminal Emulator Configurations

This task describes how to copy a terminal emulator configuration from another user who has already configured the UFT settings for a specific emulator using the Terminal Emulator Configuration Wizard.

For example, if the settings for your terminal emulator were 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 configuring the settings yourself.

This task includes the following steps:

- ["Prerequisites" below](#)
- ["Copy the registry file to your computer" below](#)
- ["Register the file" below](#)
- ["Set the new emulator as the default emulator - Optional" below](#)
- ["Modify the emulator settings - Optional" below](#)
- ["Results" on the next page](#)

1. Prerequisites

- The existing configuration file must be saved to a registry file, using the **Save terminal emulator settings to file** option in the wizard's final page. For details, see ["Completing the Terminal Emulator Configuration Wizard Page " on page 417.](#)
- 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.

2. Copy the registry file to your computer

- a. Locate the registry file containing the configuration settings for your emulator. The file has a .reg extension.
- b. Copy the file to the <UFT installation folder>\dat folder on your computer.
The path for the dat folder in a typical installation is: %ProgramFiles%\HP\Unified Functional Testing\dat

3. Register the file

- a. Double-click the registry file to open the Registry Editor message box.
- b. Click **Yes** to add the information into the registry. A message opens confirming that the information has been copied into the registry.
- c. Click **OK**. The emulator name assigned to this configuration is added to the list of available terminal emulators for your UFT installation.

4. Set the new emulator as the default emulator - Optional

- a. Open UFT with the Terminal Emulator Add-in loaded.
- b. Select the new emulator name from the list in the **Tools > Options > GUI Testing tab > Terminal Emulator** pane, and set it as your default emulator.

5. Modify the emulator settings - Optional

- a. Open UFT with the Terminal Emulator Add-in loaded.
- b. Open the Terminal Emulator Configuration Wizard. For details, see ["Terminal Emulator Configuration Wizard Overview" on page 405.](#)

6. Results

After you copy a configuration file from another location, the emulator name assigned to this configuration is added to the list of available terminal emulators for your UFT installation.

Note: If you copy a configuration file after starting UFT, you need to close and reopen UFT to see the updated list of available emulators.

How to Check the Validity of a Terminal Emulator Configuration

1. Make sure that a GUI test is open.
2. Open the **Terminal Emulator** pane of the Options dialog box (**Tools > Options > GUI Testing tab > Terminal Emulator** node) (described on page 397).
3. Click **Validate**.

If a problem is detected, a brief description (error response) is displayed in the pane. For [details](#) on handling the error, click **Troubleshoot** to open a Help page that displays error-specific information.

Validating a Terminal Emulator - Possible Error Responses

The following possible error responses may be displayed in the **Terminal Emulator** pane of the Options dialog box (**Tools > Options > GUI Testing tab > Terminal Emulator** node) when you click the **Validate** button:

- "Invalid HLLAPI DLL" below
- "Cannot detect an open session" on the next page
- "Cannot locate the main window class " on the next page
- "Cannot detect the emulator screen" on the next page
- "Cannot connect to the open session" on the next page
- "Cannot retrieve session text" on page 394
- "Cannot detect open session, or Cannot locate the main window class" on page 394
- "HLLAPI DLL not found" on page 394
- "More than one session open" on page 394
- "Unknown error" on page 394

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 > GUI Testing tab > Terminal Emulator node > Open Wizard**).

For more details, see the table listing the DLL names used by supported terminal emulators in "[Configure HLLAPI Properties Page](#)" on page 409, or the documentation provided by your emulator provider.

Cannot detect an open session

UFT 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

UFT cannot find the terminal emulator main window class name.

- Ensure that the terminal emulator main window class name is configured correctly in the Terminal Emulator Configuration Wizard (**Tools > Options > GUI Testing tab > 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

UFT cannot find the terminal emulator main window class name.

- Ensure that the terminal emulator main window class name is configured correctly in the Terminal Emulator Configuration Wizard (**Tools > Options > GUI Testing tab > 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 UFT and then restart the emulator. If this does not resolve the problem, contact your emulator provider.

Cannot retrieve session text

UFT cannot display text captured in the current session.

- HLLAPI Emulators—Restart UFT 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

UFT 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 the terminal emulator main window class name is configured correctly in the Terminal Emulator Configuration Wizard (**Tools > Options > GUI Testing** tab > **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

UFT 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 > GUI Testing** tab > **Terminal Emulator** node > **Open Wizard**).

For more details, see the table listing the DLL names used by supported terminal emulators in "[Configure HLLAPI Properties Page](#)" on page 409, 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 UFT and then restart the emulator.

How to Synchronize Steps on Terminal Emulators

The following steps describe how to perform various types of synchronization operations. For conceptual details, see ["Run Session Synchronization " on page 384](#).

- ["Insert a synchronization step while recording" below](#)
- ["Set synchronization timeout" below](#)
- ["Insert a synchronization point for an object" below](#)
- ["Wait for a specified text string" on the next page](#)

Insert a synchronization step while recording

1. Select **Design > Emulator Synchronization**.
2. (Optional) 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, UFT uses the default timeout interval, as described in ["Set synchronization timeout" below](#).

Note:

- You can adjust your emulator configuration to prevent UFT from automatically inserting `Sync` steps for TeScreen objects in your test or business component.
- You can specify the keys that generate `Sync` steps for TeTextScreen objects.

For details, see ["How to Manage Terminal Emulator Configuration Settings" on page 389](#)

Set synchronization timeout

In the Run Pane of the Test Settings Dialog Box (**File > Settings > Run** node), set the **Object Synchronization Timeout**. For details, see the *HP Unified Functional Testing User Guide*.

This enables you to specify the maximum interval (in milliseconds) that UFT waits before running each test step.

Note:

- This option is not available for business 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.

Insert a synchronization point for an object

Select **Design > Synchronization Point**. For details, see the *HP Unified Functional Testing User Guide*.

When you insert a synchronization point into your test or business component, UFT generates a `WaitProperty` statement in the Editor. This statement instructs UFT to pause the test or business component until a particular object property achieves the value you specify.

Example:

If you want the run session to wait until the **Text** property of the **Result** field has a value of `Successful`, insert the following statement:

```
TeScreen("LogOn").TeField("Result").WaitProperty "Text", "Successful"
```

Wait for a specified text string

UFT'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 **Design > Emulator WaitString**. 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. UFT inserts a step into your test or business component with the following syntax:

TeScreen object:

```
TeScreen(description).WaitString String [, TopRow, LeftColumn,  
BottomRow, RightColumn, Timeout, RegExp]
```

TeTextScreen object:

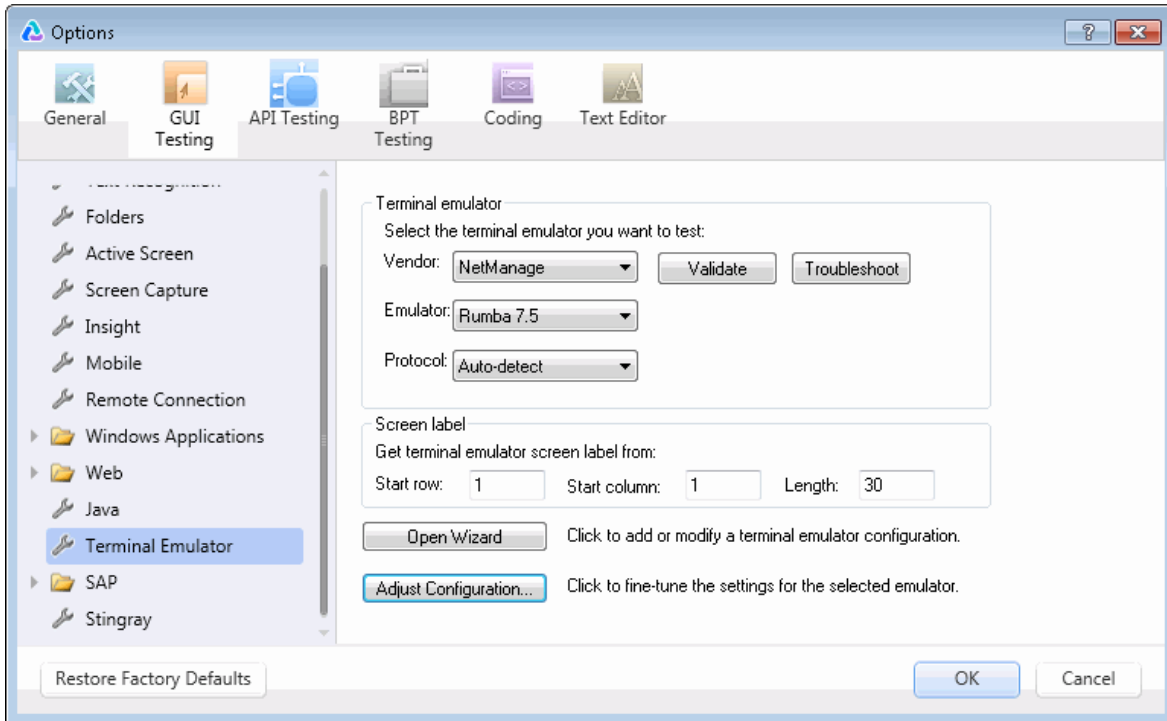
```
TeTextScreen(description).WaitString String, [TopRow, LeftColumn,  
BottomRow, RightColumn, Timeout, RegExp]
```

The position on the screen is defined by the values of the four corners of the rectangle, each corner with its own argument.

3. Optionally, 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 UFT to identify objects and text strings with varying values. For more details on regular expressions, see the *HP Unified Functional Testing User Guide*.
 - Add a 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, UFT uses the default timeout interval. For details, see ["Set synchronization timeout" on the previous page](#).

Terminal Emulator Pane (Options Dialog Box > GUI Testing Tab)

This pane enables you to change your terminal emulator settings and to validate your current terminal emulator configurations.



To access

- Do one of the following:
 - Ensure that a GUI test, action, or component is in focus in the document pane.
 - In the Solution Explorer, select a GUI test or component node or one of its child nodes.
- Select **Tools > Options > GUI Testing tab > Terminal Emulator** node.

Important information

This pane is available only when the UFT 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 business component when you click **OK**.

Terminal Emulator area: 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 **Restore Factory Defaults** button resets all Options dialog box options to their defaults.

Relevant tasks	"How to Check the Validity of a Terminal Emulator Configuration" on page 392
-----------------------	--

User interface elements are described below:

UI Elements	Description
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.
Protocol	The protocol that your emulator uses. Recommended: Auto-detect
Validate	Validates the current configurations of the selected emulator, and, if an error is detected, displays a brief description that error. Errors may result from: <ul style="list-style-type: none"> invalid terminal emulator configurations in the Terminal Emulator pane (Tools > Options > GUI Testing tab > Terminal Emulator node). invalid configurations made when configuring the terminal emulator using the Terminal Emulator Configuration Wizard (Tools > Options > GUI Testing tab > Terminal Emulator node > Open Wizard). errors in the terminal emulator itself. For a list of possible error responses, see "Validating a Terminal Emulator - Possible Error Responses" on page 392 .
Troubleshoot	Opens a specific Help page that provides a troubleshooting solution where available. Available only if an error response is displayed after you click Validate . <div style="background-color: #f0f0f0; padding: 5px; border: 1px solid #ccc;"> <p>Example: If you click Validate and Invalid HLLAPI DLL is displayed, you can click Troubleshoot to open a Help topic displaying information on handling this error.</p> </div> For details, see "Validating a Terminal Emulator - Possible Error Responses" on page 392 .
Screen Label	The area from which UFT reads the label property of the emulator screen while recording a test or business component. If the location and length of the label are correctly defined, UFT uses this value as the name of the TeScreen object. The Screen label area is enabled only for emulators that support HLLAPI. To set the screen label: 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). <div style="background-color: #f0f0f0; padding: 5px; border: 1px solid #ccc;"> <p>Note: You can change the way that UFT reads the label property of the emulator screen, by adjusting the configuration settings. For details, see "How to Manage Terminal Emulator Configuration Settings" on page 389.</p> </div>
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 details, see "Terminal Emulator Configuration Wizard Overview" on page 405 .

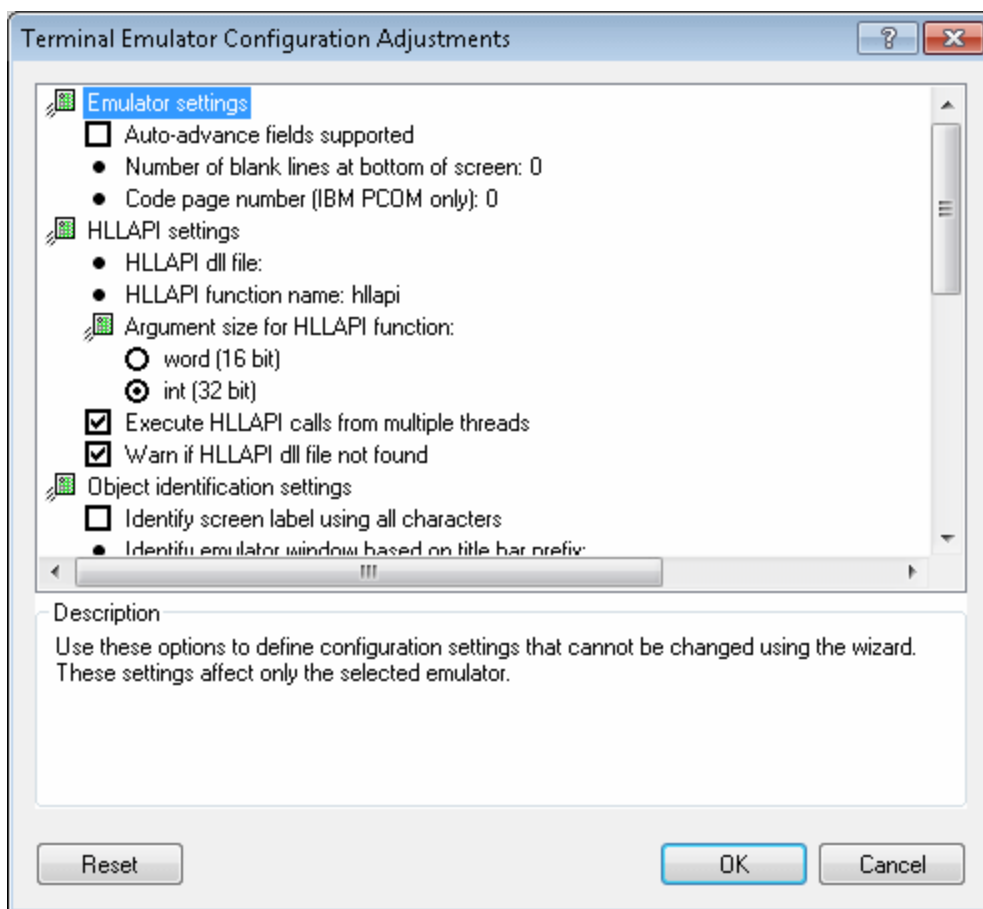
UI Elements	Description
Adjust Configuration	<p>Opens the "Terminal Emulator Configuration Adjustments Dialog Box" "Terminal Emulator Configuration Adjustments Dialog Box" (described on page 399) to allow changes to existing configuration settings in exceptional circumstances.</p> <p>Note: In general, it is recommended to use the Terminal Emulator Configuration Wizard to configure your emulator screen settings. For details, see "Configure Emulator Screen Settings Page" on page 414. 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 business components. For more details, see "How to Manage Terminal Emulator Configuration Settings" on page 389.</p>

Terminal Emulator Configuration Adjustments Dialog Box

This dialog box enables you to make adjustments to existing terminal emulator configuration settings. This is required only in exceptional circumstances. In the majority of cases, UFT works successfully with terminal emulators using the supplied preconfigured settings and with the configuration settings defined using the Terminal Emulator Configuration Wizard.

Caution: 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 business components.

The following is an example of the options displayed in the dialog box for an emulator that supports HLLAPI.



<p>To access</p>	<ol style="list-style-type: none"> 1. Do one of the following: <ul style="list-style-type: none"> • Ensure that a GUI test, action, or component is in focus in the document pane. • In the Solution Explorer, select a GUI test or component node or one of its child nodes. 2. Select Tools > Options > GUI Testing tab > Terminal Emulator node. 3. Select an emulator and click Adjust Configuration,
<p>Important information</p>	<p>The choice of options displayed in this dialog box depends on the type of terminal emulator selected in the Terminal Emulator pane.</p> <p>The majority of the options in this 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.</p> <p>For a few 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, UFT keeps this setting even if you select a different emulator.</p>
<p>Relevant tasks</p>	<p>You modify options that are displayed as bullets by clicking the text and modifying the value when the text becomes an editable box. For details, see "Change configuration settings" on page 390.</p>
<p>See also</p>	<p>"How to Manage Terminal Emulator Configuration Settings" on page 389</p>

User interface elements are described below:

UI Elements	Description
Emulator Settings	<p>The following options enable you to define configuration settings that cannot be changed using the wizard:</p> <ul style="list-style-type: none"> 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. <p>If your emulator supports auto-advance fields, select this check box to enable UFT to record Set statements on these fields.</p> <p>Displayed for: Emulators that support HLLAPI.</p> 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 UFT'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. <p>Enter the number of lines that your emulator reserves at the bottom of the screen. UFT includes this value in its algorithm for identifying field locations.</p> 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. <p>UFT uses this code page to correctly identify the keys you record.</p>
HLLAPI Settings	<p>The following options enable you to define configuration settings for the selected emulator. They are displayed only for emulators that support HLLAPI.</p> <ul style="list-style-type: none"> HLLAPI .dll file. UFT uses the HLLAPI DLL file specified for this emulator to connect to the emulator and to retrieve data concerning its current status. <p>If you are using a customized version of a preconfigured emulator, you may need to specify a different DLL file name.</p> HLLAPI function name. The HLLAPI DLL for this emulator uses this function as the entry point for all HLLAPI calls. <p>If you are using a customized version of a preconfigured emulator, you may need to specify a different function name.</p> 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. <p>Select the correct argument size for the selected emulator: word (16 bits) or integer (32 bits).</p> 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. (Selected by default for a preconfigured emulator configuration.) <p>Clear this check box to instruct UFT to open a separate process for HLLAPI calls and to execute all HLLAPI calls from this single thread.</p> Warn if HLLAPI dll file not found. Instructs UFT to display a warning message when the HLLAPI DLL file for the current configuration cannot be found. For example, UFT warns you if you try to use the Terminal Emulator Add-in before you have installed the emulator itself.

UI Elements	Description
	<p>If you clear this check box and UFT cannot find the required DLL file, it may be difficult to determine why UFT is not recording successfully. It is therefore recommended that you leave this option selected.</p> <p>Applies to: All terminal emulator configurations, regardless of the currently selected emulator.</p>
<p>Object Identification Settings</p>	<p>The following options enable you to configure the way UFT identifies objects for the selected terminal emulator:</p> <ul style="list-style-type: none"> <p>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 details, see "Terminal Emulator Pane (Options Dialog Box > GUI Testing Tab)" on page 397.</p> <p>By default, only the protected characters in the defined label area are captured for the label property value.</p> <p>Select this option if you want UFT 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.</p> <p>Identify emulator window based on title bar prefix. UFT 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 UFT to identify the window based on a static prefix in the window title bar.</p> <p>To instruct UFT to use a prefix to identify the correct window, specify the text string for the prefix.</p> <p>When no value is specified, UFT uses the object class to identify the emulator window.</p>
<p>Record Settings</p>	<p>The following options enable you to configure the way UFT records operations:</p> <ul style="list-style-type: none"> <p>Communicate with server using these keys. When recording without HLLAPI support, UFT 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.</p> <p>Default: 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).</p> <p>Specify the decimal value of the virtual key code for each key, separated by a semicolon (;). UFT inserts a Sync step whenever one of these keys is pressed. For more details on synchronization, see "How to Synchronize Steps on Terminal Emulators" on page 395.</p> <p>For a list of virtual key codes, see http://msdn.microsoft.com/en-us/library/dd375731(VS.85).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.</p> <p>Applies to: All terminal emulator configurations, regardless of the currently selected emulator.</p> <p>Record operations on menus and dialog boxes. By default, UFT records operations on the menus of the terminal emulator window and the dialog boxes that open as a result of these menu option selections.</p> <p>Clear this check box if you do not want UFT 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 business component if cross-emulator compatibility is important, or if these steps are not relevant to your test or business component.</p> <p>Applies to: All terminal emulator configurations, regardless of the currently selected emulator.</p>

UI Elements	Description
Record Settings (continued)	<ul style="list-style-type: none"> Record mode. In Text screen mode, UFT records operations as TeTextScreen steps, based on the screen coordinates. In Context-sensitive mode, UFT records field operations as TeField steps. 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 details, see "Terminal Emulator Configuration Wizard Overview" on page 405. <div style="background-color: #f0f0f0; padding: 5px; margin: 10px 0;"> <p>Note: For emulators that do not support HLLAPI and those configured as supporting text-only HLLAPI operations, UFT always uses Text screen mode and this option is not available.</p> </div> <ul style="list-style-type: none"> Record steps without synchronizing. By default, when UFT recognizes a user operation in the terminal emulator application, such as keyboard input or a mouse click, UFT suspends the processing of the user input in the application. After the recorded statement is added to the test or business component script and the Active Screen information has been saved, UFT 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 UFT 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 UFT or the emulator (or both). If you do select this option, make sure that you allow sufficient time for UFT to record each step before performing another operation.
Record Settings (continued)	<ul style="list-style-type: none"> Record cursor position. When recording in a text screen or field, UFT 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 business component. 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 UFT 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. Displayed 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 details on property patterns, see <i>Using Property Patterns to Identify Objects (Advanced)</i> in PropPattern.htm. This file is located in the help subfolder of the UFT 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. Applies to: All terminal emulator configurations, regardless of the currently selected emulator.
Run Settings	<p>The following options enable you to configure the way UFT runs tests or business components for the selected terminal emulator, if the emulator supports HLLAPI:</p> <ul style="list-style-type: none"> Beep when Sync operations are performed. Indicates whether UFT beeps after performing each Sync

UI Elements	Description
	<p>operation during a run session.</p> <p>Applies to: All terminal emulator configurations, regardless of the currently selected emulator.</p> <ul style="list-style-type: none"> • Run steps containing special emulator keys using keyboard events. Instructs UFT to perform SendKey commands using keyboard events. If you do not use this option to specify key codes, UFT performs SendKey commands using the corresponding HLLAPI function. <p>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 (;).</p> <ul style="list-style-type: none"> • Time between emulator status checks (in milliseconds). During a Sync step, UFT waits the specified period of time before checking the emulator status. UFT 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 details on synchronization, see "How to Synchronize Steps on Terminal Emulators" on page 395. <p>Specify the interval (in milliseconds) between emulator status checks.</p> <p>Default: 200</p> <p>Note: Specifying a very long interval could significantly increase the time your tests or business components take to run.</p>

Checkpoints and Output Values - Terminal Emulators

While recording your test, you can add text checkpoints for the following:

- TeScreen and TeTextScreen objects
- status bar of the terminal emulator window
- dialog boxes that open after menu options are selected

While editing your test or business component, you can:

- add text checkpoints for TeScreen objects.
- add text checkpoints for TeTextScreen objects if the test was recorded using an emulator with full HLLAPI support that was configured to record in **Text screen** mode. For details on changing the emulator mode, see ["How to Manage Terminal Emulator Configuration Settings" on page 389](#).
- output property or text values from the objects in your terminal emulator application to use in your test or business component.

Guidelines for Using Checkpoints and Output Values

- 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. UFT recognizes these as standard Windows objects. For more details on the properties of standard Windows objects, see the **Standard Windows** section of the *HP UFT Object Model Reference for GUI Testing*.

For details on standard, text, and bitmap checkpoints, and on standard and text output values, see the *HP Unified Functional Testing User Guide*.

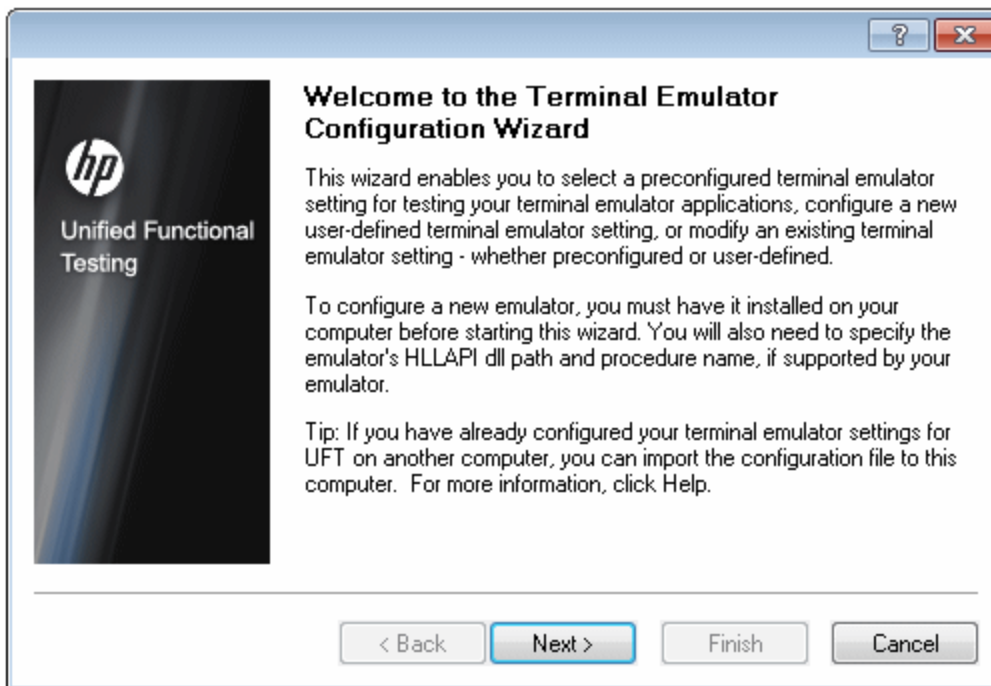
Terminal Emulator Configuration Wizard Overview

This wizard guides you through the process of configuring the settings UFT needs to identify your terminal emulator. If your emulator is not in the list of preconfigured settings to select, you can define the way UFT identifies your emulator.

<p>To access</p>	<ol style="list-style-type: none"> 1. Open UFT with the Terminal Emulator Add-in loaded. 2. In UFT, select Tools > Options > GUI Testing tab > Terminal Emulator node. 3. In the Terminal Emulator pane, click Open Wizard. <p>Note: After you install UFT, the Terminal Emulator Configuration Wizard opens automatically if you selected Run the Terminal Emulator wizard in the Additional Installation Requirements dialog box.</p>
<p>Important information</p>	<ul style="list-style-type: none"> • To configure 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 cannot connect to your terminal emulator. • If terminal emulator settings for UFT are already configured on another computer, you can copy an existing configuration file onto your computer, instead of running the wizard. For details, see "How to Copy Existing Terminal Emulator Configurations" on page 390. • After you run the wizard, the terminal emulator you selected is set as the default emulator when you open UFT with the Terminal Emulator Add-in loaded.
<p>Wizard map</p>	<p>The Terminal Emulator Configuration Wizard contains:</p> <p>"Terminal Emulator Configuration Wizard Welcome Page" (page 406 > "Specify Wizard Operation Page" (page 407) > "Emulator Setup Page" (page 408) > "Configure HLLAPI Properties Page" (page 409 > "HLLAPI Configuration Test Page" (page 410 > "Configure Emulator Classes Page" (page 413 > "Configure Emulator Screen Settings Page" (page 414 > "Completing the Terminal Emulator Configuration Wizard Page " (page 417</p> <p>Note: Pages that are in parentheses open according to the option selected in the previous page. Therefore, not all pages are displayed for every option.</p>
<p>See also</p>	<ul style="list-style-type: none"> • You can check your configurations by clicking the Validate button in the 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 details, see "How to Check the Validity of a Terminal Emulator Configuration" on page 392. • You can use the wizard to select a different emulator for use with your tests or business components. For more details, see "Terminal Emulator Configuration Wizard Overview" above.

Terminal Emulator Configuration Wizard Welcome Page

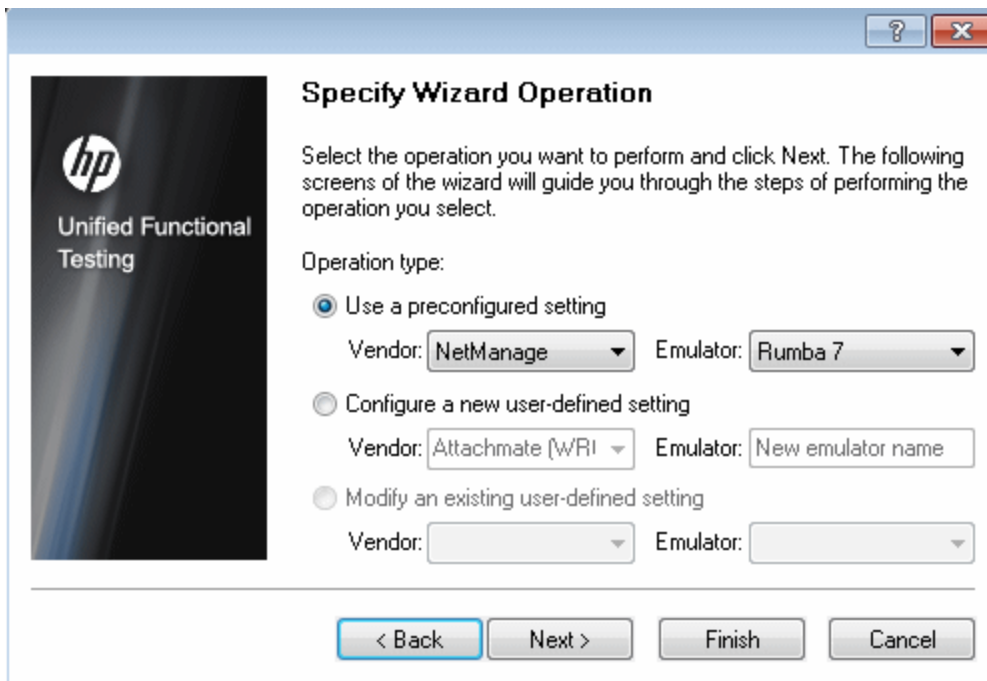
This wizard page provides general information about the different options in the Terminal Emulator Configuration Wizard.



Wizard map	<p>The Terminal Emulator Configuration Wizard contains:</p> <p>Terminal Emulator Configuration Wizard Welcome Page > "Specify Wizard Operation Page " (page 407) > "Emulator Setup Page" (page 408 > "Configure HLLAPI Properties Page" (page 409) > "HLLAPI Configuration Test Page" (page 410) > "Configure Emulator Classes Page" (page 413) > "Configure Emulator Screen Settings Page" (page 414 > "Completing the Terminal Emulator Configuration Wizard Page " (page 417</p> <p>Note: Pages that are in parentheses open according to the option selected in the previous page. Therefore, not all pages are displayed for every option.</p>
See also	<ul style="list-style-type: none">• For details on how to open the wizard, see "Terminal Emulator Configuration Wizard Overview" on the previous page.• For details on copying an existing configuration, see "How to Copy Existing Terminal Emulator Configurations" on page 390.

Specify Wizard Operation Page

This wizard page enables you to specify the operation that you want the wizard to perform.



<p>Wizard map</p>	<p>The Terminal Emulator Configuration Wizard contains:</p> <p>"Terminal Emulator Configuration Wizard Welcome Page" (page 406) > Specify Wizard Operation Page > "Emulator Setup Page" (page 408) > "Configure HLLAPI Properties Page" (page 409) > "HLLAPI Configuration Test Page" (page 410) > "Configure Emulator Classes Page" (page 413) > "Configure Emulator Screen Settings Page" (page 414) > "Completing the Terminal Emulator Configuration Wizard Page " (page 417</p> <p>Note: Pages that are in parentheses open according to the option selected in the previous page. Therefore, not all pages are displayed for every option.</p>
<p>See also</p>	<ul style="list-style-type: none"> For details on how to open the wizard, see "Terminal Emulator Configuration Wizard Overview" on page 405. For details on copying an existing configuration, see "How to Copy Existing Terminal Emulator Configurations" on page 390.

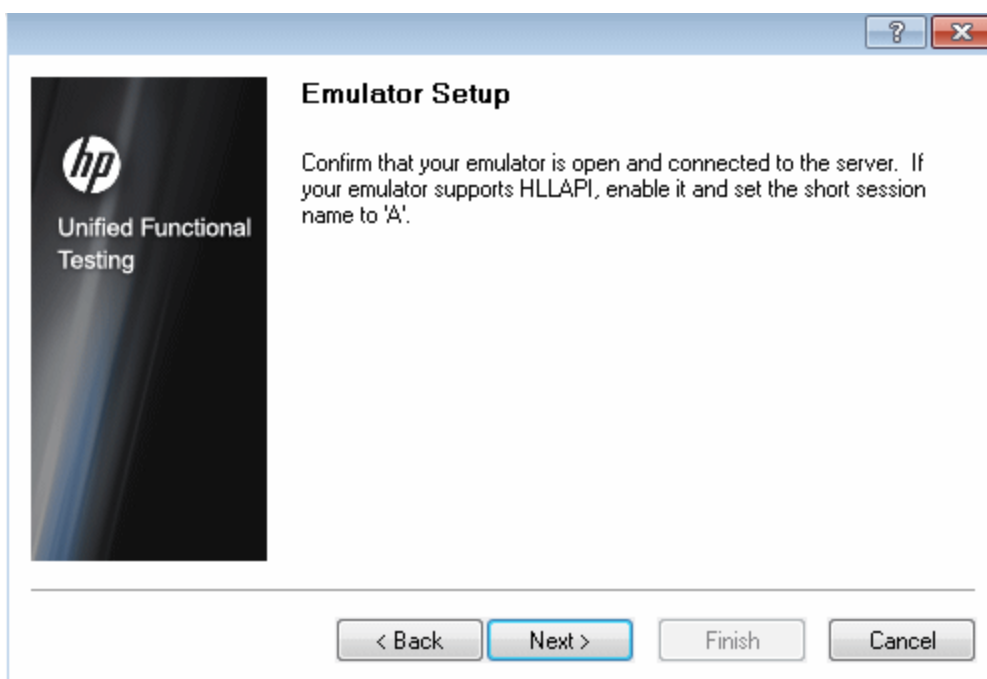
User interface elements are described below:

UI Elements	Description
<p>Use a preconfigured setting</p>	<p>Enables you to select one of the vendor/emulator settings that are supplied with your Terminal Emulator Add-in.</p> <p>Note: If you select this option, you can click Finish instead of Next to begin working with UFT to test the emulator you selected.</p> <p>However, if you are testing a Web-based emulator, or if UFT is not recording or recognizing</p>

UI Elements	Description
	<p>objects as expected, it is recommended to click Next and define the emulator screen settings.</p> <p>(The emulator screen settings do not affect run sessions; they affect only recording and other object operations (for example, inserting checkpoints and using the Object Spy.)</p> <p>For details on using your emulator with UFT, see "Terminal Emulator Add-in - Overview" on page 381.</p>
<p>Configure a new user-defined setting</p>	<p>Enables you to supply the details of your vendor and emulator.</p> <p>After completing the wizard, the vendor and emulator names that you define here are displayed in the list of vendor/emulator combinations in the Terminal Emulator pane.</p>
<p>Modify an existing user-defined setting</p>	<p>Enables you to modify settings that were previously configured using the Terminal Emulator Configuration Wizard.</p>

Emulator Setup Page

This wizard page instructs you to open your terminal emulator and connect it to the server.

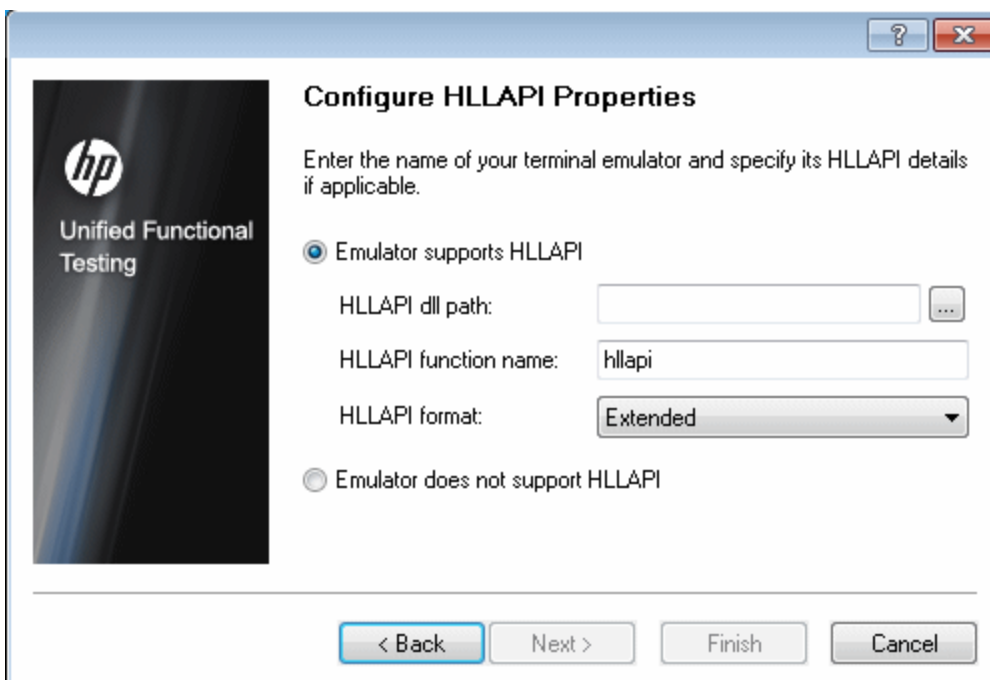


<p>Important information</p>	<p>If your emulator supports HLLAPI, enable it and set the emulator short session name to the uppercase letter A. For details, see "How to Set Your HLLAPI Terminal Emulator to Work with UFT" on page 389.</p>
<p>Wizard map</p>	<p>The Terminal Emulator Configuration Wizard contains:</p> <p>"Terminal Emulator Configuration Wizard Welcome Page" (page 406 > "Specify Wizard Operation Page" (page 407 > Emulator Setup Page) > "Configure HLLAPI Properties Page" (page 409 > "HLLAPI Configuration Test Page" (page 410) > "Configure Emulator Classes Page" (page 413 > "Configure Emulator Screen Settings Page" (page 414) > "Completing the Terminal Emulator Configuration Wizard"</p>

	<p>Page " (page 417</p> <p>Note: Pages that are in parentheses open according to the option selected in the previous page. Therefore, not all pages are displayed for every option.</p>
See also	<ul style="list-style-type: none"> For details on how to open the wizard, see "Terminal Emulator Configuration Wizard Overview" on page 405. For details on copying an existing configuration, see "How to Copy Existing Terminal Emulator Configurations" on page 390.

Configure HLLAPI Properties Page

This wizard page enables you to specify whether your terminal emulator supports HLLAPI.



Wizard map	<p>The Terminal Emulator Configuration Wizard contains:</p> <p>"Terminal Emulator Configuration Wizard Welcome Page"406 > "Specify Wizard Operation Page " (page 407 > "Emulator Setup Page" (page 408 > Configure HLLAPI Properties Page > "HLLAPI Configuration Test Page" (page 410 > "Configure Emulator Classes Page" (page 413 > "Configure Emulator Screen Settings Page" (page 414 > "Completing the Terminal Emulator Configuration Wizard Page " (page 417</p> <p>Note: Pages that are in parentheses open according to the option selected in the previous page. Therefore, not all pages are displayed for every option.</p>
See also	<ul style="list-style-type: none"> For details on how to open the wizard, see "Terminal Emulator Configuration Wizard Overview" on page 405. For details on copying an existing configuration, see "How to Copy Existing Terminal Emulator Configurations" on page 390.

User interface elements are described below:

UI Elements	Description
Emulator supports HLLAPI	If your emulator supports HLLAPI, select this option and supply the information described in the table below**. If you are not sure what values to enter, consult your terminal emulator documentation or contact the vendor for your terminal emulator.
HLLAPI dll path	The HLLAPI dynamic-link library file specified for the selected emulator that UFT uses 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	The format by which UFT 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, UFT 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.
Emulator does not support HLLAPI	Select this option if your emulator does not support HLLAPI.

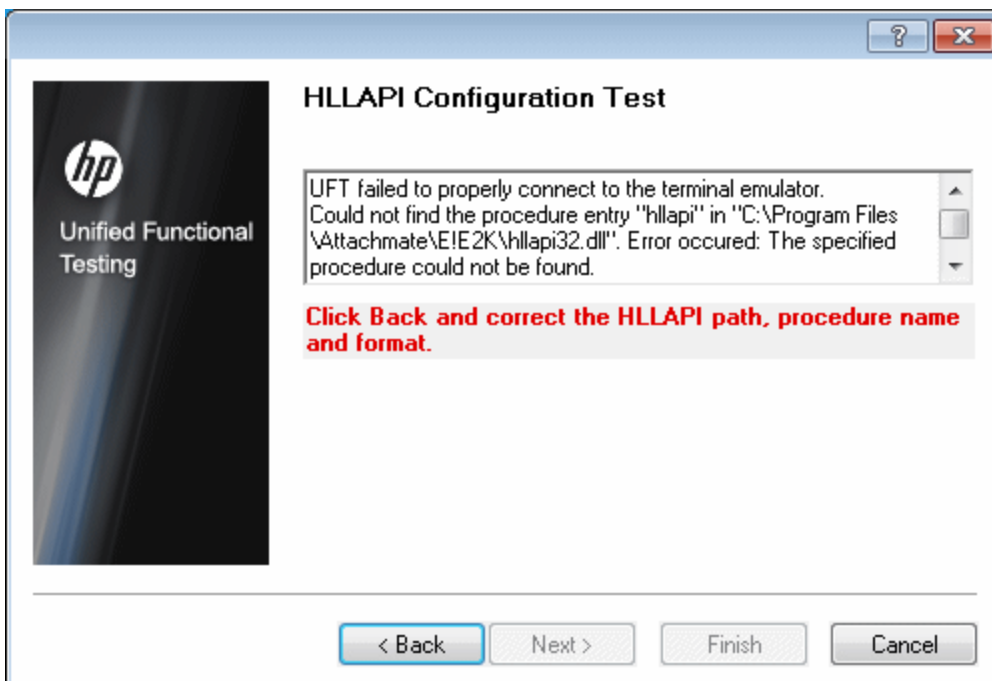
**The table below lists the DLL and function names used by the supported terminal emulators.

Emulator Name	DLL Name	HLLAPI Function Name
Attachmate EXTRA! and Attachmate myEXTRA! Terminal Viewer	ehlap132.dll	hllapi
Attachmate INFOConnect	ihlap132.dll	WinHLLAPI
Hummingbird HostExplorer	ehllap32.dll	HLLAPI32
IBM Personal Communications (PCOM) and IBM WebSphere Host On-Demand	pcshl132.dll	hllapi
NetManage RUMBA and NetManage RUMBA Web-To-Host	ehlap132.dll	hllapi
PuTTY	Not applicable	Not applicable
Seagull BlueZone	WHLAPI32.dll	hllapi
WRQ Reflection	hllap132.dll	hllapi
Zephyr (PC/Web to Host)	PassH11.dll	hllapi
HP Teem Talk	Not applicable	Not applicable

HLLAPI Configuration Test Page

This wizard page displays a screen capture test if you selected **Emulator supports HLLAPI** in the Configure HLLAPI Properties page. This test enables you to determine whether UFT accurately identifies

your terminal emulator screen.



<p>Important information</p>	<p>Check that the screen capture test is correct for your currently selected terminal emulator, and that all the text is correctly identified and displayed.</p> <ul style="list-style-type: none"> • If the wizard displays the emulator screen and the text correctly, click Next to continue. • If the wizard does NOT display the emulator screen and the text correctly, see "Troubleshooting the HLLAPI Properties Configuration" on the next page.
<p>Wizard map</p>	<p>The Terminal Emulator Configuration Wizard contains:</p> <p>"Terminal Emulator Configuration Wizard Welcome Page" (page 406 > "Specify Wizard Operation Page" (page 407 > "Emulator Setup Page" (page 408 > "Configure HLLAPI Properties Page" (page 409 > HLLAPI Configuration Test Page) > "Configure Emulator Classes Page" (page 413 > "Configure Emulator Screen Settings Page" (page 414) > "Completing the Terminal Emulator Configuration Wizard Page " (page 417</p> <p>Note: Pages that are in parentheses open according to the option selected in the previous page. Therefore, not all pages are displayed for every option.</p>
<p>See also</p>	<ul style="list-style-type: none"> • For details on how to open the wizard, see "Terminal Emulator Configuration Wizard Overview" on page 405. • For details on copying an existing configuration, see "How to Copy Existing Terminal Emulator Configurations" on page 390.

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 details, see ["How to Set Your HLLAPI Terminal Emulator to Work with UFT" on page 389](#).
 - Check that the settings you entered in the ["Configure HLLAPI Properties Page"](#) are accurate (DLL path, procedure, format). For details, see ["Configure HLLAPI Properties Page" on page 409](#).
 - Make sure that the HLLAPI .dll file you specified in the DLL path is not in use by UFT or another application.

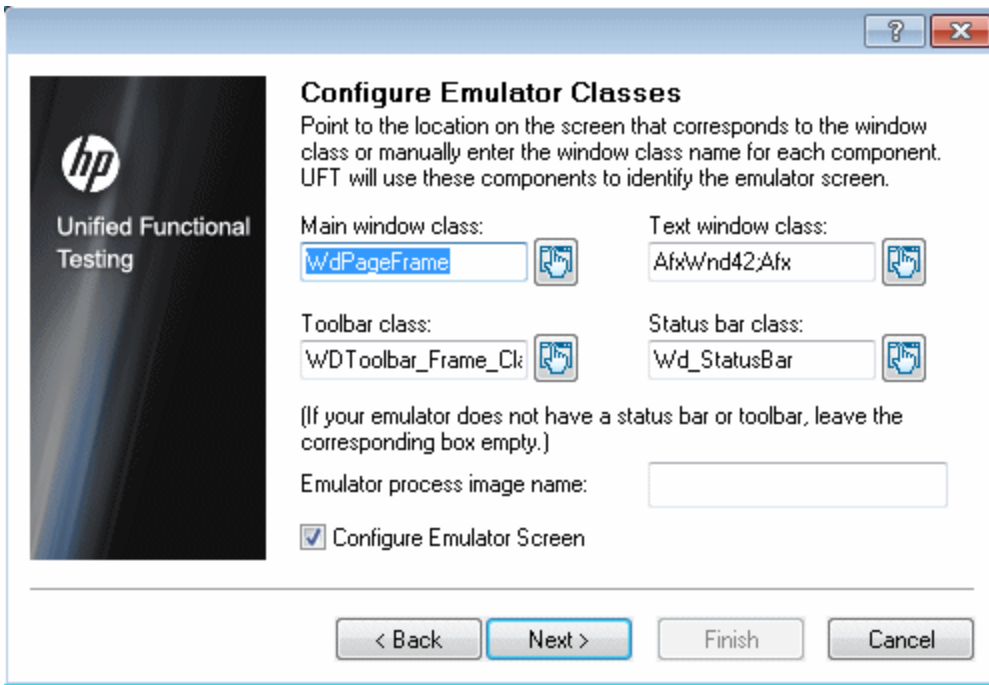
If the .dll file is currently in use by another application, click **Cancel** to close the wizard, close the application using the DLL, and restart the wizard.

If the .dll file is currently in use by UFT, select 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 page, change the **HLLAPI format** to **Text-only**. Use the **Text-only** option if you are working with a VT protocol, or if you have begun working in UFT and encountered problems with recording and running tests or business components. For details, see ["Configure HLLAPI Properties Page" on page 409](#).
3. If the above tips do not solve the problem, click **Back** and, in the Configure HLLAPI Properties page, select **Emulator does not support HLLAPI**. For details, see ["Configure HLLAPI Properties Page" on page 409](#).

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 details, see ["How to Manage Terminal Emulator Configuration Settings" on page 389](#).

Configure Emulator Classes Page

This wizard page enables you to specify emulator class information. UFT uses this information to locate classes on the emulator screen. By identifying the business components of the terminal emulator window, UFT distinguishes between the window of the terminal emulator and the screens in the host application.



Important information	You identify the business components of the emulator by clicking the pointing hand and then clicking the corresponding object on your terminal emulator window.
Wizard map	<p>The Terminal Emulator Configuration Wizard contains:</p> <p>"Terminal Emulator Configuration Wizard Welcome Page" (406) > "Specify Wizard Operation Page" (page 407) > "Emulator Setup Page" (page 408) > "Configure HLLAPI Properties Page" (page 409) > "HLLAPI Configuration Test Page" (410) > Configure Emulator Classes Page > "Configure Emulator Screen Settings Page" (page 414) > "Completing the Terminal Emulator Configuration Wizard Page" (page 417)</p> <p>Note: Pages that are in parentheses open according to the option selected in the previous page. Therefore, not all pages are displayed for every option.</p>
See also	<ul style="list-style-type: none"> For details on how to open the wizard, see "Terminal Emulator Configuration Wizard Overview" on page 405. For details on copying an existing configuration, see "How to Copy Existing Terminal Emulator Configurations" on page 390.

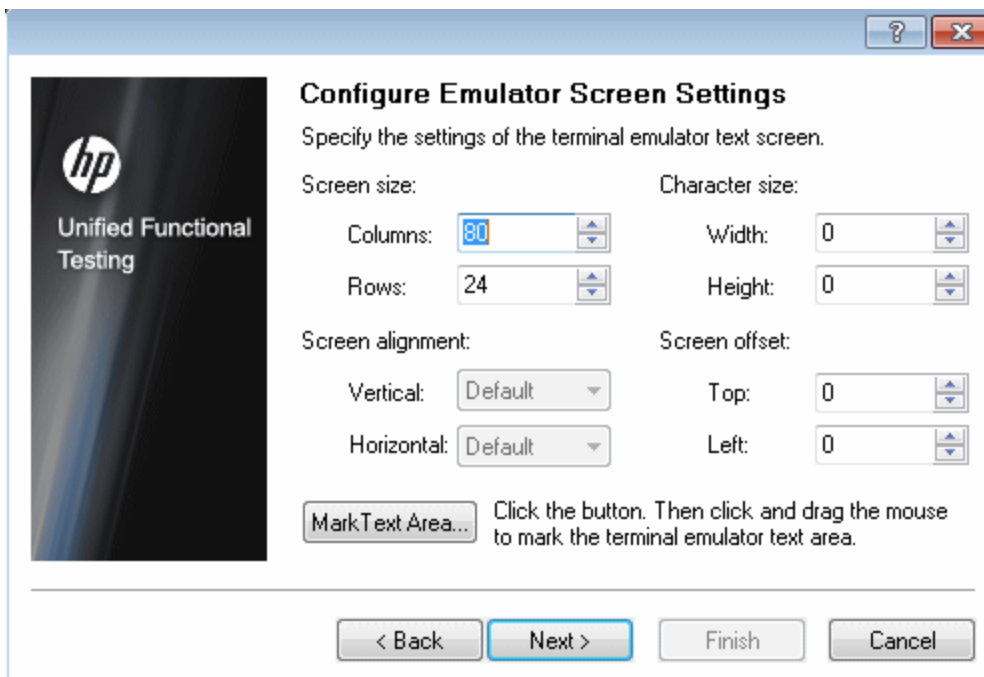
User interface elements are described below (unlabeled elements are shown in angle brackets):

UI Elements	Description
<emulator class options>	<p>You identify the business components of the emulator by clicking the pointing hand and then clicking the corresponding object on your terminal emulator window.</p> <ul style="list-style-type: none"> • Main window class. The uppermost title bar of the main emulator window. • Text window class. The text within the emulator screen. • Toolbar class. The terminal emulator's toolbar (if applicable). • Status bar class. The lowest status bar of the main emulator window (if applicable).
Emulator Process Image Name	<p>The process name for the emulator. (The wizard detects the process name for the emulator and displays it in this box after identifying the emulator main window class.)</p> <p>UFT uses this process name to identify the correct process for this terminal emulator when recording and running tests or business components.</p> <p>Make sure that the displayed process name is correct for this emulator.</p> <div style="background-color: #f0f0f0; padding: 5px; margin-top: 10px;"> <p>Tip: You can view the image names of the currently loaded processes in the Image Name column of the Windows Task Manager Processes tab.</p> </div>
Configure Emulator Screen	<p>Select this check box to configure your emulator screen correctly for use with UFT if:</p> <ul style="list-style-type: none"> • Your emulator does not support HLLAPI. • Your emulator supports HLLAPI, but you want to review or change the configuration settings for the emulator screen. <div style="background-color: #f0f0f0; padding: 5px; margin-top: 10px;"> <p>Note: If you do not select this check box, the Terminal Emulator Add-in automatically:</p> <ol style="list-style-type: none"> 1. Retrieves the configuration settings for the emulator screen. 2. Adjusts the screen size and alignment using a proprietary algorithm with the settings retrieved for the emulator. These settings are normally correct, so you may not need to select this check box. </div>

Configure Emulator Screen Settings Page

This wizard page enables you to specify the settings of the terminal emulator text screen.

This page opens only if you selected a preconfigured setting, or if you selected the **Configure Emulator Screen** check box in the Configure Emulator Classes page. In this case, your emulator screen is displayed with a red grid overlay.



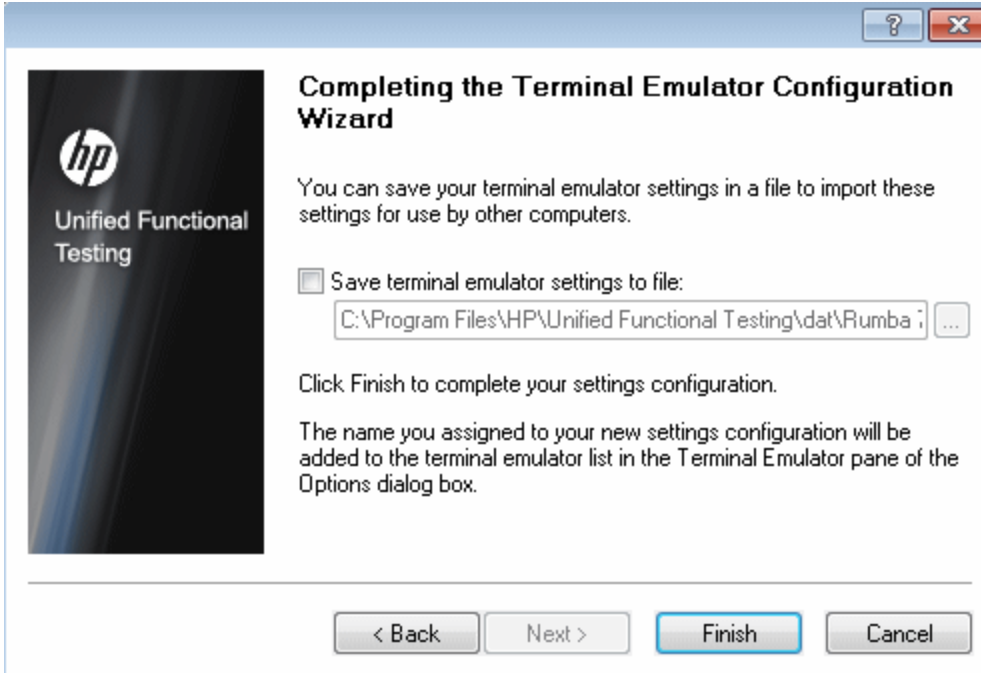
<p>Important information</p>	<p>You change the settings of the emulator screen to correspond to the required settings for your emulator. The character size, column, and row details for your terminal emulator are generally available from your emulator's connection configuration menu.</p> <p>As you change the settings for the emulator screen, the grid automatically adjusts to show the new settings.</p> <p>It is recommended that you specify the text screen settings in the following order:</p> <ol style="list-style-type: none"> 1. Use the Mark Text Area button to mark the text area on your emulator screen. 2. Fine-tune the text screen settings using the options above the Mark Text Area button.
<p>Wizard map</p>	<p>The Terminal Emulator Configuration Wizard contains:</p> <p>"Terminal Emulator Configuration Wizard Welcome Page" (406 > "Specify Wizard Operation Page " (page 407 > "Emulator Setup Page" (page 408 > "Configure HLLAPI Properties Page" (page 409 > "HLLAPI Configuration Test Page" (page 410) > "Configure Emulator Classes Page" (page 413 > Configure Emulator Screen Settings Page) > "Completing the Terminal Emulator Configuration Wizard Page " (page 417</p> <p>Note: Pages that are in parentheses open according to the option selected in the previous page. Therefore, not all pages are displayed for every option.</p>
<p>See also</p>	<ul style="list-style-type: none"> • For details on how to open the wizard, see "Terminal Emulator Configuration Wizard Overview" on page 405. • For details on copying an existing configuration, see "How to Copy Existing Terminal Emulator Configurations" on page 390.

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

UI Elements	Description
Mark Text Area	<p>Enables you to define the dimensions of the terminal emulator text area on your emulator screen. When you click this button, the wizard is minimized and the cursor becomes a crosshairs pointer. Drag the pointer on your emulator screen to define the text area.</p> <p>After you mark the text area on your emulator screen you can fine-tune your settings by adjusting the text screen settings.</p>
<text screen settings>	<p>Screen and Character Sizes</p> <p>You can specify your emulator's screen size in terms of:</p> <ul style="list-style-type: none"> • 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. <p>Alignment and Offset</p> <p>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:</p> <ul style="list-style-type: none"> • 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. <div style="background-color: #f0f0f0; padding: 10px; margin: 10px 0;"> <p>Tip: The screen alignment settings determine how UFT identifies the information on your emulator screen. If you are having trouble recording and running tests or business components (for example, the ClickPosition method is not accurately determining the coordinates), try changing the Screen alignment settings.</p> </div> <ul style="list-style-type: none"> • 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.

Completing the Terminal Emulator Configuration Wizard Page

This wizard page enables you complete the configuration and, optionally, to save your terminal emulator settings to a specific registry file.



<p>Important information</p>	<ul style="list-style-type: none"> • 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 details, see "Terminal Emulator Pane (Options Dialog Box > GUI Testing Tab)" on page 397. • If you save your settings to a registry file, other users will be able to copy and use your terminal emulator configuration. For more details, see "How to Copy Existing Terminal Emulator Configurations" on page 390. • If you are using the wizard from the Terminal Emulator pane of the Options dialog box (Tools > Options > GUI Testing tab > Terminal Emulator node), any changes you make are not applied to the currently open test or business component. To apply your changes, close your test or business component and reopen it.
<p>Wizard map</p>	<p>The Terminal Emulator Configuration Wizard contains:</p> <p>"Terminal Emulator Configuration Wizard Welcome Page" (page 406 > "Specify Wizard Operation Page" (page 407 > "Emulator Setup Page" (page 408 > "Configure HLLAPI Properties Page" (page 409) > "HLLAPI Configuration Test Page" (page 410 > "Configure Emulator Classes Page" (page 413 > "Configure Emulator Screen Settings Page" (page 414) > Completing the Terminal Emulator Configuration Wizard Page</p> <p>Note: Pages that are in parentheses open according to the option selected in the previous page. Therefore, not all pages are displayed for every option.</p>
<p>See also</p>	<ul style="list-style-type: none"> • For details on how to open the wizard, see "Terminal Emulator Configuration Wizard Overview" on

	<p>page 405.</p> <ul style="list-style-type: none"> For details on copying an existing configuration, see "How to Copy Existing Terminal Emulator Configurations" on page 390.
--	---

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

UI Elements	Description
Save terminal emulator settings to file	Saves your settings to a registry file when you specify a location. This enables others to import these settings as needed.
<file location>	The location in which to save the terminal emulator settings registry file.
Finish	Adds the name you assigned to your new configuration settings to the list of available terminal emulators in the Terminal Emulator pane of the Options dialog box as described in " Terminal Emulator Pane (Options Dialog Box > GUI Testing Tab) " on page 397 .

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" below](#)
- ["Connecting and Disconnecting from the Terminal Emulator Add-in" below](#)
- ["Configuration and Settings" on the next page](#)
- ["Creating and Running Tests and Components" on the next page](#)
- ["Working with Terminal Emulator Controls" on page 422](#)
- ["Test Objects, Methods, and Properties" on page 422](#)
- ["Checkpoints and Output Values " on page 423](#)
- ["Multilingual Support" on page 423](#)

Installing and Loading the Terminal Emulator Add-in

- When installing a Hummingbird HostExplorer terminal emulator or patches, make sure that UFT is closed.
- If the UFT Terminal Emulator Add-in is installed and loaded, but there is no terminal emulator installed on your computer, the following error message is displayed: UFT 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 UFT, 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 details, see ["How to Manage Terminal Emulator Configuration Settings" on page 389](#).

- You may experience unexpected behavior after you install an EXTRA! emulator. You may not be able to run UFT or various features may stop working. This happens because the EXTRA! installation may have copied and registered an outdated version of the at1.dll file on your computer.

Workaround: Locate the at1.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, UFT does not recognize either session.

Workaround: While recording or running your test or business component, make sure that only one terminal emulator session is connected at a time.

- If your test or business component contains steps that disconnect the current emulator session during the run session, followed immediately by a `TeScreen.Sync` command, the test or business component run might stop responding or take a long time to respond.
Workaround: Remove the `Sync` command from the test or business component, or replace it with a `Wait` statement. For more details, see the **Utility Objects** section of the *HP UFT Object Model Reference for GUI Testing*.
- Inserting a checkpoint, creating a new test or business component, or opening an existing test or business 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 UFT 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 > GUI Testing** tab > **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 Unified Functional Testing User Guide*.)
- The UFT 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, UFT 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 UFT 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 details about low-level recording, see the *HP Unified Functional Testing User Guide*.

- If you record a test or business 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, UFT 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 UFT is running a test or business component may cause unexpected results.

Workaround: Wait until the end of the test or business component, or pause the test or business component execution before using the emulator.

- To record and run tests or business 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.

- You might encounter unexpected results when you run the Reflection HLL API in multiple threads mode.

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.
- UFT 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 > GUI Testing** tab > **Terminal Emulator** pane > **Adjust Configuration > Run Settings > Run steps containing special emulator keys using keyboard events > Keys for RESET function** option.

- By default, UFT uses the **attached text** and **protected** properties in TeField test object descriptions. If the attached text for a field changes from session to session, UFT cannot find the field during the run session.

Workaround: Open the Object Repository Window or the Object Repository Window 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 business 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 details on Smart Identification, see the *HP Unified Functional Testing 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, UFT may ignore special European language characters while recording or running a test or business component.

Workaround: Set the code page for your IBM PCOM emulator in UFT, using the **Tools > Options > GUI Testing** tab > **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.

Part 17: VisualAge Smalltalk Add-in

This section includes:

["VisualAge Smalltalk Add-in - Quick Reference" on page 425](#)

["How to Configure the VisualAge Smalltalk Add-in" on page 427](#)

VisualAge Smalltalk Add-in - Quick Reference

You can use the UFT VisualAge Smalltalk Add-in to test VisualAge Smalltalk user-interface objects (controls).

The following tables summarize basic information about the VisualAge Smalltalk Add-in and how it relates to some commonly-used aspects of UFT.

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 "Windows-Based Application Support" on page 81 .
Supported Environments	For details on supported VisualAge Smalltalk environments, see the VisualAge Smalltalk Add-in section of the <i>HP Unified Functional Testing Product Availability Matrix</i> , available from the HP Support Matrix page (requires an HP passport).
Important Information	You must configure your VisualAge Smalltalk environment by importing the <code>qt - adapter . dat</code> file and then recompiling your application. See "How to Configure the VisualAge Smalltalk Add-in" on page 427 .
Test Object Methods and Properties	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 details, see the Visual Age Small Talk section of the <i>HP UFT Object Model Reference for GUI Testing</i> .
Checkpoints and Output Values	<ul style="list-style-type: none"> See the sections describing checkpoints and output values in the <i>HP Unified Functional Testing User Guide</i>. See "GUI Checkpoints and Output Values Per Add-in" on page 512
Prerequisites	
Opening Your Application	You can open your VisualAge Smalltalk application before or after opening UFT.
Add-in Dependencies	None
Configuration	
Configuring Your Application	You configure your VisualAge Smalltalk environment by importing the <code>qt - adapter . dat</code> file and then recompiling your application. See "How to Configure the VisualAge Smalltalk Add-in" on page 427 .
Record and Run Settings Dialog Box (tests only)	Use the Windows Applications tab. (Record > Record and Run Settings) See "Windows Applications Tab (Record and Run Settings Dialog Box)" on page 93 . Note: <ul style="list-style-type: none"> UFT can recognize only VisualAge Smalltalk applications that have been precompiled with the <code>qt - adapter agent</code>. For details, see "How to Configure the VisualAge Smalltalk Add-in" on page 427.

	<ul style="list-style-type: none"> • The Record and Run only on radio button applies only to record and run sessions. UFT recognizes all VisualAge Smalltalk 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>Tools > Options > GUI Testing tab > Active Screen node > Custom Level</p> <p>See the section on the Custom Active Screen Capture Settings dialog box in the <i>HP Unified Functional Testing User Guide</i>.</p>
<p>Application Area Additional Settings pane (business components only)</p>	<p>Use the Applications pane.</p> <p>In the application area, select Additional Settings > Applications in the sidebar.</p> <p>See the section the Applications pane in the <i>HP Unified Functional Testing User Guide</i>.</p>

How to Configure the VisualAge Smalltalk Add-in

This task describes how to configure the VisualAge Smalltalk Add-in by importing the `qt-adapter.dat` file to your VisualAge Smalltalk development environment and then recompiling your application to include the **qt-adapter** agent.

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 **AllNames** pane and select **Import > Selected Versions**.
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 `<UFT installation folder>/dat` folder and select `qt-adapter.dat`.
6. In the Selection Required dialog box, do the following:
 - In the **Names** pane, select **Unified Functional Testing**.
 - In the **Versions** pane, select **UFT Adapter 1.0**.
 - Click the **>>** button and click **OK**.
7. In the Configuration Maps Browser window, do the following:
 - In the **AllNames** pane, click **Unified Functional Testing**.
 - In the **Editions and Versions** pane, click **UFT Adapter 1.0**. 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 18: Visual Basic Add-in

This section includes:

["Visual Basic Add-in - Quick Reference" on page 429](#)

["Troubleshooting and Limitations - Visual Basic Add-in" on page 431](#)

Visual Basic Add-in - Quick Reference

You can use the UFT Visual Basic Add-in to test Visual Basic user-interface objects (controls).

The following tables summarize basic information about the **Visual Basic Add-in** and how it relates to some commonly-used aspects of UFT.

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 "Windows-Based Application Support" on page 81 .
Supported Environments	For details on supported Visual Basic environments, see the Visual Basic Add-in section of the <i>HP Unified Functional Testing Product Availability Matrix</i> , available from the HP Support Matrix page (requires an HP passport).
Test Object Methods and Properties	The Visual Basic Add-in provides test objects, methods, and properties that can be used when testing objects in Visual Basic applications. For details, see the Visual Basic section of the <i>HP UFT Object Model Reference for GUI Testing</i> .
Checkpoints and Output Values	<ul style="list-style-type: none"> See the sections describing checkpoints and output values in the <i>HP Unified Functional Testing User Guide</i>. See "GUI Checkpoints and Output Values Per Add-in" on page 512
Troubleshooting and Limitations	See "Troubleshooting and Limitations - Visual Basic Add-in" on page 431 .

Prerequisites	
Opening Your Application	You can open your Visual Basic application before or after opening UFT.
Add-in Dependencies	None

Configuration	
Options Dialog Box	Use the Windows Applications pane. (Tools > Options > GUI Testing tab > Windows Applications node) See "Windows Applications > General Pane (Options Dialog Box > GUI Testing Tab)" on page 85 .
Record and Run Settings Dialog Box (tests only)	Use the Windows Applications tab. (Record > Record and Run Settings) See "Windows Applications Tab (Record and Run Settings Dialog Box)" on page 93 . Note: <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

	<p>operations.</p> <ul style="list-style-type: none"> • UFT 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 > GUI Testing tab > Active Screen node > Custom Level)</p> <p>See the section on the Custom Active Screen Capture Settings dialog box in the <i>HP Unified Functional Testing User Guide</i>.</p>
<p>Application Area Additional Settings pane (business components only)</p>	<p>Use the Applications pane.</p> <p>In the application area, select Additional Settings > Applications in the sidebar.</p> <p>See the section on the Applications pane in the <i>HP Unified Functional Testing User Guide</i>.</p>

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.
- Visual Basic .NET applications are supported by the UFT .NET Add-in.

Part 19: Web Add-in

This section includes:

["Web Add-in - Quick Reference" on page 433](#)

["Web Add-in - Testing and Configuration" on page 436](#)

["Web Add-in - Multiple Browser Support" on page 451](#)

Web Add-in - Quick Reference

You can use the Web Add-in to test HTML user-interface objects (controls).

The following tables summarize basic information about the Web Add-in and how it relates to some commonly-used aspects of UFT. 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 "Web-Based Application Support" on page 34.
Supported Environments	<ul style="list-style-type: none"> For details on supported Web browsers and versions, see the <i>HP Unified Functional Testing Product Availability Matrix</i>, available from the HP Support Matrix page (requires an HP passport). See also Child Add-ins, below.
Child Add-ins	<ul style="list-style-type: none"> UFT 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 440). 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 498.
Test Object Methods and Properties	<p>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 <i>HP UFT Object Model Reference for GUI Testing</i>.</p>
Checkpoints and Output Values	<ul style="list-style-type: none"> See the sections describing checkpoints and output values in the <i>HP Unified Functional Testing User Guide</i>. See "Checkpoints for Web Pages" on page 38. See "GUI Checkpoints and Output Values Per Add-in" on page 512
Extending the Web Add-in	<p>"Web Add-in Extensibility" (described on page 440) enables you to develop support for testing third-party and custom Web controls that are not supported out-of-the-box by the UFT 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 details, see "Siebel Add-in - Overview" on page 324. You can create steps on more than one browser tab, if your browser supports tabbed browsing.

Prerequisites	
Opening Your Application	<p>You must open UFT before opening your Web application.</p>
Testing in Mozilla Firefox	<p>The Functional Testing Extension for Firefox is supported from Firefox versions 33 and higher. If you need to test versions of Firefox earlier than 33, you need to enable UFT support for these Firefox versions.</p> <p>To test your Web application in Mozilla Firefox version 33 and later, you must enable the Functional Testing Extension in one of the following ways:</p>

	<ul style="list-style-type: none"> • If the Select Your Add-ons screen is displayed when you open Firefox, select to enable the Functional Testing Extension. • If the Install Add-on tab opens and displays Functional Testing Extension when opening Firefox, select the Allow this installation check box and click Continue. <p>Otherwise:</p> <ol style="list-style-type: none"> a. In Firefox, open the browser menu. b. In the menu, click Add-ons. c. In the Add-ons Manager tab, select the Extensions node. d. In the Functional Testing Extension row, click the Enable button. <p>To test your Web application in Mozilla Firefox versions earlier than 33, you must enable the legacy Firefox agent:</p> <ol style="list-style-type: none"> 1. Open the <UFT installation folder>\Installations\Firefox folder. 2. From the Firefox folder, drag the Legacy.xpi file into Firefox. 3. In Firefox, open the browser menu. 4. In the menu, click Add-ons. 5. In the Add-ons Manager tab, select the Extensions node. 6. In the Functional Testing Extension row, disable the Functional Testing 12.50 extension and enable the extension you added to Firefox. <p>Note: Only one version of the Unified Functional Testing Extension can be enabled in Firefox at a time.</p> <p>For more details, see "Working With Mozilla Firefox" on page 467.</p>
<p>Testing in Google Chrome</p>	<ul style="list-style-type: none"> • UFT communicates with the Functional Testing Agent Chrome Extension to test Web applications running in Google Chrome. <p>The extension is available on the Chrome web store and downloads automatically when possible, for Chrome versions 31 or later. If the extension does not download, go to https://chrome.google.com/webstore/detail/kgpdpdnaophdehalonapacdghamnb and download it manually. If you do not have an Internet connection or are working with Chrome version 30 or earlier, see "How to Enable the HP Functional Testing Agent Chrome Extension" on page 479.</p> <p>IMPORTANT: If you have a previous version of the Functional Testing Agent for Google Chrome installed, you must manually remove this extension before enabling the new version.</p> <ul style="list-style-type: none"> • If you need to test local HTML pages in Google Chrome, you must make additional configuration changes. For details, see "How to Enable UFT to Test Local HTML Pages in Google Chrome" on page 480.
<p>Testing in Apple Safari on a Remote Mac</p>	<p>See "Working With Apple Safari on a Remote Mac Computer" on page 469.</p>
<p>Add-in Dependencies</p>	<p>None</p>

Configuration	
Options Dialog Box	Use the Web pane. (Make sure that a GUI test is open and select Tools > Options > GUI Testing tab > Web > General node.) For details, see " Web > General Pane (Options Dialog Box) " on page 58.
Record and Run Settings Dialog Box (tests only)	Use the Web tab. (Record > Record and Run Settings) See " Web Tab (Record and Run Settings Dialog Box) " on page 71.
Test Settings Dialog Box (tests only)	Use the Web pane. (File > Settings > Web pane) See " Web Pane (Test/Business Component Settings Dialog Box / Application Area - Additional Settings Pane) " on page 68.
Custom Active Screen Capture Settings Dialog Box (tests only)	Use the Web section. (Tools > Options > GUI Testing tab > Active Screen node > Custom Level) See the section on the Custom Active Screen Capture Settings dialog box in the <i>HP Unified Functional Testing User Guide</i> .
Application Area Additional Settings pane (business components only)	Use the Web pane. In the application area, select Additional Settings > Web in the sidebar. <ul style="list-style-type: none"> • See "Web Pane (Test/Business Component Settings Dialog Box / Application Area - Additional Settings Pane)" on page 68. • See the section on defining Application Settings for your application area in the <i>HP Unified Functional Testing User Guide</i>.

Web Add-in - Testing and Configuration

This chapter includes:

- [Event Recording Configuration for Web Objects - Overview](#) 437
 - [Event Listening and Recording for Web Objects](#) 438
 - [Considerations - Event Listening and Recording](#) 439
 - [Event Listening and Recording - Use-case Scenario](#) 440
- [Web Add-in Extensibility](#) 440
 - [Extensibility Accelerator for HP Functional Testing](#) 441
- [How to Manage Custom Web Event Recording Configurations](#) 442
- [How to Manage Listening and Recording Events for Web Objects](#) 443
- [Web Event Recording Configuration Dialog Box](#) 445
- [Custom Web Event Recording Configuration Dialog Box](#) 447
- [Troubleshooting and Limitations - Web Add-in](#) 449

Event Recording Configuration for Web Objects - Overview

When you record on a Web application, UFT 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 need to record more or fewer events than UFT automatically records by default. If so, you can modify the default event recording settings for Web objects using the "[Web Event Recording Configuration Dialog Box](#)" (described on page 445) to use one of three predefined configurations, or you can customize the individual event recording configuration settings to meet your specific needs.

For example, UFT 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.

Considerations for Configuring Web Event Recording for Web Objects

- 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 UFT 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](#)" (described on page 445) affect recording only for objects that UFT 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, UFT 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.

- 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](#)" (described on page 447).

This section also includes:

- ["Event Listening and Recording for Web Objects" below](#)
- ["Considerations - Event Listening and Recording" on the next page](#)

Event Listening and Recording for Web Objects

For each event, you can instruct UFT to:

- listen every time the event occurs on the object.
- listen only if an event handler is attached to the event.
- listen only if a DHTML behavior is attached to the event.
- listen if either an event handler or DHTML behavior are attached to the event.
- 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: UFT 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.

For each event, you can enable recording, disable recording, or enable recording only if the next event is dependent on the selected event.

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.

For task details, see ["How to Manage Listening and Recording Events for Web Objects" on page 443](#).

Recording Right Mouse Button Clicks

UFT enables you to record click events 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.

UFT records the `Click` statement when the `OnClick` event is triggered. UFT 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](#)" (described on page 445).

Note:

- UFT does not record the simultaneous clicking of more than one mouse button.
- UFT does not record the right-click that opens the browser context menu, or the selection of an item from the context menu. For details on modifying the script manually to enable these options, visit the HP Software Self-solve knowledge base and search for document ID KM185231.

For details, see "[How to Configure UFT to Record Mouse Clicks](#)" on page 52.

Considerations - 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, UFT gives first priority to settings for specific `HTML Tag Objects` and second priority to `Web Objects` settings. UFT 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 UFT 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 an example of this, see "[Event Listening and Recording - Use-case Scenario](#)" on the next page.

- Instructing UFT 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 UFT, 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 > GUI Testing** tab > **Web > Advanced** node). For details, see "[Web > Advanced Pane \(Options Dialog Box\)](#)" on page 63.
- 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.

Event Listening and Recording - Use-case Scenario

When you are creating your test, you may want UFT to record a specific event on an object. As a result, you must instruct UFT to listen for the event and to record the event when it occurs.

In this use-case scenario, you are trying to record an `onmouseover` event for a table cell containing two images. When the mouse moves over either of the images contained in the table cell, the event bubbles up to the cell, and the bubbling action includes information on the image that the mouse moved over. You want to record the steps performed on the images.

In order to enable UFT to record the image mouseover event, you can do a number of things:

- In the "[Custom Web Event Recording Configuration Dialog Box](#)" (described on page 447), you configure the mouseover event for the `<TD>` tag (table cell) to **If Handler**. You also disable the **Record** for the cell to **Never**, thereby disabling the recording option. This enables UFT to "hear" the mouseover event on the table cell when it happens in the application but not record the event as part of the test flow. (This is important because the actual "work" of the application is done with the images contained in the table cells.)
- Also in the "[Custom Web Event Recording Configuration Dialog Box](#)" (described on page 447), you disable listening on the `` tag (the image) by setting the **Listen** option to **Never**. In addition, you set the Record option on the image(s) to Enable. This enables UFT to record the actual action on the images.

By setting the recording to be done on the images, but listening to be done on the table cells, you have taught UFT that while something happens (the images appear) when you mouseover the table cell, the important part and the objects to include in the test flow are the images that appear when performing a mouseover on the table cell.

You can also record the actions on the images by setting the **Listen** option on the `` tag to **Always** (which enables UFT to listen for the mouseover event even though the image does not contain a behavior or event handler). You then set **Record** option on the image to **Enable**.

For task details on setting event listening and recording options, see "[How to Manage Custom Web Event Recording Configurations](#)" on page 442. For details on the available options for listening and recording events, see the "[Custom Web Event Recording Configuration Dialog Box](#)".

Web Add-in Extensibility

UFT 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 UFT Web Add-in.

If the test object class that UFT 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 UFT 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:

- UFT 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 441) is an 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 UFT setup program.

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 UFT Extensibility Documentation program group (**Start > All Programs > HP Software > Unified Functional Testing > Extensibility > Documentation** or the <UFT installation folder>\help\Extensibility folder).

Note: For details on accessing UFT and UFT tools and files in Windows 8 and Windows Server 2012, see "[Accessing UFT in Windows 8.X or Higher Operating Systems](#)" on page 517.

A printer-friendly (PDF) version of the *HP UFT Web Add-in Extensibility Developer Guide* is available in the <UFT 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, GWT, and JQueryUI to add dynamic and interactive content to their sites. The controls in these toolkits are complex and require sophisticated and flexible testing capabilities.

UFT Web Add-in Extensibility enables you to extend the Web Add-in to customize how UFT 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 an 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 UFT 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 UFT or to package it so that you can share it with other UFT users.

The Extensibility Accelerator for HP Functional Testing installation is available from the **Add-in Extensibility and Web 2.0 Toolkits** option in the Unified Functional Testing setup program.

Note: As part of the installation process, an html page opens in your browser. To complete the installation successfully, this page must be opened in Internet Explorer.

How to Manage Custom Web Event Recording Configurations

This task describes the different ways you can define, modify, export, and reset custom Web event recording configurations.

This task includes the following steps:

- ["Prerequisite - Open the Custom Web Event Recording Configuration dialog box" below](#)
- ["Add objects to the HTML Tag Objects list" below](#)
- ["Delete objects from the HTML Tag Objects list" below](#)
- ["Save a custom configuration in an XML file" on the next page](#)
- ["Load a custom configuration from an XML file" on the next page](#)
- ["Modify a custom configuration file manually - Optional" on the next page](#)
- ["Reset configuration settings to a pre-configured basic level" on the next page](#)

Prerequisite - Open the Custom Web Event Recording Configuration dialog box

Select **Record > Web Event Recording Configuration**, and then click **Custom Settings**. For user interface details, see ["Custom Web Event Recording Configuration Dialog Box" on page 447](#).

Add objects to the HTML Tag Objects list

1. 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.

Delete objects from the HTML Tag Objects list

1. 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.

Save a custom configuration in an XML file

1. Select **File > Save Configuration As**. The Save As dialog box opens.
2. 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`.
3. Click **Save** to save the file and close the dialog box.

Load a custom configuration from an XML file

1. Select **File > Load Configuration**. The Open dialog box opens.
2. Locate the event configuration file (`.xml`) that you want to load and click **Open**. The dialog box closes and the selected configuration is loaded.

Modify a custom configuration file manually - Optional

Open the `.xml` file that you saved in any text editor, and modify the file according to your needs. To enable UFT to recognize the modifications that you made, the `.xml` file must keep its original structure. For details on the XML file structure, see "[Web Event Recording Configuration XML File Structure](#)" on page 40.

For example, you can modify the file to enable right mouse click recording, as described in "[How to Configure UFT to Record Mouse Clicks](#)" on page 52.

Reset configuration settings to a pre-configured basic level

- **From the Custom Web Event Recording Configuration dialog box.** In the **Reset to** box, select the predefined event recording level you want, and click **Reset**. All event settings are restored to the defaults for the level you selected.
- **From the Web Event Recording Configuration dialog box.** Reset basic level configuration settings by selecting **Default Settings**. The configuration slider is displayed again, and all event settings are restored to the **Basic** event recording configuration level.

Note: When you choose to reset predefined settings, your custom settings are cleared completely. If you do not want to lose your changes, make sure to save your settings in an event configuration file. For details, see "[Custom Web Event Recording Configuration Dialog Box](#)" on page 447.

How to Manage Listening and Recording Events for Web Objects

This task describes the different ways you can manage listening and recording events for Web objects.

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 details, see ["Considerations - Event Listening and Recording" on page 439](#).

This task contains the following steps:

- ["Prerequisite - Open the Custom Web Event Recording Configuration dialog box" below](#)
- ["Add listening events for an object" below](#)
- ["Delete listening events for an object" below](#)
- ["Specify the listening criterion for an event" below](#)
- ["Set the recording status for an event" below](#)
- ["Configure UFT to record mouse click events" on the next page](#)

Prerequisite - Open the Custom Web Event Recording Configuration dialog box

Select **Record > Web Event Recording Configuration**, and then click **Custom Settings**. For user interface details, see ["Custom Web Event Recording Configuration Dialog Box" on page 447](#).

Add listening events for an object

1. 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. For details, see ["Custom Web Event Recording Configuration Dialog Box" on page 447](#).
3. Select the event you want to add. The event is displayed in the **Event Name** column in alphabetical order. By default, UFT listens to the event when a handler is attached and always records the event (as long as it is listened to at some level).

Delete listening events for an object

1. 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.

Specify the listening criterion for an event

1. 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. You can select **Always, If Handler, If Behavior, If Handler or Behavior**, or **Never**. For details, see ["Custom Web Event Recording Configuration Dialog Box" on page 447](#).

Set the recording status for an event

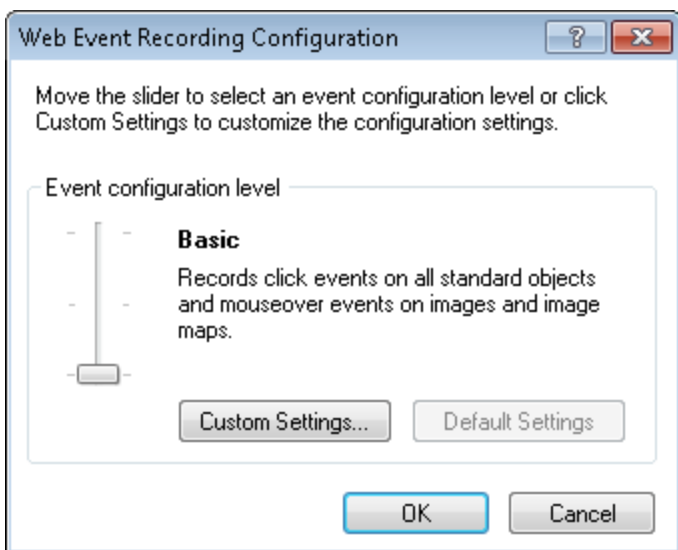
1. Select the object for which you want to modify the recording status or select **Any Web Object**.
2. In the row of the event you want to modify, select a recording status from the **Record** column. For details, see ["Custom Web Event Recording Configuration Dialog Box" on page 447](#).

Configure UFT to record mouse click events

For details, see ["How to Configure UFT to Record Mouse Clicks" on page 52.](#)

Web Event Recording Configuration Dialog Box

This dialog box enables you to select a predefined event configuration level to use when recording on Web objects. By default, UFT uses the **Basic** level. If UFT does not record all the events you need, you may require a higher level.



To access	Select Record > Web Event Recording Configuration .
Important Information	Only the Default settings are supported for recording on Google Chrome browsers.
See also	"Event Recording Configuration for Web Objects - Overview" on page 437

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

UI Elements	Description
<Event configuration level slider>	Enables you to select a predefined event recording configuration. Note: If you define a custom event recording configuration, the slider scale is hidden and the configuration description displays Custom .
Basic (selected by default)	Instructs UFT to: <ul style="list-style-type: none"> Always record click events on Web objects that commonly support clicking, such as images, buttons, and radio buttons. Always record the submit event within forms. Record click events on other Web objects with an Internet Explorer handler or behavior connected. For details on handlers and behaviors, see "Event Listening and Recording for Web Objects" on

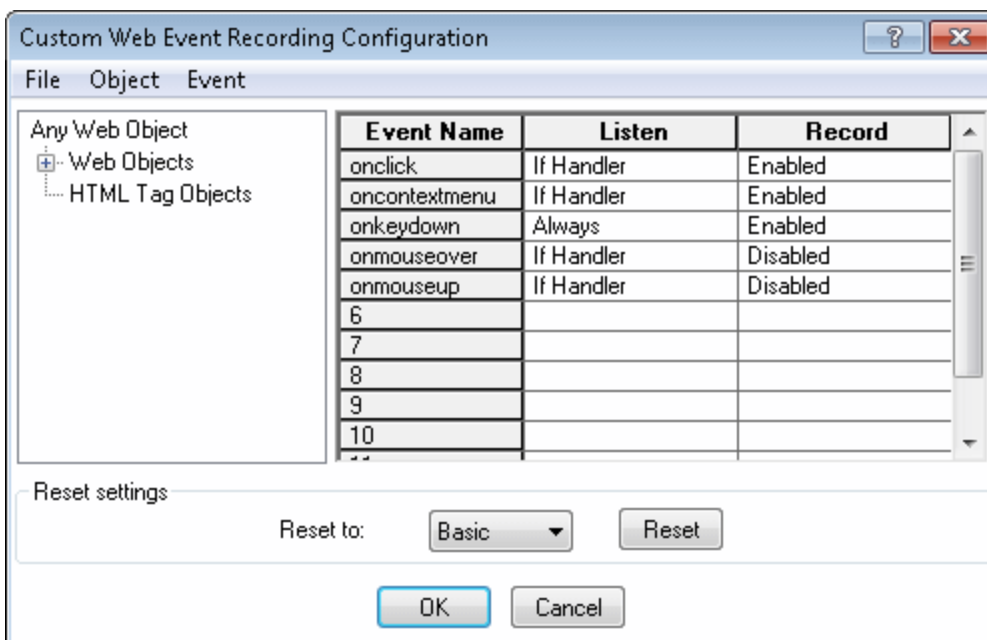
UI Elements	Description
	<p>page 438.</p> <ul style="list-style-type: none">Record the mouseover event on images and image maps only if the event following the mouseover is performed on the same object.
Medium	Instructs UFT to record click events on the <DIV>, , and <TD> HTML tag objects, in addition to the events recorded in the basic level.
High	Instructs UFT to record mouseover, mousedown, and double-click events on Web objects with handlers or behaviors attached, in addition to the events recorded in the basic level.
Custom Settings	Opens the " Custom Web Event Recording Configuration Dialog Box " (described on page 447), which enables you to customize the event recording configuration.
Default Settings	Restores the settings to the Basic level. Note: When you choose to reset predefined settings, your custom settings are cleared completely. If you customized the event recording configuration and do not want to lose these settings, make sure to save your settings in an event configuration file. For details, see " How to Manage Custom Web Event Recording Configurations " on page 442 .

Custom Web Event Recording Configuration Dialog Box

This dialog box enables you to customize the event recording configuration in cases where the predefined Web event configuration levels do not exactly match your recording needs.

You can customize event recording in the following ways:

- Add or delete objects to which UFT should apply special listening or recording settings.
- Add or delete events for which UFT should listen.
- Modify the listening or recording settings for an event.



To access	<ol style="list-style-type: none"> 1. Select Record > 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.
Important information	<ul style="list-style-type: none"> • 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.
Relevant tasks	<ul style="list-style-type: none"> • "How to Manage Custom Web Event Recording Configurations" on page 442 • "How to Manage Listening and Recording Events for Web Objects" on page 443 • "How to Modify Event Recording Configuration for Web-Based Applications" on page 51

	<ul style="list-style-type: none"> • "How to Configure UFT to Record Mouse Clicks" on page 52
See also	<ul style="list-style-type: none"> • "Event Recording Configuration for Web Objects - Overview" on page 437 • "Event Listening and Recording for Web Objects" on page 438

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

UI Elements	Description
<menu options>	<p>File Menu</p> <ul style="list-style-type: none"> • Load Configuration. Enables you to load a custom configuration XML file. • Save Configuration As. Enables you to save your custom configuration in an XML file. <p>Object Menu</p> <ul style="list-style-type: none"> • Add. Enables you to add an HTML tag object to the Objects pane. • Delete. Enables you to delete an HTML tag object from the Objects pane. <p>Event Menu</p> <ul style="list-style-type: none"> • Add. Enables you to add the following events to the Events pane: <code>onchange</code>, <code>ondblclick</code>, <code>onblur</code>, <code>onfocus</code>, <code>onmousedown</code>, <code>onmouseup</code>, <code>onmouseover</code>, <code>onmouseout</code>, <code>onsubmit</code>, <code>onreset</code>, <code>onpropertychange</code>. <p>Depending on the selected object, not every event type is available.</p> <ul style="list-style-type: none"> • Delete. Delete the selected events from the Events pane.
<Objects pane>	<p>The list of Web test object classes and HTML tag objects. The top of the hierarchy is Any Web Object.</p> <div style="background-color: #f0f0f0; padding: 5px;"> <p>Note:</p> <ul style="list-style-type: none"> • 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. • Only HTML tag objects can be added or deleted to this list. </div>
<Events pane>	The list of events associated with the object.
Event Name	The name of the event to which UFT listens and/or records, depending on the settings you specify.
Listen	<p>The criteria that instructs UFT when to listen to the event. The following criteria are available:</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.

UI Elements	Description
Record	<p>The record status for the selected object. The following statuses may be displayed:</p> <ul style="list-style-type: none">• Enabled. Records the event each time it occurs on an object as long as UFT 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.
Reset	<p>Enables you to reset your settings to a pre-configured level. The following levels are available:</p> <ul style="list-style-type: none">• Basic• Medium• High <p>Note: When you choose to reset predefined settings, your custom settings are cleared completely. If you do not want to lose your changes, make sure to save your settings in an event configuration file. For details, see "How to Manage Custom Web Event Recording Configurations" on page 442.</p>

Troubleshooting and Limitations - Web Add-in

This section contains troubleshooting and limitation information about working with the Web Add-in.

Object Identification

- If UFT does not recognize your objects in the correct location, check to see that you are viewing the page at 100%, and are not zooming in or out of the page.

For example, if you view the page at 90% or 120%, you may be required to click or select an area to the left or the right of the actual object in order to recognize it.

- If you open UFT with the Siebel Add-in loaded, the Web Add-in does not appear in the Object Identification dialog box.

Workaround: Change the value of the **HKEY_CURRENT_USER\Software\Mercury Interactive\QuickTest Professional\MicTest\Packages\SiebelPackage\Settings\AllowWebToConfig** registry key value to **1**.

Recording

Autocomplete is not supported during recording sessions. When recording on an autocomplete control, enter the full string instead of selecting the suggestion provided by the control. The step is recorded in the same way as other (non-autocomplete) controls.

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.

Troubleshooting and Limitations - Browser Specific

- ["Troubleshooting and Limitations - Internet Explorer" on page 487](#)
- ["Troubleshooting and Limitations - Mozilla Firefox" on page 488](#)
- ["Troubleshooting and Limitations - Google Chrome and Apple Safari" on page 491](#)

Web Add-in - Multiple Browser Support

This chapter includes:

- [Testing Applications on Multiple Browsers](#) 452
 - [Working with Multiple Browsers - Object Identification Issues](#) 452
 - [Testing Applications on Multiple Browsers - Creating a Single Test for All Browser Testing](#) 457
 - [Testing Applications on Multiple Browsers - Running the Test on Multiple Browsers](#) 458
 - [Using Descriptive Programming for Multiple Browser Testing - Use-case Scenario](#) 459
- [Working With Mozilla Firefox](#) 467
- [Working With Apple Safari on a Remote Mac Computer](#) 469
 - [The UFT Connection Agent for Mac Computers](#) 471
 - [Securing the Communication With the Remote Mac Computer](#) 473
- [How to Set Up Multiple Browser Testing](#) 474
- [How to Enable the HP Functional Testing Agent Chrome Extension](#) 479
- [How to Enable UFT to Test Local HTML Pages in Google Chrome](#) 480
- [How to Connect to a Remote Mac Computer](#) 480
- [How to Install and Configure UFT Connection Agent on Your Mac](#) 483
- [Troubleshooting and Limitations - Internet Explorer](#) 487
- [Troubleshooting and Limitations - Mozilla Firefox](#) 488
- [Troubleshooting and Limitations - Google Chrome and Apple Safari](#) 491

Testing Applications on Multiple Browsers

Web applications and Web controls may be implemented or displayed differently on different browsers. This may affect the behavior of your tests and components, especially if you design them on one browser, and then run them on another. The run results may also differ when running the same test or component on different browsers. For example, if properties are implemented or stored differently on different browsers, UFT may use different properties for object identification or checkpoints depending on the browser you use to open the application.

If you are aware of differences in your application's behavior on different browsers, you may be able to design your tests and components to be browser-independent by anticipating these differences.

There are a number of areas where potential issues can exist:

- [Object Identification](#)
- [Creating a Single Test for Cross-Browser Testing](#)
- [Running Cross Browser Tests](#)

For details on steps to assist you with cross browser testing, see "[How to Set Up Multiple Browser Testing](#)" on page 474.

For a use-case scenario, see "[Using Descriptive Programming for Multiple Browser Testing - Use-case Scenario](#)" on page 459.

Working with Multiple Browsers - Object Identification Issues

When testing web applications and web pages in multiple browsers, one of the foremost challenges is identification of the application/page controls or objects. Due to differences in browser architecture, each of the browsers recognizes and displays the controls and/or objects differently. This can be a visual difference or a property difference that is invisible to the eye. However, because UFT uses these properties (both the visual and hidden ones) as described in the HTML tags, object identification can present issues between browser types.

This can be due to something as basic as different browser layout settings. For example, each browser type structures their toolbar, bookmark, and tab layout differently. However, because of this, the amount of available space for the browser content differs, and likewise the display of the content differs. If you are using location-based identification properties to identify different objects in an application, the layout of the browser can change these properties.

Object property differences also provide a point of potential issue. Within each browser, the properties for a given object can be different (even when your application or Web page uses a CSS to standardize the appearance of application or page elements). For example:

- Link controls are displayed differently on Firefox and Chrome than in Internet Explorer. A link from the Mercury Tours website is identified differently between each of the browsers (with the font, color, and background color properties highlighted):

The screenshot shows the Object Spy application window. At the top, it says "From Internet Explorer". Below that is the "Object hierarchy:" section, which is a tree view showing the following structure:

- WebTable : SIGN-ON
 - WebTable : Atlanta to Las Vegas
 - WebTable : Atlanta to Las Vegas
 - WebTable : Dec 16, 2014
 - WebTable : featured vacation destinations
 - Link : featured vacation destinations

The "Properties" tab is selected, and the "Identification" radio button is chosen. A table lists the properties and their values for the selected link object:

Properties	Values
Class Name	Link
abs_x	527
abs_y	496
background color	transparent
class	
color	#000080
font	Arial, Helvetica, sans-serif, Verdana
height	46
href	http://newtours.demoaut.com/mercuryu
html id	
html tag	A
innerHTML	featured vacation destinations
innerText	featured vacation destinations
name	featured vacation destinations
outerhtml	<a href="mercuryunderconst.php?osCsid
outertext	featured vacation destinations
target	
text	featured vacation destinations
title	
url	http://newtours.demoaut.com/mercuryu
visible	True
width	71
x	527
y	448

Below the table, there are fields for "Selection:" (containing "Class Name") and "Description:" (containing "Descriptions are available only for test object operations."). A "Close" button is located at the bottom right of the window.

The screenshot shows the Object Spy application window. At the top, it says "From Firefox". Below that is the "Object hierarchy:" section, which is a tree view showing the following structure:

- WebTable : SIGN-ON
 - WebTable : Atlanta to Las Vegas
 - WebTable : Atlanta to Las Vegas
 - WebTable : Dec 16, 2014
 - WebTable : featured vacation destinations
 - Link : featured vacation destinations

The "Properties" tab is selected. Under "Identification", there are two radio buttons: "Native" (unselected) and "Identification" (selected). Below this is a table with two columns: "Properties" and "Values".

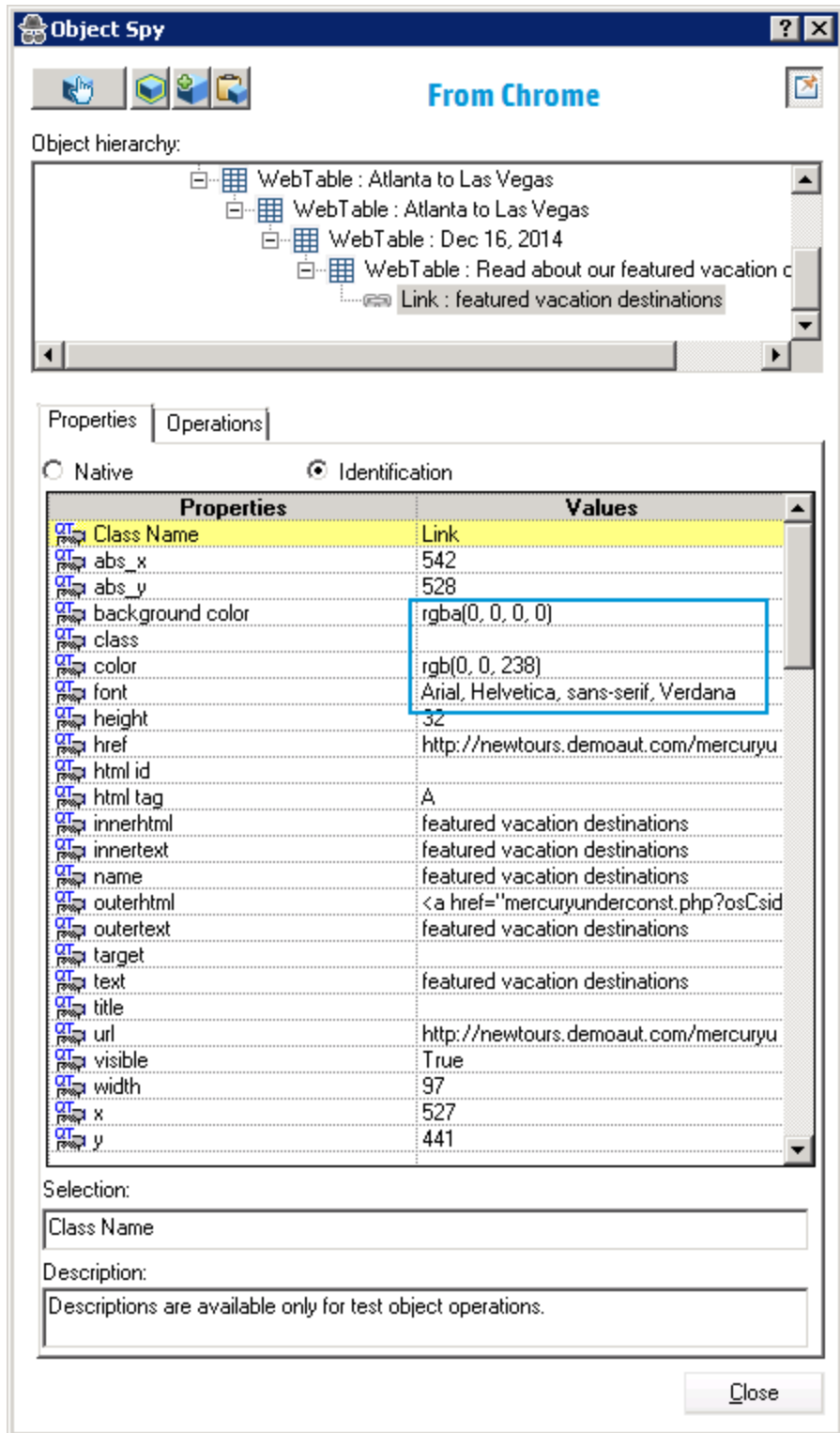
Properties	Values
Class Name	Link
abs_x	527
abs_y	512
background color	transparent
class	
color	#0000ee
font	Arial,Helvetica,sans-serif,Verdana
height	32
href	http://newtours.demoaut.com/mercuryu
html id	
html tag	A
innerHTML	featured vacation destinations
innerText	featured vacation destinations
name	featured vacation destinations
outerhtml	<A href=mercuryunderconst.php?osCsid
outertext	featured vacation destinations
target	
text	featured vacation destinations
title	
url	http://newtours.demoaut.com/mercuryu
visible	True
width	97
x	527
y	441

Below the table, there are two text input fields:

Selection:
Class Name

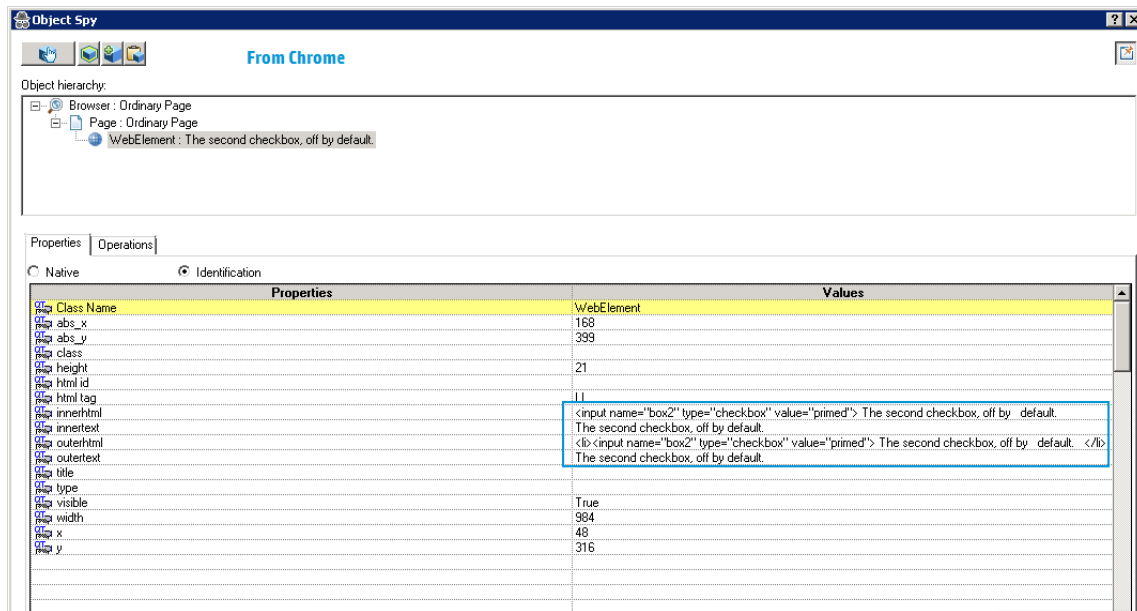
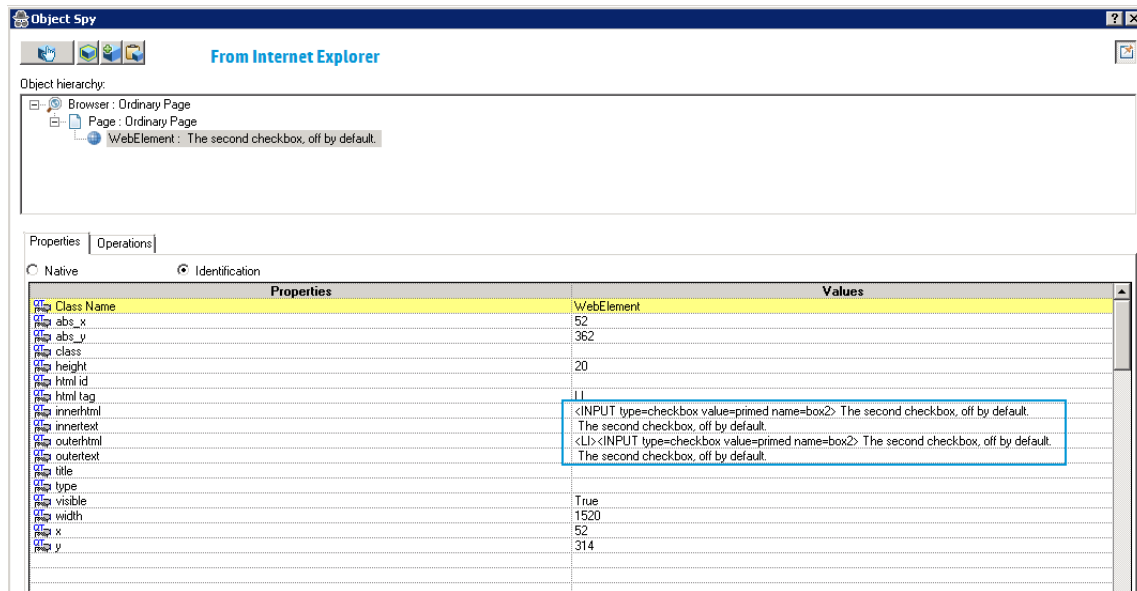
Description:
Descriptions are available only for test object operations.

At the bottom right, there is a "Close" button.



If you were using these properties to ensure correct identification of the link object, or using these properties in a checkpoint, you would achieve varying results.

- When using Chrome or Safari, the **innertext**, **outertext**, **innerhtml** and **outerhtml** property values may differ from other browsers:



Note that while the content of the innerhtml and outerhtml is basically the same - the browsers are definitely representing the properties differently. This can potentially cause identification problems for objects, especially if the property values are critical elements in identifying the object.

- In some cases, applications or Web sites detect which browser you are using and change the actual HTML content of the page. In this case, the identification properties that UFT uses to identify the object in the browser could be looking for a specific HTML tag, which may or may not exist in a different browser.

- Some HTML5 input types are not supported in older browsers or browser versions. This means that a control might have different properties, depending on the browser you use. If you record a test on a browser that supports HTML5 but run it on a browser that does not support HTML5, or vice versa, UFT might not be able to match the object description in your test to an object in the Web page being tested.

Testing Applications on Multiple Browsers - Creating a Single Test for All Browser Testing

When you are testing your application or Web page on multiple browsers, you are trying to see that the application or Web page performs the same between browser types (unless designed otherwise). When you set up your test, you would also expect that the same test can be used for each browser type, with minimal maintenance and updating to account for browser differences. However, creating a single test for cross browser testing provides a set of challenges:

<p>Having the correct objects with the correct properties for each browser</p>	<p>In UFT, a test of your application accesses the object repository or repositories for your test, which contains the necessary objects to test the application's objects. Ideally, you would create a single object repository or object repositories (per application section or Web page). Then, you would run the test in all the browser versions using that single object repository or set of object repositories, and UFT could identify the objects in the application or Web page without issue.</p> <p>In reality, providing the correct objects in the correct object repository is not always simple. Firstly, there are the object identification issues. If UFT identifies a certain object very differently between browser types or versions, you may need to create separate repositories for each browser type to enable UFT to find the right object. However, when the test runs on a specific browser, you need the correct object repository for that test run.</p> <p>When you have a scenario such as this, you can enable UFT to dynamically add an object repository at the beginning of the test run. For details, see "Dynamically load an object repository during the test run" on page 477.</p>
<p>Dynamically created objects that are not included in an object repository</p>	<p>In some cases, you may also have dynamically created objects that are displayed on a page as a result of previous operations performed in the application. However, in the process of creating the test and the object repositories, these objects are not identified (as they do not exist when UFT is learning the application initially). Like many other objects, these objects can be created and identified very different between browsers, making it even more difficult for UFT to identify them.</p> <p>To help UFT identify these objects, you can use descriptive programming. For details, see the section on Programmatic Descriptions in the <i>HP Unified Functional Testing User Guide</i>.</p>
<p>Dynamic page updates</p>	<p>Each browser version can also have dynamic updates to the browser or page as part of its normal workflow. For example, alert dialogs are different between each of the browser versions, making it difficult to UFT to know how to recognize, handle or ignore the dialogs. As a result, if you create test steps to close the alert dialog using one browser type, the other browser types may have trouble recognizing or performing steps on the dialog. In other cases, these dialogs do not exist in the other browsers. In some cases, the browser may not even enable you to continue the test if the alert dialog is not closed, thereby causing the test to fail, simply because it did not recognize the dialog and perform the correct steps on it.</p> <p>In the case of the browser dialog boxes, there are special methods that can ensure you handle the pop-up dialogs appropriately, including the Browser.HandleDialog, Browser.GetDialogText, and the Browser.DialogExists methods. (Note that even these methods do not work exactly the same between</p>

	<p>browsers also.)</p> <p>In other cases, you can add steps to your test to account for browser-specific behavior. For details, see "Add steps for browser specific behavior" on page 478.</p>
<p>Browser-specific or application-specific behavior for user actions</p>	<p>There are scenarios where the same application operation - such as pressing the Back button - may result in very different behavior between browsers. For example, in an appointment booking application, pressing the Back button for Internet Explorer returns you to the previously viewed page, while pressing Back in Firefox or Chrome logs you out of the application.</p> <p>However, because of such problems, your test must be prepared to address the different behaviors. There are some potential solutions to help:</p> <ul style="list-style-type: none"> • Navigate to a specific URL/location in the application instead of a previous/next page (such as in the example above) • Insert Wait steps that pause the test until an application or an application object achieves a certain state (which can be checked using the Exist property for an object) <p>For details, see "Add steps for browser specific behavior" on page 478</p>

Testing Applications on Multiple Browsers - Running the Test on Multiple Browsers

After you create a single test of your application or Web page to use in different browsers, you still must run it to actually test the application or Web page. You have a number of options on how to run the test across different browser types:

<p>Manually configure the browser type for each test run</p>	<p>UFT provides you the opportunity to select the browser type before each test run. You can do this in one of the following places:</p> <ul style="list-style-type: none"> • The Web tab of the Record and Run Settings dialog box. <p>In the Web tab, you can select the browser type from the drop-down list. Then, when you run the test, UFT opens the appropriate browser and runs the test.</p> <ul style="list-style-type: none"> • A user-defined environment variable specified in the Environment pane of the Test Settings dialog box. <p>UFT uses the BROWSER_ENV environment variable, and the requisite values for each browser type to enable you to set this variable before each test run. When you enter a value for the BROWSER_ENV variable, UFT automatically opens up the necessary browser (ignoring any other browser launch settings).</p> <p>However, this requires manual intervention by the person running the test, and does not enable you to run subsequent tests of the application or Web page on the different browser types in sequence.</p> <p>For details, see "Configure the Record and Run settings to launch a browser" on page 474 or "Set the value of the BROWSER_ENV environment variable to launch a specific browser" on page 475</p>
<p>Instruct UFT to open a browser type defined by a parameter in the test step</p>	<p>Instead of manually setting the browser before each test run (which thereby defeats the purpose of automated testing), you can insert a parameter into a test step that defines the browser to open. (You can also define a single reusable action that opens the browser, which can be reused in all the tests of your application or Web page.)</p> <p>The values for this parameter (which are the .exe programs for each browser) are then defined in</p>

	<p>the Data pane. When UFT reaches this test step, it reads the data pane and decides which browser needs to open based on the selected data.</p> <p>This removes the need for you or another person to manually configure settings or variables in a test before running the test, enabling you automatically test your application or Web page on all browser types and/or versions. If you create a reusable action with the step that opens the browser, the parameter and data must be added to every test that calls this external action.</p> <p>For task details on how to set up automatic opening of browser types from UFT, see "Configure UFT to automatically launch a browser using a data parameter" on page 476.</p>
Use a test parameter or data table parameter to launch the appropriate browser	<p>In the Record and Run Settings dialog box (Web tab), you can instruct UFT to use either a test parameter or data table parameter to launch the browser. You set the parameter at the beginning of the test run (for a test parameter) or insert the BROWSER_ENV value in the Data table in the Data pane.</p> <p>Then, when UFT runs the test, it launches the correct browser according to the values you inserted.</p> <p>For details, see "Configure UFT to automatically launch a browser with a parameter using the Record and Run Settings" on page 476.</p>

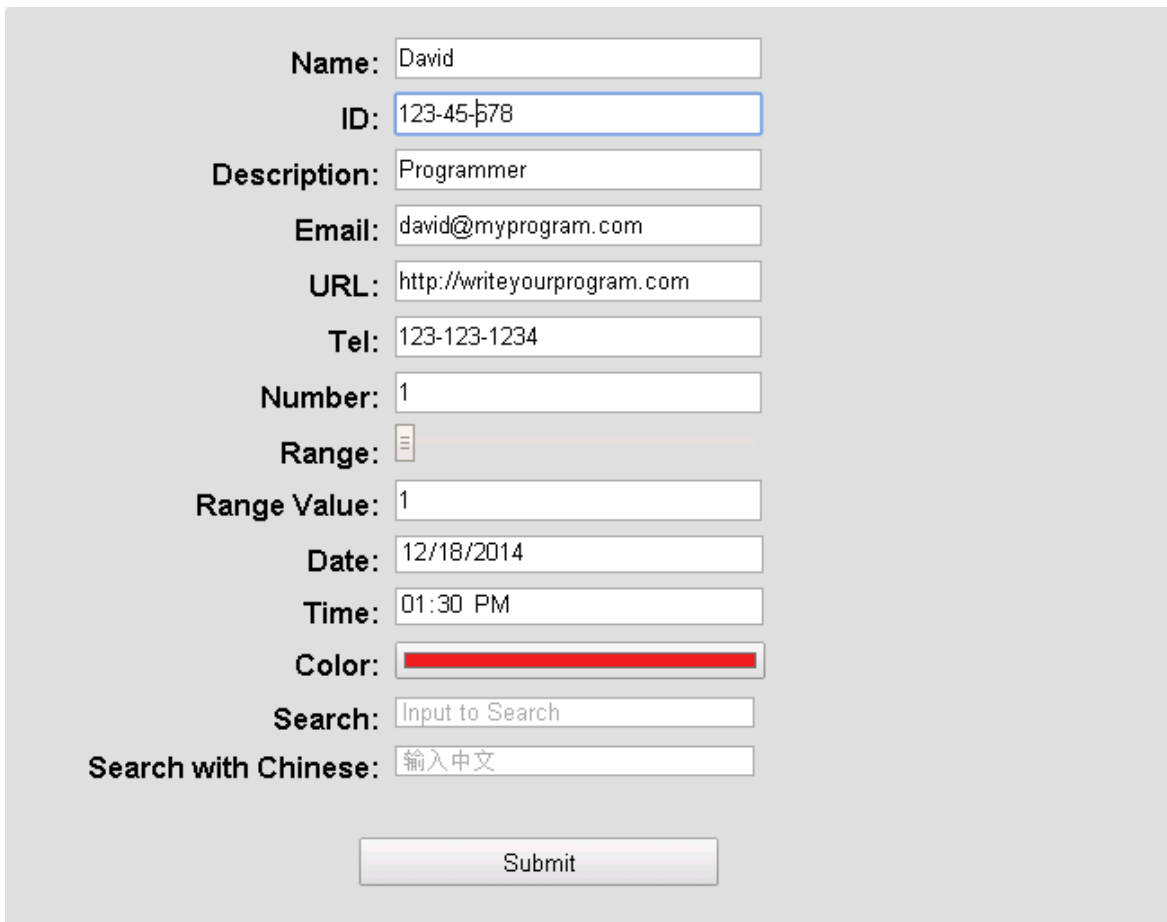
Using Descriptive Programming for Multiple Browser Testing - Use-case Scenario

One of the challenging parts of cross-browser testing of your applications or Web pages is the object identification of objects in different browser types. Since each browser type can read the HTML code of your application and translate this differently, UFT may have trouble identifying the same objects in different browsers.

One technique that you can use when UFT is not identifying objects correctly is descriptive programming. When you insert a programmatic description into your test instead of the actual test object name, UFT searches for the object in your application matching the description.

In this use-case scenario, you can see how UFT can find a problematic object using description instead of the test object name for the object (as stored in the object repository.).

In your application, you are trying to test this area of your application, containing a number of edit fields:

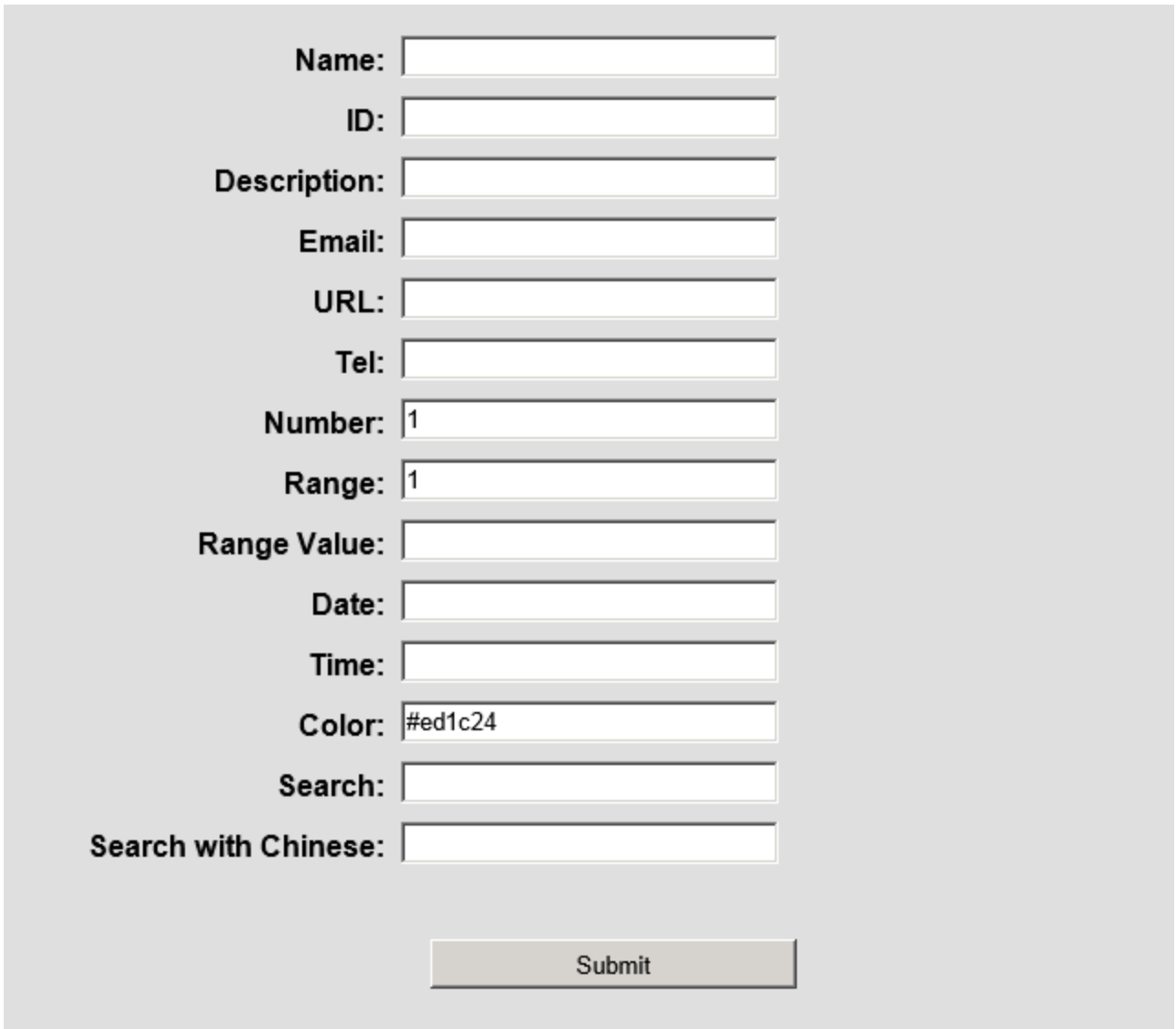


The image shows a web form with the following fields and values:

- Name:** David
- ID:** 123-45-678
- Description:** Programmer
- Email:** david@myprogram.com
- URL:** http://writeyourprogram.com
- Tel:** 123-123-1234
- Number:** 1
- Range:** (dropdown menu)
- Range Value:** 1
- Date:** 12/18/2014
- Time:** 01:30 PM
- Color:** (color picker showing red)
- Search:** Input to Search
- Search with Chinese:** 輸入中文

At the bottom of the form is a **Submit** button.

In Chrome and Firefox, the application area displays as seen above. However, in Internet Explorer, the window has a different appearance, particularly the **Color** field:

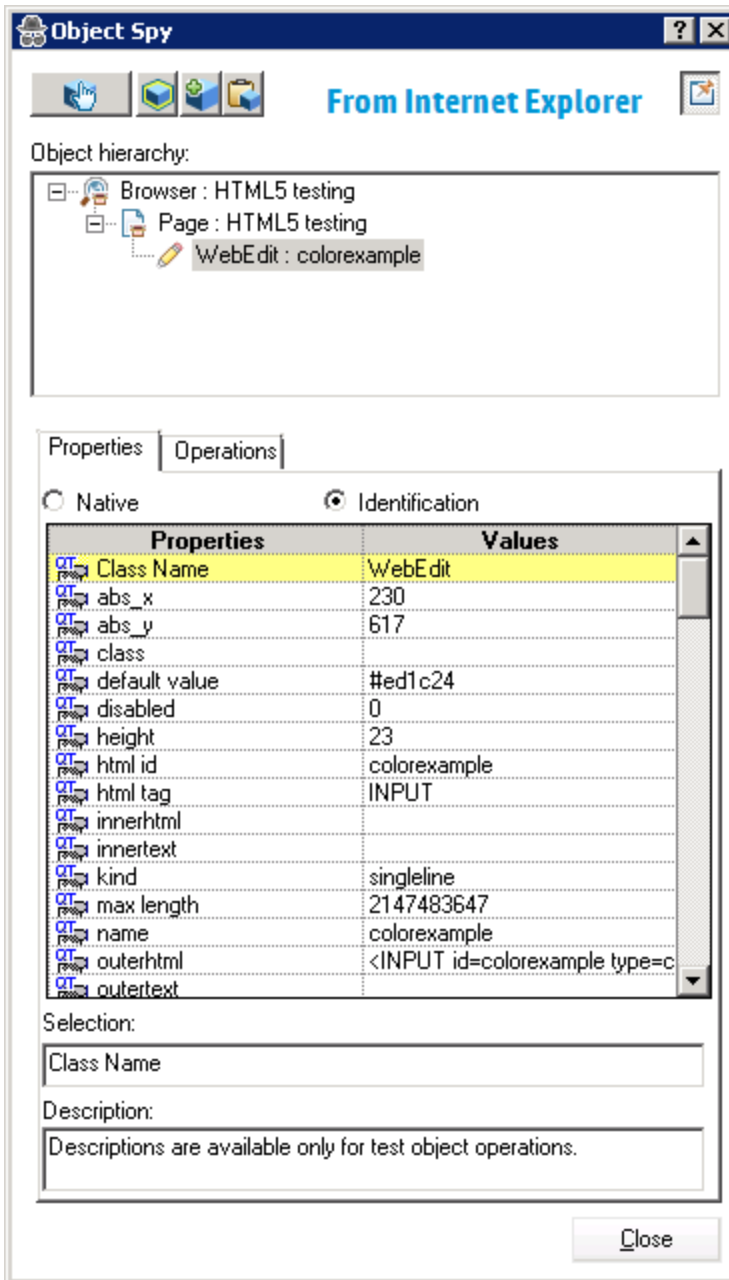


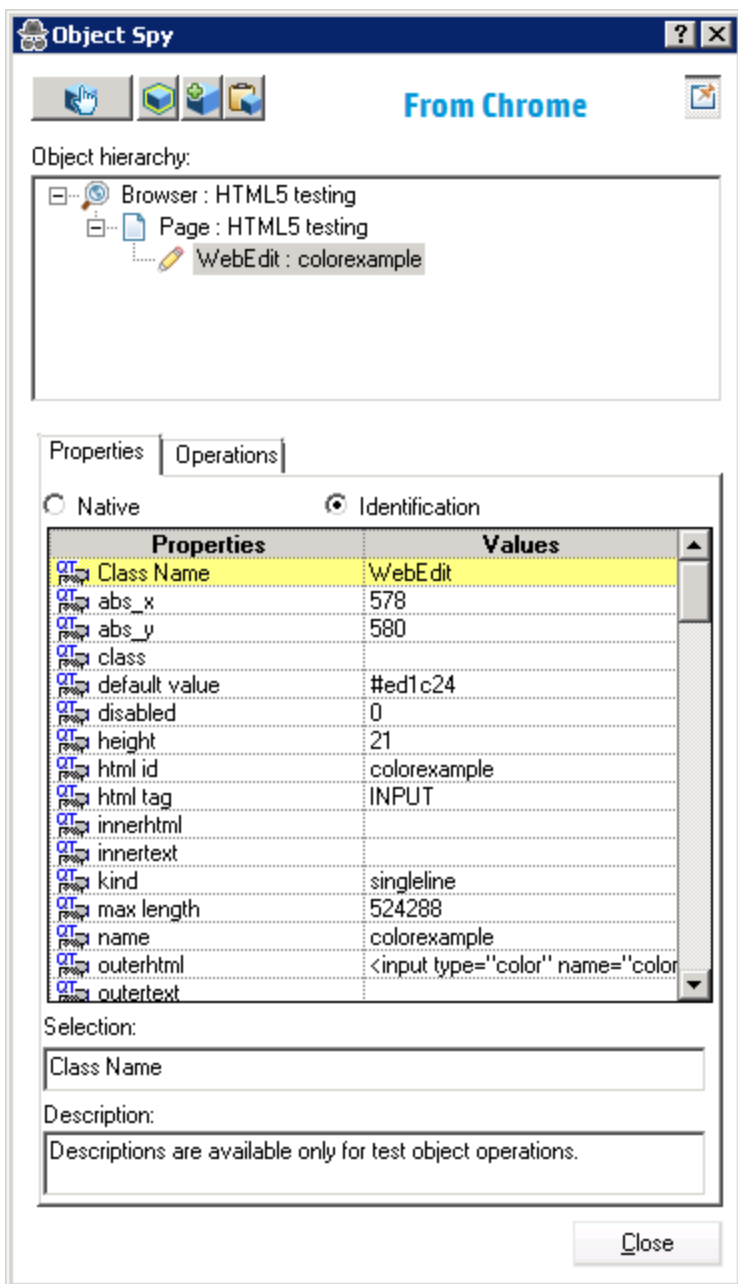
The image shows a web form with the following fields and values:

- Name:
- ID:
- Description:
- Email:
- URL:
- Tel:
- Number:
- Range:
- Range Value:
- Date:
- Time:
- Color:
- Search:
- Search with Chinese:

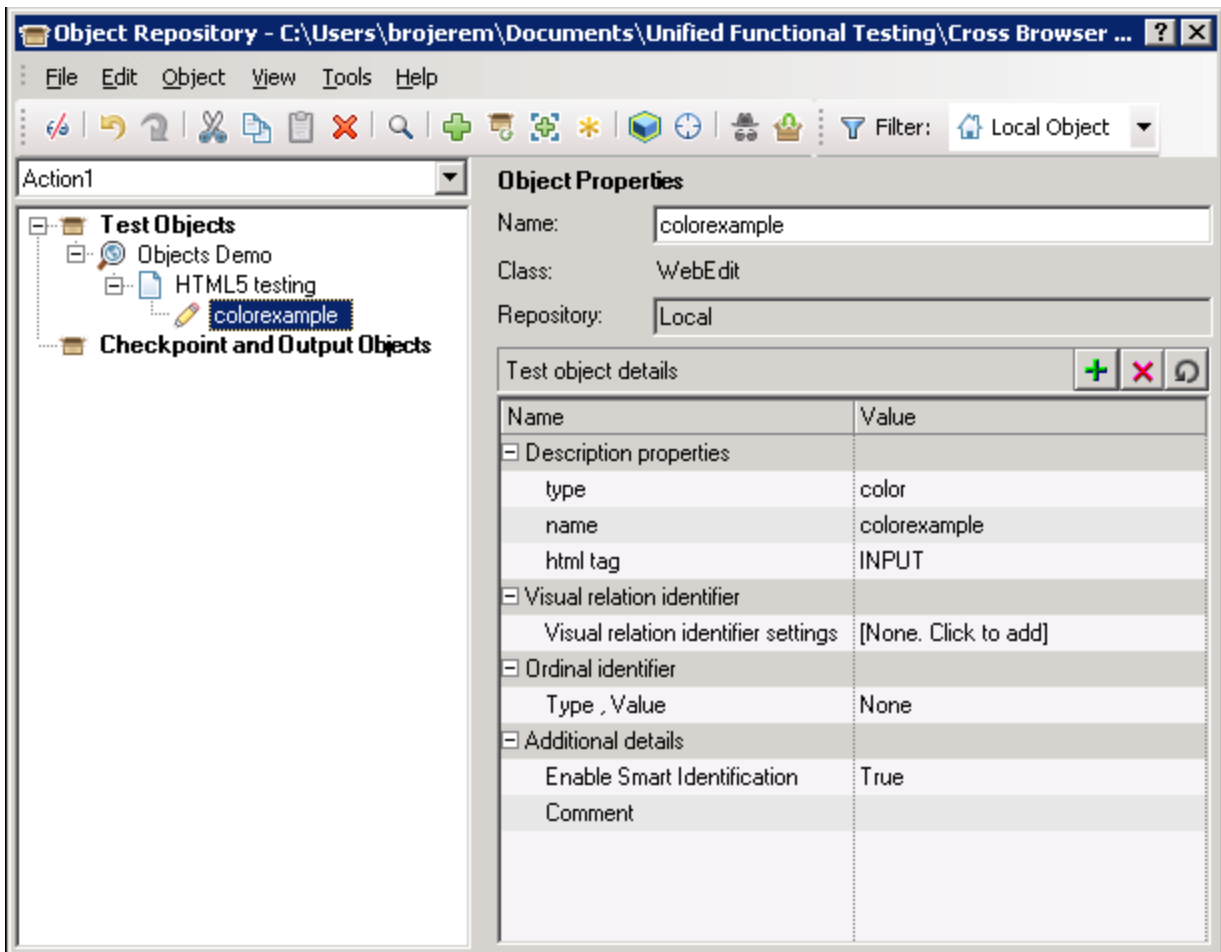
Submit

Even though the visual appearance is different, a closer look shows that the object properties of the **Color** field are basically the same:





In the object repository, the Color field object is recognized as a WebEdit object with the name **colorexample** (as it was by the Object Spy):

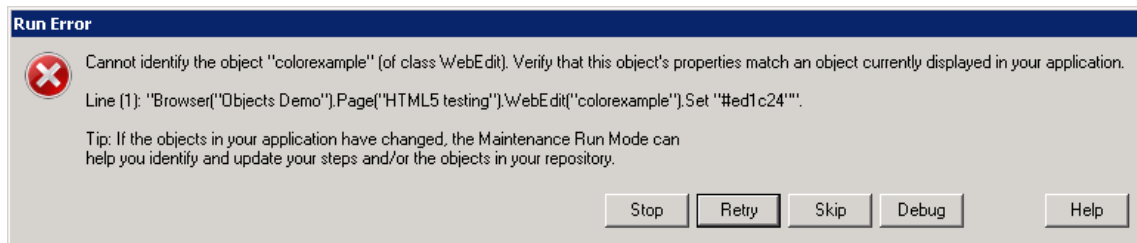


Based on this, when you insert a test step for this object, it is displayed like this:

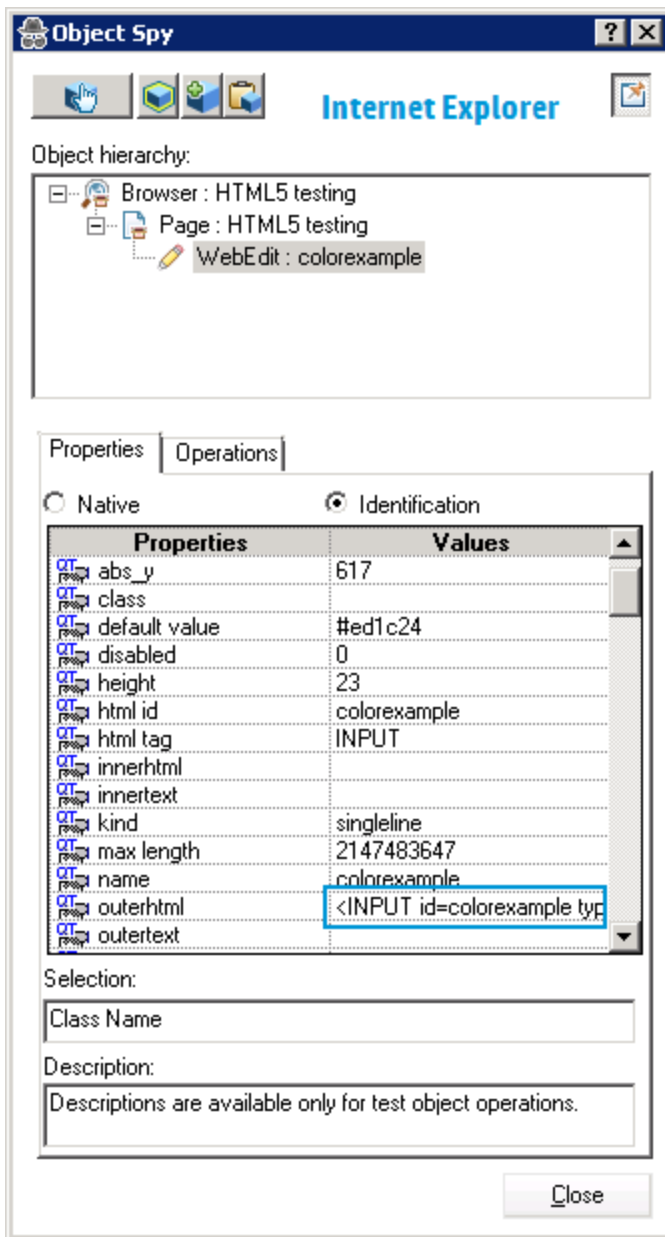
```
Browser("Objects Demo").Page("HTML5 testing").WebEdit("colorexample").Set
```

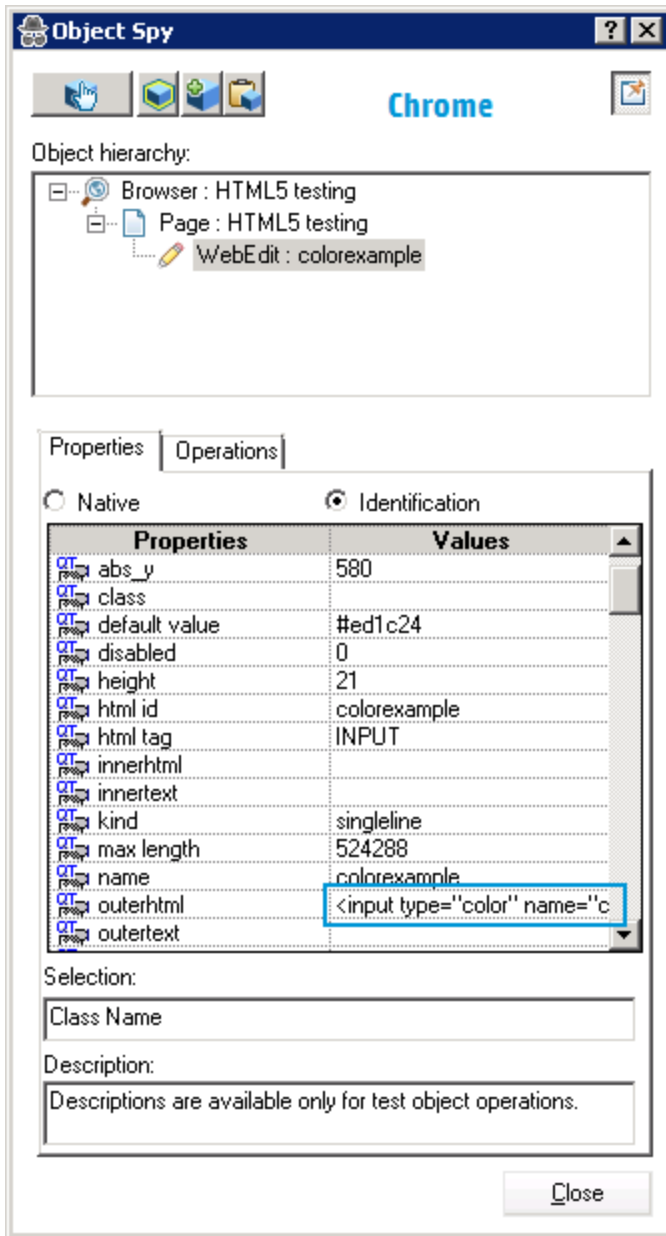
However, when you run the test step, there are varying results:

- The step runs on Chrome and Firefox without problem.
- The step fails on Internet Explorer:



A closer look at the properties in the Object Spy for the **colorexample** object shows slight differences in the properties between Internet Explorer and Chrome:





In this case, the property difference is causing UFT to not identify the object in Chrome. As a result, you can modify the step using descriptive programming:

```
Browser("Objects Demo").Page("HTML5 testing").WebEdit("name:=colorexample",  
"htmltag:=INPUT").Set "#ed1c24"
```

Using this statement, when the test runs, the step runs without a problem:

Step Name: [colorexample].Set

Step Done

Object	Details	Result	Time
[colorexample].Set	"#ed1c24"	Done	12/18/2014 - 13:25:29

Thus, by changing the test step to use descriptive programming, UFT is able to identify the object and run the test step across all browsers.

Working With Mozilla Firefox

UFT tests and business 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 general considerations on working with Web browsers, see ["Considerations - Web-Based Application Support" on page 35](#).

The following sections describe specific items to consider when working with Mozilla Firefox:

- ["Running Steps on Mozilla Firefox Browser Controls" below](#)
- ["Running Steps on Mozilla Firefox Dialog Boxes" on the next page](#)
- ["Checkpoints and Output Values" on the next page](#)

Running Steps on Mozilla Firefox Browser Controls

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:

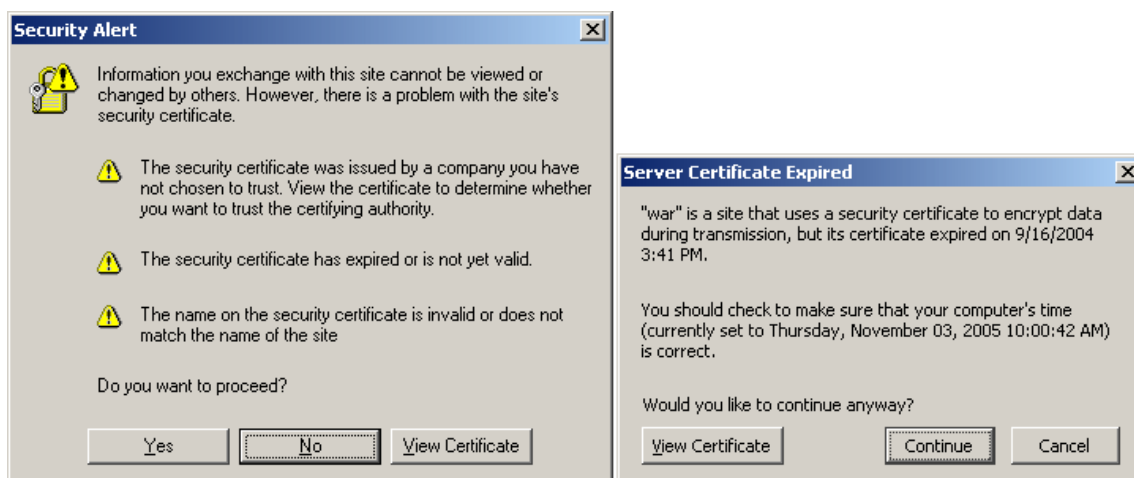
- UFT does not support Mozilla Firefox menus or sidebars.
- UFT supports specific Browser menu operations that are represented by the following toolbar buttons:
 - **Back**
 - **Forward**
 - **Home**
 - **Refresh**
 - **Stop**

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 business component on Mozilla Firefox.

Running Steps on Mozilla Firefox Dialog Boxes

- 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.



Checkpoints and Output Values

Cross-browser UFT operations may affect the results of some types of checkpoints and output values:

- 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 details on regular expressions, see the section on understanding and using regular expressions in the *HP Unified Functional Testing 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 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 **Text Recognition** pane of the Options dialog box (**Tools > Options > GUI Testing** tab > **Text Recognition** node).

Working With Apple Safari on a Remote Mac Computer

You can use UFT to test Web applications on an Apple Safari browser that is running on a remote Mac computer. UFT uses the WebSockets protocol to connect to the Mac computer that you specify. Note that only Web test object steps can run on Safari. All other steps, including Utility object steps, such as **SystemUtil.Run**, run locally on the UFT computer.

To test Web applications on the Safari browser, you must install the UFT Connection Agent and the Unified Functional Testing Agent Safari browser extension on your Mac computer. For details on how to do this, see "[The UFT Connection Agent for Mac Computers](#)" on page 471.

Recording steps and learning objects (in the Object Repository Window or Manager) are not supported on the Safari browser. However, you can use the [Remote Object Spy](#) on a Safari browser to view the properties and operations of Web controls, and optionally, add the corresponding test objects to your object repository.

Creating and Editing Tests and Components that Run on Safari

It may be more convenient to create and edit your object repositories, tests and components working with a supported browser installed locally on the UFT computer, and then [connect to a remote Mac computer](#), fine-tune your tests, and run them on Safari.

Tip: Use Google Chrome to create, edit, and debug your tests and components, as Chrome and Safari render Web pages similarly.

Once your basic test is designed, you can [connect UFT to a remote Mac computer](#) running Safari, and fine-tune or debug your test based on the object properties available on Safari:

- Use the Remote Object Spy to see how UFT recognizes the objects in your application.
- View object properties using statement completion for the **Object** method.

Note: Statement completion for the Object method is only available if the connection to the remote Mac is fast enough.

- Create standard checkpoints and output value steps on objects displayed in the Safari browser. Use the **Design > Checkpoint** and **Design > Output Value > Standard Output Value** commands.

Multiple UFT-Mac Connections

- Multiple users from different UFT instances can connect to the same Mac simultaneously using different ports.

Each user must install the UFT Connection Agent and UFT Safari extension in their Mac user account, and configure the relevant port number in UFT, in the connection agent on the Mac.

- The Mac connection information is in the Record and Run settings, which are defined per test, therefore you can connect to different Macs or different ports when running different tests.

However, UFT can connect to only one Mac at any time.

How UFT Handles Connection Attempts When a Previous Connection Exists

UFT attempts to set up a remote connection using the RemoteConnection.Connect method, but...	Result
... it is already connected to the same Mac and port with the same security level .	The existing connection is used and a statement about this is added to the test results.
... it is already connected using a different Mac, port, or security level .	A run error occurs on the Connect step

UFT attempts to set up a remote connection using the Record and Run settings, but...	Result
... it is already connected with the same security level or higher (to the same Mac and port, or different ones). Note: This means UFT initiates a connection without SSL, or it is initiating an SSL connection and the existing connection is also using SSL.	The existing connection is used and a warning is added to the test results. Note: This may lead to a situation where steps that were to be carried out on the new connection on a different Mac or port, are carried out on the existing one.
... it is already connected with a lower security level (to the same Mac and port, or different ones). Note: This means UFT initiates an SSL connection but the existing connection is not using SSL.	The existing connection is closed and a new connection is initiated. (The new connection can succeed only if the UFT Connection Agent on the Mac is also set up to require an SSL connection.) Notifications about the connection changes are added to the test results. Note: This behavior prevents communication intended for a secure connection from being carried out on a non-secured connection.

For details about how to connect UFT to the remote Mac computer, see ["How to Connect to a Remote Mac Computer"](#) on page 480.

For additional details about working with UFT and the Safari browser, see ["Troubleshooting and Limitations - Google Chrome and Apple Safari"](#) on page 491.

For general considerations on working with Web browsers, see ["Considerations - Web-Based Application Support"](#) on page 35.

The UFT Connection Agent for Mac Computers

This section includes:

- ["What is the UFT Connection Agent?" below](#)
- ["What happens when UFT activates the Remote Object Spy?" below](#)
- ["What Do I Need to Do to Upgrade from UFT 12.00" below](#)
- ["What Do I Need to Configure on the Mac so UFT Can Test Web Applications?" on the next page](#)
- ["How Do I Secure the Communication Between UFT and the Mac Computer" on the next page](#)

What is the UFT Connection Agent?



The UFT Connection Agent is a service installed on your Apple Mac computer, which enables UFT to communicate with the Safari browser using the WebSockets protocol. This enables UFT to run tests on Web applications running in Safari on the Mac and to spy on objects in these applications.

When you install the UFT Connection Agent, the agent service is installed on your Mac, and the Unified Functional Testing Agent extension is installed on the Apple Safari browser.

The UFT Connection Agent runs automatically after installation and after each restart of the Mac and communicates with UFT on the one hand and the Unified Functional Testing Agent Safari extension on the other.

What happens when UFT activates the Remote Object Spy?

When UFT initiates a Spy session (using the Remote Object Spy), the mouse on the Mac computer can be used only for spying on Web objects in Safari.

The Unified Functional Testing Agent extension icon  in the Safari toolbar changes to a UFT Spy button , which you can use to suspend and then resume the Spy session.

For more details on using UFT to spy on Web objects in Safari, see the section on the Remote Object Spy in the *HP Unified Functional Testing User Guide*.

What Do I Need to Do to Upgrade from UFT 12.00

If you previously used UFT 12.00 to test Web applications on Safari, then after you upgrade UFT you must reinstall the UFT Connection Agent on the Mac from the current UFT version.

The UFT Connection Agent preferences and the Unified Functional Testing Agent Safari extension preferences are reset to their defaults.

If you want to use non-default preferences, for example, if you configured UFT to use a port other than the default 8822 for remote connections, then you must reconfigure these preferences on the Mac computer.

This is required because of significant changes in the UFT Connection Agent's preferences. For example, the UFT Connection Agent now uses different ports to communicate with UFT and the Safari extension. In UFT 12.00, the same port was used for both communications.

What Do I Need to Configure on the Mac so UFT Can Test Web Applications?

By default, you do not need to modify the preferences set for the UFT Connection Agent and the Unified Functional Testing Agent extension.

- UFT and the connection agent communicate using port **8822**
- The connection agent and the UFT Safari extension communicate using port **8823**.

However, in some situations, you might want to specify different ports. For example:

- If another application on the Mac uses these ports.
- If multiple UFT users need to connect to the same Mac computer. Each UFT user can use a connection agent installed under a different Mac user account, with a different port number configured.

Caution: If you modify the port numbers, make sure that:

- The **UFT port** defined in the UFT Connection Agent preferences matches the one defined in UFT. For details on setting this in UFT, see ["How to Connect to a Remote Mac Computer" on page 480](#).
- The **Safari port** defined in the UFT Connection Agent preferences matches the one defined in the Unified Functional Testing Agent extension in Safari.

In addition to the port numbers, you can configure the level of log messages to collect. By default, log messages are collected from the connection agent, but not the Safari extension.

How Do I Secure the Communication Between UFT and the Mac Computer

When UFT communicates with the Mac, UFT acts as a client and the UFT Connection Agent acts as a server.

You can secure this communication on different levels:

1. You can set up client authentication by defining a passphrase for UFT to use when contacting the Mac.
2. You can secure the communication between UFT and the UFT Connection Agent by requiring that they use an SSL connection.

For more details, see ["Securing the Communication With the Remote Mac Computer" on the next page](#)

For task details on configuring the UFT Connection Agent preferences, the Unified Functional Testing Agent Safari extension preferences, and the security settings, see ["Configure the UFT Connection Agent Preferences \(Optional\)" on page 483](#).

Securing the Communication With the Remote Mac Computer

When UFT connects to a remote Mac computer, it can access the Safari application and perform steps on Web applications running in Safari. Therefore, it is important to secure this connection, to prevent inappropriate access to your Mac and Web pages that the Mac can access.

When UFT communicates with the Mac, UFT acts as a client and the UFT Connection Agent acts as a server.

You can secure this communication on different levels. You can:

Set up client authentication by defining a passphrase for UFT to use when connecting to the Mac.

Define the same passphrase in UFT's **Remote Connection** pane in the Options dialog box (**Tools > Options > GUI Testing > Remote Connection**) and in the UFT Connection Agent's preferences.

By default, the passphrase is empty. To provide better security, provide a passphrase that is long, complicated, and difficult to guess.

UFT uses this passphrase whenever it initiates a connection with any Mac computer.

The UFT Connection Agent accepts a connection request only if the passphrase included in the request matches the passphrase defined in the agent's preferences.

You can define the same passphrase on multiple UFT instances (on different computers, or in different user accounts on the same computer), and multiple instances of the UFT Connection Agent, (on different Mac computers or in different Mac user accounts). This way, you can set up a group of computers that all share the same passphrase and are used for similar testing purposes (like a virtual lab).

Secure the communication between UFT (the client) and the UFT Connection Agent (the server) by requiring that they use an SSL connection.

In the UFT Connection Agent preferences, set the following:

- Specify whether communications with this agent must take place over SSL connections (**use SSL**)
- If they must, then:
 - Specify the path to an **SSL certificate file** for the server to use for the communication. (Chain certificate files are also supported)
 - Specify the path to the **SSL private key** that matches the certificate.

In UFT:

- In the **Remote Connection** pane in the Options dialog box (**Tools > Options > GUI Testing > Remote Connection**), specify the path to an **SSL CA certificate file** that UFT can use to validate the SSL certificate provided by the server. (Certificate bundles are also supported.)

- When defining the details for a specific remote connection, you specify whether the connection should **use SSL**. This can be done in the Remote Connection dialog box, in the Record and Run settings (on a per-test basis), or using the RemoteConnection utility method in a test step. For details, see ["How to Connect to a Remote Mac Computer" on page 480](#).
- For a connection to succeed, the **use SSL** option must have the same value in UFT and the UFT Connection Agent.

At different times, UFT can connect to different UFT Connection Agents, by using different Mac or port numbers. You can instruct UFT to initiate non-SSL connections with agents that you know do not require SSL, and SSL connections with agents that you know require it. UFT uses the CA certificate file for certificate validation only on connections initiated using SSL.

How to Set Up Multiple Browser Testing

This task describes some of the techniques you can use to enable effective cross-browser testing for your application or Web page.

Tip: For a use-case scenario related to this task, see ["Using Descriptive Programming for Multiple Browser Testing - Use-case Scenario" on page 459](#).

This task includes the following steps:

- ["Prerequisite- turn off auto updates for the browsers" below](#)
- ["Configure the Record and Run settings to launch a browser" below](#)
- ["Set the value of the BROWSER_ENV environment variable to launch a specific browser" on the next page](#)
- ["Configure UFT to automatically launch a browser with a parameter using the Record and Run Settings" on page 476](#)
- ["Configure UFT to automatically launch a browser using a data parameter" on page 476](#)
- ["Dynamically load an object repository during the test run" on page 477](#)
- ["Add steps for browser specific behavior" on page 478](#)

Prerequisite- turn off auto updates for the browsers

To ensure that you are testing on the specific browser version you want, turn off the automatic update feature for your browser.

Configure the Record and Run settings to launch a browser

Using the Record and Run settings, you can change the browser on which you run the test for each test run.

1. Select **Record > Record and Run Settings**.
2. In the Record and Run Settings dialog box, select the **Web** tab.


3. In the Web tab, select the **Open the following address when a record or run session begins:** option.
4. In the web address drop-down list, enter a Web address to open or select a Web address from the drop-down list.
5. In the **Open the following browser when a record or run session begins:** drop-down list, select the browser on which you want to run your test.

Note: If you choose Apple Safari, you must provide additional connection information. For details, see "[How to Connect to a Remote Mac Computer](#)" on page 480

When you start the test run, the specified browser opens the Web address entered.

Set the value of the **BROWSER_ENV** environment variable to launch a specific browser

Using the **BROWSER_ENV** environment value, you can change the browser to launch each test run, and in some cases specify a specific browser version (if installed).

1. Select **File > Settings**.
2. In the Settings dialog box, select the **Environment** node.
3. In the Environment node, from the **Variable type** drop-down list, select **User-defined**.
4. In the user-defined variables list, click the **Add** button .
5. In the Add New Environment Parameter dialog box, in the **Name** field, enter **BROWSER_ENV** (case-sensitive).
6. In the **Value** field, enter the value (case-sensitive) for the browser to open:

IE	Opens the installed version of Internet Explorer.
IE64	Opens the installed 64-bit version of Internet Explorer.
CHROME	Opens the installed version of Google Chrome.
FIREFOX	Opens the latest installed and supported version of Firefox.
FIREFOX64	Opens the latest version of 64-bit Mozilla Firefox that is both installed on the computer and supported by UFT.
FF<version#>	Opens a specified version of Firefox.
SAFARI	Opens Safari on the remote Mac computer connected to UFT.

7. Click **OK** to save the Name and Value of the variable.
8. In the Test Settings dialog box, click **Apply** and **OK** to save the variable and close the dialog box.


Configure UFT to automatically launch a browser with a parameter using the Record and Run Settings

You can instruct UFT to automatically launch a specific browser for a test run using the Record and Run Settings:

1. Select **Record > Record and Run Settings**.
2. In the Record and Run Settings dialog box, select the **Web** tab.
3. In the Web tab, select the **Open the following when a record and run session begins:** option.
4. (Optional) In the **Address** field, enter the address to which to open the browser.
 You can leave the **Browser** drop-down list as the default value.
5. In the **Parameter type** drop-down list, select the parameter type: **Global Data Table** or **Test Parameter**.
6. In the **Parameter Name** field, enter the parameter name. (The default parameter name is Browser.)
7. Click **Apply** to save the changes and **OK** to close the dialog box.

UFT automatically adds a column to the Global tab in the Data pane or a test parameter.

8. Before running the test, do one of the following, depending on the parameter type selected:

For a Global Data Table parameter	In the Global tab of the Data pane, set the value of the parameter.	Value to use: <ul style="list-style-type: none"> • IE. Opens Internet Explorer. • IE64. Opens a 64-bit version of Internet Explorer. • CHROME. Opens Google Chrome. • FIREFOX. Opens the latest version of Mozilla Firefox that is both installed on the computer and supported by UFT. • FIREFOX64. Opens the latest version of 64-bit Mozilla Firefox that is both installed on the computer and supported by UFT. • FF<VersionNumber>. Opens the specified version of Mozilla Firefox. For example: FF36 (version 3.6), FF40 (version 4.0), FF140 (version 14.0). • SAFARI. Opens Safari on the remote Mac computer connected to UFT (defined in the Web tab of the Record and Run Settings dialog box or in the REMOTE_HOST environment variable).
For a Test Parameter	<ol style="list-style-type: none"> a. In the toolbar, click the Run button . b. In the Run dialog box, select the Input Parameters tab. c. In the Value column for the parameter, enter the value of the parameter 	

Configure UFT to automatically launch a browser using a data parameter

1. (Optional) Create a reusable action to use in all your tests for launching the browsers.
2. In the Data pane, open the **Global** tab.
3. In the Global tab, double-click the header of the first column in the table or the first column of the column where you want to store the parameter.

4. In the Change Parameter Name dialog box, enter the name for the parameter and click **OK**.
For example, you could name this parameter **BrowserName** (to identify it as the name of the browser to open).

The column name of the selected column is renamed to reflect the renamed parameter.

5. In the data table, enter the .exe names for the browsers you want to open.
For example, if you need to run the test on Internet Explorer, Firefox, and Chrome, you would enter `iexplore.exe`, `firefox.exe`, and `chrome.exe` in the first three rows of the column, respectively:

A4		
	BrowserName	B
1	iexplore.exe	
2	firefox.exe	
3	chrome.exe	
4		
5		
6		

6. Add a test step with the following format:

```
SystemUtil.Run DataTable("<parameter name>", dtGlobalSheet), <address to open the browser to>
```

For example, if you wanted to open up to the Mercury Tours site, you could enter the following:

```
SystemUtil.Run DataTable("BrowserName", dtGlobalSheet),  
http://newtours.demoaut.com
```

Note: If you add this step to a reusable action which is then called by other tests, the relevant rows must be added in the Global tab of all tests that call that action.

Dynamically load an object repository during the test run

If your test requires you to have different object repositories for each browser type, you can load the relevant object repositories as part of the test run without having to manually configure anything before the test run:

1. In the Data pane, open the **Global** tab.
2. In the Global tab, double-click the header of the first column in the table or the first column of the column where you want to store the parameter.
3. In the Change Parameter Name dialog box, enter the name for the parameter and click **OK**.
For example, you could name this parameter **Browser** (to identify it as the name of the browser on which to run the test).
The column name of the selected column is renamed to reflect the renamed parameter.

4. In the data table, enter the names for the browsers on which you want to run the test.
5. Add a test step with the following format:

```
If DataTable("<data table parameter>") = <Browser 1> Then
    RepositoriesCollection.Add "<location to object repository>"
ElseIf DataTable("<data table parameter>") = <Browser 2> Then
    RepositoriesCollection.Add "<location to object repository>"
End If
```

Example

```
If DataTable("Browser") = "Internet Explorer" Then
    RepositoriesCollection.Add "C:\IE.tsr"
ElseIf DataTable("Browser") = "Chrome" Then
    RepositoriesCollection.Add "C:\Chrome.tsr"
End If
```

6. Add the additional steps for the application/Web page.

When the test runs, the appropriate object repository loads, and the test steps use the objects in the loaded object repository.

Add steps for browser specific behavior

If you need to add steps to perform browser specific behavior in the course of the test, you can use test parameters to create steps for this behavior.

1. In the canvas, select an action.
2. In the Properties pane, select the **Parameters** tab.
3. In the Parameters tab, click the **Add** button.
4. In the Add Parameter dialog box, provide a name for the parameter. For example, you could name the parameter **ActiveBrowser** to show that the value of the parameter represents the browser currently in use.
5. Add steps to the test. You can use the value of the parameter by using the Parameter object:

```
Select Case Parameter("<parameter name>")
    Case "<Browser 1>"
        'Do something specific for browser 1
    Case "<Browser 2>"
        'Do something specific for browser 2
End Select
```

Note: You can add additional Case statement as needed for each browser type.

When the test runs, the test steps run as specified in the necessary **Case** statement.

How to Enable the HP Functional Testing Agent Chrome Extension

Note: For the most up-to-date list of supported versions of Google Chrome, see the *HP Unified Functional Testing Product Availability Matrix*, available from the [HP Support Matrix page](#) (requires an HP passport).

UFT communicates with the Functional Testing Agent Chrome Extension to test Web applications running in Google Chrome.

IMPORTANT: If you have a version of the Functional Testing Agent for Google Chrome from version 12.00 or earlier installed, you must manually remove this extension before enabling the new version.

The following table describes the ways to enable the extension:

Internet Connection?	Chrome Auto Updates?	Chrome Version	Extension Enablement
Yes	Yes	31 or later	Automatic: Google Chrome automatically downloads and installs the Agent the first time you open your Chrome browser. If you do not have an internet connection at that moment, Chrome will try to download and install the UFT Agent each time you open Chrome. After downloading the Agent, Chrome prompts you to activate the Agent.
Yes	No	31 or later	Manually: Install and enable the Extension from the Web Store: https://chrome.google.com/webstore/detail/kgpdpdnaoephdehalonapacdghamnb
No	N/A	31 or later	Manually: See steps below to enable the Unified Functional Testing Agent Chrome Extension.
N/A	N/A	30 or earlier	Manually: See steps below to enable the Unified Functional Testing Agent Chrome Extension.

For Google Chrome versions 31 and later

1. Select **Tools > Extensions** in Google Chrome.

Note: You can also access this page by opening the **chrome://extensions** page in the Google Chrome browser.

The Extensions page opens.

2. In the Extensions page, select the **Developer mode** option. Additional options are displayed after you select this option.
3. Click on the **Load unpacked extension** button.
4. In the Browse for Folder dialog, navigate to the **<UFT installation folder>\Installations\Chrome** folder and select the **AgentLegacy.crx** file.
5. In the Confirm New Extension dialog, click **Add** when prompted,
The Functional Testing Agent is now displayed in the Chrome extensions list.

For Google Chrome versions 30 and earlier

1. If you have a previous version of the UFT Agent for Google Chrome installed, manually remove this extension.
2. Open the Chrome folder included with the UFT installation, found at **<UFT installation folder>\Installations\Chrome**.
3. Select **Tools > Extensions** in Google Chrome.

Note: You can also access this page by opening the **chrome://extensions** page in Google Chrome. The Extensions page opens.

4. From the **<UFT installation folder>\Installations\Chrome** folder, drag the **AgentLegacy.crx** file to the **chrome://extension** page.
5. In the Confirm New Extension dialog, click **Add** to install the extension.
The Functional Testing Agent for Google Chrome is now displayed in the Chrome extensions list.

How to Enable UFT to Test Local HTML Pages in Google Chrome

By default, the ability to run extensions on local HTML files is disabled in Google Chrome. Do the following to allow the UFT Google Chrome extension to run on local HTML files:

1. In Google Chrome, browse to the following URL: `chrome://extensions`
2. Locate the UFT extension, named Unified Functional Testing Agent.
3. Click the arrow ► located to the left of the icon to expand details about the extension.
Select **Allow access to file URLs**. Your selection is automatically saved.

How to Connect to a Remote Mac Computer

This task describes how to control the UFT connection to a remote Mac computer, to enable testing Web applications on the Safari browser.

For details on how to work with a connected remote computer and how UFT handles connection attempts when a previous connection exists, see ["Working With Apple Safari on a Remote Mac Computer" on page 469](#).


This task includes the following steps:

- ["Prerequisite" below](#)
- ["Use the Remote Connection dialog box to control the connection to the Mac while designing your test" below](#)
- ["In the Record and Run Settings dialog box, specify the remote Mac computer to use when running the test or component" on the next page](#)
- ["Alternatively, you can add steps to your test or component to control the remote connection to the Mac during the run session" on the next page](#)
- ["Configuring the Port Number to Use for the UFT-Mac Connection" on the next page](#)

Prerequisite

To test Web applications on the Safari browser, you must install the UFT Connection Agent and the Unified Functional Testing Agent Safari browser extension on your Mac computer. For details on how to do this, see ["The UFT Connection Agent for Mac Computers" on page 471](#).

Use the Remote Connection dialog box to control the connection to the Mac while designing your test

- Click the Remote Connection button  in UFT's toolbar.
In the dialog that opens, enter the host name or IP Address to use for the Mac.
Optionally, append a port number to the host name. For details, see ["Configuring the Port Number to Use for the UFT-Mac Connection" on the next page](#).
- If the UFT Connection Agent on the Mac is configured to expect an SSL connection, you must select the **Use SSL** option.
To use an SSL connection to secure the communication between UFT and the Mac, make sure that the relevant certificates and key are defined in the Options dialog box in UFT (**Tools > Options > GUI Testing > Remote Connection**) and in the UFT Connection Agent preferences on the Mac. For details, see the ["Securing the Communication With the Remote Mac Computer" on page 473](#).
- Use the **Connect/Disconnect** button on this dialog to control the connection status while you edit the test.
- This dialog also displays the current status of the connection.
- Before you run the test, make sure that you use one of the methods below to set up the UFT-Mac connection for the run session.

In the Record and Run Settings dialog box, specify the remote Mac computer to use when running the test or component

Do one of the following:

- Select **Record > Record and Run Settings** to open the Record and Run Settings dialog box.
 - a. In the Web tab, select **Open the following browser when a record or run session begins**.
 - b. Select **Apple Safari (on remote Mac computer)** from the list of browsers.
 - c. Set the host (and port) information.
 - d. Select whether to initiate an SSL connection.
 - e. Select whether to disconnect from the Mac at the end of the run session, and whether to close the browser.

For details, see "[Web Tab \(Record and Run Settings Dialog Box\)](#)" on page 71.

- Set the environment variables **REMOTE_HOST**, **BROWSER_ENV**, **URL_ENV**, and **USE_SSL** (the last two being optional).

In these environment variables, you specify the Mac connection details, the SAFARI browser, and, optionally, the URL to open in the browser.

If you need an SSL-secured connection, set the **USE_SSL** variable to TRUE. For details, see "[Environment Variables for a Web-Based Environment](#)" on page 55.

UFT sets up the connection with the specified Mac and runs the Safari browser at the beginning of the run session, whether the test runs it from UFT's UI, or from an ALM test set.

Alternatively, you can add steps to your test or component to control the remote connection to the Mac during the run session

Use the **RemoteConnection** utility object and its methods: **Connect** (*hostname*, [*useSSL*]), **Disconnect**, **IsConnected** to set up and disconnect the connection with the Mac.

Use `RemoteConnection.Run (Safari, <URL>)`, to run the Safari browser on the remote Mac after you establish the connection.

For details, see the **Utility Objects** section of the *HP UFT Object Model Reference for GUI Testing*.

Configuring the Port Number to Use for the UFT-Mac Connection

By default, UFT connects to the Mac using port **8822**. However, in some situations, you might want to use a different port. For example:

- If another application on the Mac uses this port.
- If multiple UFT users need to connect to the same Mac computer. Each UFT user can use a connection agent installed under a different Mac user account, with a different port number configured.

To use a different port, append the port number to the host name: *<hostname>:<port number>*.

Make sure to configure the same port number on the Mac, in the **UFT port** option in the UFT Connection Agent preferences.

How to Install and Configure UFT Connection Agent on Your Mac

The UFT Connection Agent is a service installed on your Apple Mac computer, which enables UFT to communicate with the Safari browser using the WebSockets protocol. This enables UFT to run tests on Web applications running in Safari on the Mac and to spy on objects in these applications. For details, see ["The UFT Connection Agent for Mac Computers" on page 471](#).

This task describes installing and configuring the UFT Connection Agent, and includes the following steps:

- ["Install or Uninstall the UFT Connection Agent" below](#)
- ["Configure the UFT Connection Agent Preferences \(Optional\)" below](#)
- ["Configure the Unified Functional Testing Agent Extension in Safari \(Optional\)" on page 485](#)
- ["Troubleshooting the UFT Connection Agent" on page 486](#)

Install or Uninstall the UFT Connection Agent

- You must have administrator permissions to install the UFT Connection Agent.
- All Mac users that use the UFT Connection Agent must have the agent installed in their Mac user account.

1. Copy the installer image file (**UFTConnectionAgent.dmg**) from the **<UFT installation folder>/Installations/Safari** folder to the Mac.
2. Open the **UFTConnectionAgent.dmg** image file.
3. Double-click (**HP UFT Connection Agent.pkg**) to start the installation wizard, or **Uninstall** to remove the agent from your Mac.

This installs or removes both the UFT Connection Agent and the Unified Functional Testing Agent Safari extension.

Note: If Safari or the System Preferences pane are open when you uninstall the agent, reopen them for the uninstallation to take effect.

Configure the UFT Connection Agent Preferences (Optional)

1. To check or modify the status of the UFT Connection Agent, or to update its preferences, open **System Preferences** and double-click **HP UFT Connection Agent**.

2. You can modify any of the following preferences (separately for each Mac user):

- **Connection Port Numbers**

- **Safari port.** The port on which the UFT Connection Agent communicates with the Unified Functional Testing Agent Safari extension. **(Default: 8823)**

This number must match the **Remote Agent Port** number defined in the Unified Functional Testing Agent extension in Safari.

- **UFT port.** The port on which UFT communicates with the UFT Connection Agent. **(Default: 8822)**

This number must match the number defined in UFT for initiating the remote connection to the Mac.

- **Security Configuration**

- **Passphrase.** The passphrase that UFT must use for authentication when initiating a remote connection to the Mac computer.

This string must match the passphrase defined in UFT's **Remote Connection** pane in the Options dialog box (**Tools > Options > GUI Testing > Remote Connection**).

The passphrase can be empty, but a long and complicated passphrase that is difficult to guess provides better security.

- **Use SSL.** Indicates that UFT must use SSL when initiating the connection to the Mac computer.

If you select this option, then:

- The **Use SSL** option, available in UFT when initiating a remote connection, must also be selected.
- You must specify the paths to the **SSL certificate file** to use for the communication and its **SSL private key**.
These files must be accessible to the UFT Connection Agent when it sets up the connection. (Chain certificate files are also supported)
- In UFT (**Tools > Options > GUI Testing > Remote Connection**), specify the **CA certificate** that UFT can use to validate the **SSL certificate file** when received from the UFT Connection Agent.

For more details, see "[Securing the Communication With the Remote Mac Computer](#)" on [page 473](#).

- **Log Messages Configuration**

- **Log folder.** The folder in which to save activity log messages.

Make sure you have write permissions for this folder.

- **Agent log level.** The level of information to save in the agent log (Fatal, Error, Warning, Debug).

The log is saved in the **UFTAgent.log** file in the specified folder.

- **UFT Safari extension logs.** Specifies whether to maintain a log of the Unified Functional

Testing Agent Safari extension activity in addition to the UFT Connection Agent log. (**Default: OFF**).

There is no need to turn these logs on, unless HP Support personnel request it.

This log is saved in the **SafariLog.log** file in the same folder.

Note: To activate the Unified Functional Testing Agent Safari extension log, you must also select **Enable Remote Logging** in the extension's preferences in Safari.

3. Make sure the Safari browser is closed and then click **Apply & Save** to save your changes and restart the agent with the new preferences.
 - The agent's status is displayed in the HP UFT Connection Agent preference pane.
Check the status to make sure the agent runs successfully with the new preferences. If it does not, check the UFTAgent.log file for problem details.
 - If you set the **UFT Safari extension logs** option to ON and the remote logger service that logs the Safari extension activity fails to run, this option is automatically set back to OFF. Check the **SafariLog.log** file for problem details.

Configure the Unified Functional Testing Agent Extension in Safari (Optional)

1. Open the Unified Functional Testing Agent extension in Safari.
 - a. Select **Safari > Preferences**
 - b. In the Preferences dialog box, select the **Extensions** tab.
 - c. In the Extensions list in the left pane of the dialog box, select the Unified Functional Testing Agent extension.
2. In the right pane, you can modify any of the following preferences:
 - **Remote Agent Port:** The port number on which the UFT Connection Agent communicates with the Safari browser. (**Default: 8823**)

Caution: This number must match the **Safari port** number defined in the UFT Connection Agent preferences.

- **Show Object Spy popover when Spy starts:** If you turned off the notification displayed on Safari when UFT initiates a Spy session, you can turn it back on using this option.
- Options related to maintaining a log of the Unified Functional Testing Agent Safari extension activity. There is no need to modify these options, unless HP Support personnel request it.
 - **Default Logging Level.** The level of information to save in the Unified Functional Testing Agent Safari extension log file. (All, Trace, Debug, Info, Warn, Error, Fatal, Off)
 - **Enable Remote Logging.** Specifies whether to maintain a log of the Unified Functional Testing Agent Safari extension activity in addition to the UFT Connection Agent log. (By default, this option is cleared.)

This log is saved in the `SafariLog.log` file in the same folder as the UFT Connection Agent logs.

Note: To activate the Unified Functional Testing Agent Safari extension log, you must also select **UFT Safari Extension logs** in the UFT Connection Agent preferences.

- Leave the **Logging Exceptions** box empty. This is intended for use by HP Support only.

Troubleshooting the UFT Connection Agent

- If the agent remains in **not running** status after you update its preferences, check the **UFTAgent.log** file for problem details.

For example, the ports you specified might be busy. They may be used by another application, or they may be in use by a UFT Connection Agent installed on the Mac under another user's account.

Alternatively, you might have entered a port number that is restricted by Apple. Consult the Apple support site regarding restricted or blocked ports.

- If UFT fails to connect to the Mac, make sure that the security settings defined in UFT match those defined in the UFT Connection Agent. For more details, see "[Securing the Communication With the Remote Mac Computer](#)" on page 473.

- If UFT fails to run tests on Safari or to recognize that it is installed on the Mac:

- Make sure that **Safari port** number defined in the UFT Connection Agent preferences, matches the **Remote Agent Port** number defined in the Unified Functional Testing Agent extension.

Make sure that the **UFT port** number defined in the UFT Connection Agent preferences, matches the number defined in UFT for initiating the remote connection to the Mac.

- Make sure that you are not using a port number that is blocked by Safari. Consult the Apple support site regarding restricted or blocked ports.
 - If you manually remove the Unified Functional Testing Agent extension from Safari, or the *HP* UFT Connection Agent preferences pane from the System Preferences, you can reinstall them by reinstalling the UFT Connection Agent on the Mac.
- Reinstalling the UFT Connection Agent resets its preferences to their defaults. Make sure to update the preferences, if necessary, so that the correct port numbers are defined in the agent preferences and in the Safari extension.
- If you turn on the UFT Safari extension logs, and when you save the preferences, the log option is set to OFF, check the **SafariLog.log** file for problem details.

For example, the port used for these logs is the **Safari port + 1**. If this port is busy, the remote logger service fails.

Troubleshooting and Limitations - Internet Explorer

General

If Internet Explorer 9 displays the message: Speeding up browsing by disabling add-ons, choose **Don't disable** or select a bigger threshold value.

Test Objects, Methods, and Properties

- Recognition of test objects when using XPath is very slow for web pages in Quirks Mode and Almost Standards Mode.

Workaround: Convert the web page into Standards Mode by adding or changing the DOCTYPE of the page into `<!DOCTYPE.html>` or `<!DOCTYPE.HTML PUBLIC "-//W3C//DTD HTML 4.01//EN" "http://www.w3.org/TR/html4/strict.dtd"` or disabling XPath capabilities in UFT by clearing the **Learn and run using automatic XPath identifiers** in the **Web > Advanced** pane (**Tools > Options > GUI Testing tab > Web > Advanced** node).

- When using the `RunScript` or `RunScriptFromFile` methods, `Alert`, `Confirm`, or `Prompt` dialogs are not displayed.

Workaround: When writing a `RunScript` method, use the following method syntax:

```
Set wnd=Browser("<browser name>").Page("<page name>").RunScript("window")
wnd.alert()
```

or

```
Browser("<browser name>").Page("<page name>").RunScript "setTimeout(function ()
{alert(); }, 0)"
```

- The `WebXML` test object is not supported on Internet Explorer 9 or later running in standard mode. Accordingly, features related to `WebXML` test objects, such as XML checkpoints and output value steps, are not supported on such browsers.
- In some cases, UFT does not support or recognize `about:blank` tabs in Internet Explorer 11.
- When using Internet Explorer 9, the **innertext**, **outertext**, **innerhtml** and **outerhtml** property values may differ from earlier versions of Internet Explorer. Therefore, using these values in parameters or running checkpoints that use these property values may cause the steps to fail.
- Java applets running on Internet Explorer 11 (with JRE 8, update 45) are not supported.

Recording

- When recording a test on Internet Explorer 8 or earlier with the Active Screen enabled, performance on the site may become very slow. This is due to a performance issue in the JavaScript engine used in these browsers.

Workaround: Record the test using Internet Explorer 9, or disable the Active Screen. For details, see the *HP Unified Functional Testing User Guide*.

- If you are using Internet Explorer with:
 - UAC enabled
 - Protected mode enabled for either the Internet zone or the Intranet zone (**Internet Options** > **Security** tab)
 - Protected mode disabled for the other zone (the Internet or Intranet zone)

UFT does not record steps when switching between an Internet site and an Intranet site.

Workaround: Set the integrity level (Protected mode enabled or not enabled) to be the same for the Internet and Intranet zones.

- If you record a click on an area of an image map that is not mapped to a URL in Microsoft Internet Explorer, UFT will perform a click on the first mapped area of that map during the run session.
- UFT 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.)
- UFT does not record on the Find window of the Microsoft Internet Explorer browser.
- UFT 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).

- 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.

Active Screen

When using Internet Explorer versions 10 or 11, if you use the Active Screen to view objects or add steps to a test or component, the Active Screen does not capture the updated state of the Web page.

Workaround: Open the Developer Tools for the Web page and change the **Document Mode** to **8** or lower.

Troubleshooting and Limitations - Mozilla Firefox

General Limitations

- If two minor versions of Mozilla Firefox are installed on the same computer, and the earlier version was installed after the later version, UFT may not recognize which is the latest version.

- If you are working on a computer where the UAC (User Account Control) option is set to ON, UFT does not support testing on Mozilla Firefox browsers that were installed (or upgraded to a new version) after you installed UFT.

Workaround: After installing Mozilla Firefox on the environment described above, log in as an administrator and open UFT. This enables UFT to install files that are required for Mozilla Firefox support.

- The legacy Functional Testing Agent for Firefox (found at **<UFT installation folder>\Installations\Firefox\AgentLegacy.xpi**) may experience unexpected results on a Windows 8 operating system.
- UFT does not support anonymous content elements in non-XUL frames. (For example, the buttons in the Mozilla Firefox SSL exception page.)
- It is recommended to disable other Mozilla Firefox add-ins when performing tests of your Web application or Web page.
- The SAP WDJ Add-in is not supported on Firefox browsers.
- If you have the Add-ons Manager enabled, in the Web tab of the Run and Record Settings dialog box, if you set Firefox to open to a specific URL, Firefox does not open to the specified URL at the beginning of a recording or run session.

Workaround: Disable the Add-ons Manager before using Firefox.

Recording

- Recording on Mozilla Firefox pages is only possible when the page is fully loaded.
- When recording steps in Mozilla Firefox, additional steps may be recorded.
Workaround: Manually remove the extraneous steps after the recording session ends.
- The following methods are not supported for recording on Mozilla Firefox:
 - **Browser.Home**
 - **Browser.FullScreen**
- When recording on some Mozilla Firefox versions, if you perform a search on **google.com** by entering the search string and then pressing **Enter**, the **Submit** operation is not recorded. Subsequently, when you run the test, UFT enters the search string but does not perform the search.
Workaround: Do one of the following:
 - When recording the search operation, click the search button instead of pressing **Enter**.
 - Manually add the **Submit** (or **Click**) step to the test or component.
- The If Handler option in the "[Web Event Recording Configuration Dialog Box](#)" (described on page [445](#)) works on Mozilla Firefox browsers only if the handler is assigned as an attribute (for example, ``) and not if it is assigned as a property (for example, `aObj.onclick = function() {some code}`)
- The Active Screen is not supported for use with Mozilla Firefox browsers.

- UFT does not record steps performed on browser dialog boxes (such as alert, confirmation, or prompt) if these dialog boxes are opened from a new browser tab.
- When recording on Mozilla Firefox, UFT does not use the location identifier identification property to learn the object.

Test Objects, Methods, Properties, and Checkpoints

- UFT does not support accessing browser dialog boxes (such as alert, confirmation, or prompt) directly in Mozilla Firefox.

Workarounds for Firefox only:

- Use the **HandleDialog** or **GetDialogText** methods described in the **Web** section of the *HP UFT Object Model Reference for GUI Testing*.
- If the **Browser("xxx").Dialog("xxx").Page("xxx")** object is recognized, either use a recovery scenario with the **HandleDialog** method, or a **Browser("xxx").Dialog("xxx").Page("xxx").Type** step to handle the pop-up dialog box. For example, use an Enter key to click the default button and close the dialog box.
- The Object Spy and Checkpoint Properties dialog boxes do not retrieve the current value of edit boxes in Mozilla Firefox dialog boxes.
- The Object Spy and Navigate and Learn dialog boxes do not retrieve the current value of tabmodal dialogs in Mozilla Firefox.
- 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.

- When using Mozilla Firefox, the `innertext`, `outertext`, `innerHTML` and `outerhtml` property values may differ from other browsers. Therefore, using these values in parameters or running checkpoints that use these property values may cause the steps to fail.
- The following test objects, methods, and other Web-specific functionalities are not supported in Firefox:
 - iFrame (with a Javascript source) and Frameset objects are not supported in Mozilla Firefox browsers.
 - Firefox Start Page
 - `about://*` pages
 - Modal or modeless dialog boxes
 - **ViewLink** test object
 - **WebXML** test object (and, accordingly, XML checkpoints and output value steps)
 - **Browser.Object** method
 - **Browser.Stop**
- Text area checkpoints are not supported on Mozilla Firefox.

- If you drag a tab to create a separate window in Firefox, UFT stops recognizing, recording, or running any web objects in the new window.
- If you need to test Java applets in Mozilla Firefox browsers, you must use the legacy Functional Testing Agent for Firefox, found in the **<UFT installation folder>\Installations\Firefox** folder.
- If you take a snapshot of an ASPAjaxRichTextArea object on FireFox, it might not be displayed correctly in the Active screen.
- If a test or business component contains a step that closes a Mozilla Firefox browser, UFT may behave unexpectedly when that step is reached during a run session.

Workaround: Do not include a step that closes a Mozilla Firefox browser.

- If you open the Search toolbar or developer toolbar in Mozilla Firefox, when you spy on an object or highlight an object in the object repository, the highlight rectangle is displayed in the wrong location in the browser window.

Workaround: Float the toolbar in the browser window.

Troubleshooting and Limitations - Google Chrome and Apple Safari

Google Chrome and Apple Safari

See below for additional issues relevant only to [Chrome](#) or [Safari](#).

Functionality and Settings

- Web pages that modify the browser's JavaScript functionality (for example, a Web page that replaces the JSON object) may cause UFT to behave unexpectedly.
- The font and color properties for link objects contain different values in different browsers. Therefore, if you create standard checkpoints in Microsoft Internet Explorer and select the **font** and **color** properties, running these checkpoints in Chrome or Safari may cause the checkpoints to fail.
- If you have multiple Chrome or Safari users defined, you must delete all users.

Test Objects, Methods and Properties

- When using Chrome or Safari, the **innertext**, **outertext**, **innerhtml** and **outerhtml** property values may differ from other browsers. Therefore, using these values in parameters or running checkpoints that use these property values may cause the steps to fail.
- The following test objects, methods, and other Web-specific functionalities are not supported in Chrome or Safari:
 - **ViewLink** test object
 - `chrome://*` pages

- `about://*` pages
- **Browser.Home** method
- **Browser.FullScreen** method
- **Browser.Object** method
- **Browser.Stop**
- Modal or modaless dialog boxes
- Developer Tools pane. (Running steps on Chrome or Safari while the Developer Tools pane is open is supported.)
- Dialog boxes opened by the browser, such as Alert, Confirmation, or Prompt messages on versions of Chrome earlier than 26.
- **WebXML** test object (and, accordingly, XML checkpoints and output value steps).
- Web-based environments, such as Web-based SAP, Siebel, Java, .NET Web Forms, and so on.

Google Chrome

- In the following cases, you must manually enable the Functional Testing Agent for Google Chrome extension:
 - You have no internet connection
 - You have not enabled the automatic updates for Google Chrome
 - You are using Google Chrome versions 31 or earlier.For details on manually enabling the Extension, see ["How to Enable the HP Functional Testing Agent Chrome Extension" on page 479](#).
- Recording on Google Chrome is supported only from versions 31 and higher.
- If you have the Chrome developer tools currently open, UFT cannot spy on Web objects.
- If you have a Chrome alert or warning dialog open, UFT cannot run or record tests or components on the browser.
Workaround: Close the alert or warning dialog before trying to run or record a test or component.
- Internal Google Chrome pages, such as the `about:blank` page, Google Chrome sign-in page, Google Chrome Web Store, and Google Chrome's default tabs homepage are not recognized as Web pages but **WinObjects**.
- When working with Chrome version 36 or earlier, UFT does not support Web test objects located inside `iFrame` controls with a `blank` or `about:blank` **SRC** identification property value.
- Page checkpoints and bitmap checkpoints may fail when running on Google Chrome because of differences between Chrome and Internet Explorer. For page checkpoints this is related to differences in the handling of casing in HTML source files.
- Text checkpoints fail when running on Google Chrome.
- When recording on Google Chrome, UFT does not use the **location** identification property to learn the object.

- When spying on a Web file in Google Chrome, you get a fakepath for a Webfile value property.
- If you manually uninstall the UFT Agent extension from Chrome, you must manually reinstall it if you reinstall UFT.
- The Active Screen is not supported for use with Google Chrome browsers.
- The Page/Frame options (**Tools > Options > GUI Testing** tab > **Web > Page/Frame Options** node) are not supported for recording in Google Chrome.
- If the **Record Coordinates** option is selected in the Web > Advanced pane of the Options dialog box (**Tools > Options > GUI Testing** tab > **Web > Advanced > Record Settings** section), UFT does not record correct coordinates on images in Google Chrome browsers.

Workaround: Manually add the correct coordinates for the image after recording.

- iFrame and Frameset objects are not supported in Google Chrome browsers.
- Using the Object Spy on Alert or Confirm dialogs in Google Chrome causes unexpected behavior in UFT.

Apple Safari

- Only Web test object steps can run on Safari. All other steps run locally on the UFT computer.
- Recording steps on Safari is not supported.
- Web 2.0 test objects or Web Add-in Extensibility-based test objects are not supported on Safari Browsers
- UFT does not recognize internal Safari pages, such as the New Tab page.
- The following functionalities are not supported when working with the Safari browser:
 - Learning objects in the Object Repository Window or Object Repository Manager (you can learn objects using the Remote Object Spy)
 - Maintenance Mode
 - Highlighting an object from the object repository in the application
 - Creating an Active Screen or using the Update Run Mode to update the Active Screen
 - Checkpoints and output values that are not standard (such as File Content checkpoints and output values, Text checkpoints and output values, and Bitmap checkpoints).

Note that Page checkpoints are supported, with the following limitations:

- Page checkpoints and bitmap checkpoints may fail when running on Google Chrome because of differences between Chrome and Internet Explorer. For page checkpoints this is related to differences in the handling of casing in HTML source files.
- The value of the **load time** identification property for Page and Frame test objects is always 0 when running on Safari. In Page checkpoints, the **load time** property is selected by default, therefore Page checkpoints may fail when running on Safari.

Workaround: Deselect the **load time** property in Page checkpoints that need to run on Safari.

- The following test objects and test object methods are not supported when running on Safari:
 - **WebFile.Set** method
 - **Browser.ClearCache** method
 - **Browser.DeleteCookies** method
 - **Drag & Drop** methods (on all Web test objects)
 - Web test objects located inside iFrame controls with a blank or about:blank **SRC** identification property value.
- Depending on the performance of your connection to the Mac, statement completion for the **Object** property might not work.
- During a run session, steps that click on a Web object that is supposed to open a new browser tab fail to open the new tab.

Workaround: Disable the popup blocker in Safari. (**Safari > Preferences > Security**, clear the **Block pop-up windows** check box).
- When running tests or components on Safari, recovery scenarios are not supported.
- Running tests or components on Safari using the **Mouse** replay type is not supported.

Make sure that **Event** is selected as the **Replay type** in the **Run settings** section of the **Web > Advanced** options pane (**Tools > Options > GUI Testing** tab > **Web** pane > **Advanced** node).
- In the automation object model, **WebLauncher.Browser** does not support the Safari browser. To connect to a remote Mac during a test run and open the Safari browser, use the **RemoteConnection** utility object and its methods. For details, see the **Utility Objects** section of the *HP UFT Object Model Reference for GUI Testing*.

Part 19: Web 2.0 Add-ins

This section includes:

["Web 2.0 Add-ins - Quick Reference" on page 496](#)

["Web 2.0 Toolkit Support " on page 498](#)

["Troubleshooting and Limitations - Web 2.0 Add-ins" on page 501](#)

Web 2.0 Add-ins - Quick Reference

You can use the Web 2.0 Add-ins to test HTML user-interface objects (controls) in Web 2.0 environments.

The Web 2.0 Add-ins include the following:

- ASP .NET AJAX
- Dojo
- Google Web Toolkit (GWT)
- jQueryUI
- Siebel Open UI
- EXT-JS
- Yahoo User Interface (Yahoo UI)

The following tables summarize basic information about the Web 2.0 Add-ins and how they relates to some commonly-used aspects of UFT.

General Information	
Add-in Type	<p>Much of the functionality of these add-ins is the same as other Web add-ins. These add-ins extend the capabilities of the Web Add-in functionalities.</p> <ul style="list-style-type: none"> • See the section on the "Web Add-in" on page 432 and "Web-Based Application Support" on page 34.
Supported Environments	<p>For details on supported Web 2.0 toolkits and versions, see the <i>HP Unified Functional Testing Product Availability Matrix</i>, available from the HP Support Matrix page (requires an HP passport).</p>
Test Object Methods and Properties	<p>The Web 2.0 Add-ins provide test objects, methods, and properties that can be used when testing objects in Web applications. For details, see the relevant toolkit section in the Web 2.0 toolkits section of the <i>HP UFT Object Model Reference for GUI Testing</i>.</p>
Checkpoints and Output Values	<ul style="list-style-type: none"> • See the sections describing checkpoints and output values in the <i>HP Unified Functional Testing User Guide</i>. • See "GUI Checkpoints and Output Values Per Add-in" on page 512

Prerequisites	
Opening Your Application	You must open UFT before opening your Web application.
Add-in Dependencies	You must have the Web Add-in installed and loaded.
Other Important Information for SiebelOpenUI users	<ul style="list-style-type: none"> • When working with SiebelOpenUI objects, load the Web Add-in and the SiebelOpenUI Add-in, but do not load the Siebel Add-in. • If you load both the Siebel and the SiebelOpenUI add-ins, the add-ins sometimes conflict with each other, and prevent successful object recognition. • In order to enable UFT to run tests for SiebelOpenUI objects, you must associate a special

	<p>function library with your test.</p> <p>This function library is found at <UFT installation folder>\dat\Extensibility\Web\Toolkits\SiebelOpenUI\Function Libraries\SiebelOpenUI.qfl.</p> <p>You can set this function library as the default function library in the Resources pane in the Test Settings dialog box. For details, see the section on the Test Settings Resources pane in the <i>HP Unified Functional Testing User Guide</i>.</p>
Other Important Information for EXT-JS users	<p>In order to enable UFT to run tests for EXT-JS objects, you must associate a special function library with your test.</p> <p>This function library is found at <UFT installation folder>\dat\Extensibility\Web\Toolkits\ExtJS\Function Libraries\ExtJS.qfl.</p> <p>You can set this function library as the default function library in the Resources pane in the Test Settings dialog box. For details, see the section on the Test Settings Resources pane in the <i>HP Unified Functional Testing User Guide</i>.</p>

Configuration	
Options Dialog Box	<p>Use the Web pane.</p> <p>(Make sure that a GUI test is open and select Tools > Options > GUI Testing tab > Web > General node.)</p> <p>For details, see "Web > General Pane (Options Dialog Box)" on page 58.</p>
Record and Run Settings Dialog Box (tests only)	<p>Use the Web tab. (Record > Record and Run Settings)</p> <p>See "Web Tab (Record and Run Settings Dialog Box)" on page 71.</p>
Test Settings Dialog Box (tests only)	<p>Use the Web pane (File > Settings > Web pane).</p> <p>See "Web Pane (Test/Business Component Settings Dialog Box / Application Area - Additional Settings Pane)" on page 68.</p>
Custom Active Screen Capture Settings Dialog Box (tests only)	<p>Use the Web section (Tools > Options > GUI Testing tab > Active Screen node > Custom Level).</p> <p>See the section on the Custom Active Screen Capture Settings dialog box in the <i>HP Unified Functional Testing User Guide</i>.</p>
Application Area Additional Settings pane (business components only)	<p>Use the Web pane.</p> <p>In the application area, select Additional Settings > Web in the sidebar.</p> <ul style="list-style-type: none"> • See "Web Pane (Test/Business Component Settings Dialog Box / Application Area - Additional Settings Pane)" on page 68. • See the section on defining Application Settings for your application area in the <i>HP Unified Functional Testing User Guide</i>.

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 UFT 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 UFT Web 2.0 Add-in Support

UFT 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.

UFT 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 UFT setup, or on your UFT computer:

<UFT installation folder>\Installations\Web2AddinSetup\Web2AddinSetup.exe.

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 UFT functionality on controls from these toolkits.

UFT provides support for the following toolkits:

- ASP .NET Ajax - <http://www.asp.net/ajax/>
- Dojo - <http://www.dojotoolkit.org>

- Google Web Toolkit (GWT) - <http://code.google.com/webtoolkit/>
- jQuery UI - <http://jqueryui.com/>
- Siebel Open UI
- Yahoo User Interface (Yahoo UI) - <http://developer.yahoo.com/yui/>
- EXT-JS: <http://www.sencha.com/products/extjs/S>

For details on the test objects and operations supported for these toolkits, see the **Web 2.0 Toolkits** section of the *HP UFT Object Model Reference for GUI Testing*.

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, UFT injects the jQuery JavaScript library into every Web page that opens in a browser while UFT is open (unless a jQuery library is already included in the page).

The specific jQuery UI file injected for each Web 2.0 add-in is specified in the add-in's toolkit XML file, located in: `<UFT 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 UFT. If these files were customized or modified in any way, the details in the Help files supplied with UFT 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 501. 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 UFT maps Web controls to that class.

In the example below, UFT identifies a control as a GWTToggleButton 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="GWTToggleButton">  
  <Settings>  
    <Variable name="default_imp_file" value="JavaScript\GWTToggleButton.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 UFT to correctly identify those controls.

If UFT 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: <UFT 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 the next page.](#)

For more details on the way UFT identifies supported controls and for details 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 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 UFT to work with your custom Web controls.

Extensibility Accelerator also comes with built-in projects for the UFT 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, the setup stores the previous files in a backup folder before installing. You may need to merge any customizations you made to the previous version into the new version.

For details on how to make these changes and how to customize the support files, see the UFT Web Add-in Extensibility documentation, available in the <UFT installation folder>\help\Extensibility folder.

For details on working with Extensibility Accelerator, see the *HP UFT Extensibility Accelerator for HP Functional Testing User Guide*.

Troubleshooting and Limitations - Web 2.0 Add-ins

This section contains troubleshooting and limitation information about working with the Web 2.0 Add-ins.

General Limitations

- When working with test objects that are supported using Web Add-in Extensibility, such as Web 2.0 test objects, if you create a checkpoint from the Active Screen, or try to view the object's properties from the Active Screen, some property values may be empty.

Note: Using the Active Screen for Google Chrome is not supported.

- When working with Web 2.0 toolkits on Firefox browsers, it is recommended to load only one toolkit at a time.

Browser Specific Limitations

- Due to synchronization issues, if you navigate to a new Web page in Internet Explorer or Mozilla Firefox while recording, then UFT may not record certain operations on certain ASP .NET Ajax or jQuery UI objects in the page. Similarly, when running steps that navigate to a new page, UFT may fail to perform certain steps on certain ASP .NET Ajax or jQuery UI objects.

Workaround: If the problem occurs while recording, refresh the Web page and record the step again. If the problem occurs while running, insert a `Wait()` statement before the problematic step.

- Web 2.0 test objects or Web Add-in Extensibility-based test objects are not supported on Safari browsers.

ASP .NET AJAX

- When working with the Web 2.0 ASPAjax Add-in, running scripts in the Active Screen is not enabled by default.

Workaround:

- a. In the Options dialog box, enable running scripts in the Active Screen. In **Tools > Options > GUI Testing** tab > **Active Screen** node, set **Run scripts** to **Enabled**.
- b. Close and reopen your test or component for the setting to take effect.

Dojo

You cannot not record on objects created with Dojo 1.10 in Internet Explorer 10 or 11.

Workaround: Record the objects on Internet Explorer 9, Firefox, or Chrome.

Siebel Open UI

- In order to enable UFT to run tests for SiebelOpenUI objects, you must associate a special function library with your test.

This function library is found at <UFT installation folder>\dat\Extensibility\Web\Toolkits\SiebelOpenUI\Function Libraries\SiebelOpenUI.qfl.

You can set this function library as the default function library in the Resources pane in the Test Settings dialog box. For details, see the section on the Test Settings Resources pane in the *HP Unified Functional Testing User Guide*.

- By default, all SiebelOpenUI objects appear as child objects of the same Page object, even for objects embedded in objects that are opened from the main page.

If you need to group objects under separate Page objects, select the **Every navigation** option under the **Create a new page test object for section** in the **Page/Frame Options** pane of the Options Dialog box (**Tools > Options > GUI Testing** tab > **Web** pane > **Page/Frame Options**).

- When recording entering information for a SblOUIAdvancedEdit object, UFT records the Set method for the entered information after the opening of another object.

Workaround: Manually change the order of the steps after recording.

- When selecting a method for Siebel OpenUI objects, the autocompletion list displays some of the object's internal methods. These methods should not be used in your test.

Part 19: Windows Runtime Add-in

This section includes:

["Windows Runtime Add-in - Quick Reference" on page 505](#)

["Using the Windows Runtime Add-in in UFT - Overview" on page 507](#)

["How to Use UFT in a Windows Runtime Environment" on page 507](#)

["Troubleshooting and Limitations - Windows Runtime" on page 509](#)

Chapter 35: Windows Runtime Add-in - Quick Reference

You can use the Windows Runtime Add-in to test Windows Runtime applications (from the Windows Store) created to run on Windows Runtime environments (Windows 8.x and later, Windows RT, or Windows Server 2012).

Note: If you are testing a non-Windows Runtime application (that opens from the Desktop section in Windows 8.x or higher or Windows Server 2012), you should use the Standard Windows Add-in. For details on the Standard Windows add-in, see ["Standard Windows Support -Quick Reference" on page 347.](#)

The following tables summarize basic information about the Windows Runtime Add-in and how it relates to some commonly-used aspects of UFT.

General Information	
Add-in Type	<ul style="list-style-type: none"> 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. For details, see "Windows-Based Application Support" on page 81. There are some significant differences between the Windows Runtime add-in and the other Windows-based add-ins, due to the changes implemented in Windows 8.x and Windows Server 2012. For details, see "Using the Windows Runtime Add-in in UFT - Overview" on page 507.
Supported Environments	<ul style="list-style-type: none"> For details on supported technologies and versions, see the <i>HP Unified Functional Testing Product Availability Matrix</i>, available from the HP Support Matrix page (requires an HP passport).
Test Object Methods and Properties	The Windows Runtime Add-in provides test objects, methods, and properties that can be used when testing objects in Windows Runtime applications. For details, see the Windows Runtime section of the <i>HP UFT Object Model Reference for GUI Testing</i> .
Checkpoints and Output Values	<ul style="list-style-type: none"> See the sections describing checkpoints and output values in the <i>HP Unified Functional Testing User Guide</i>. See "GUI Checkpoints and Output Values Per Add-in" on page 512
Troubleshooting and Limitations	See "Troubleshooting and Limitations - Windows Runtime" on page 509.
UFT4WinRT Service	<p>When you install UFT, a service named UFT4WinRT is installed on your computer. By default, this service is running on your computer.</p> <p>This service enables you to use UFT in the Windows Runtime environment through the following:</p> <ul style="list-style-type: none"> Sets the integrity level of UFT to High (default is Medium) Runs UFT in elevated mode Runs UFT with the UAC for the user enabled. <p>This service enables you to use common UFT tools, such as the Object Spy, Navigate and Learn toolbar, and recording for a Windows Runtime application.</p>

Prerequisites	
Opening Your Application	<p>You can open your Windows Runtime application before or after opening UFT.</p> <p>When UFT is installed on a computer running Windows 8.x or later or Windows Server 2012, the Windows Runtime testing support is always enabled. This add-in is therefore not listed in the Add-in Manager.</p>
Add-in Dependencies	None
Configuration	
Options Dialog Box	<p>Use the Windows Applications pane (Tools > Options > GUI Testing tab > Windows Applications node).</p> <p>(Make sure that a GUI test is open and select Tools > Options > GUI Testing tab > Web > General node.)</p> <p>For details, see "Windows Applications > General Pane (Options Dialog Box > GUI Testing Tab)" on page 85.</p>
Record and Run Settings Dialog Box (tests only)	<p>Use the Windows Applications tab. (Record > Record and Run Settings)</p> <ul style="list-style-type: none"> The Applications opened by UFT and Applications opened via the Desktop (by the Windows shell) options are not supported for the Windows Runtime Add-in. For the Applications specified below option, you enter the information differently, depending on the type of application being tested: <ul style="list-style-type: none"> For WPF and XAML-based applications: the name of the .exe process of the application <div style="background-color: #f0f0f0; padding: 5px; margin: 5px 0;"> <p>Note: You do not need to enter the location of the working folder for the application.</p> </div> <ul style="list-style-type: none"> For HTML and Javascript-based applications: WWAHOST.exe Analog recording and Low level recording are not supported for the Windows Runtime add-in.
Custom Active Screen Capture Settings Dialog Box (tests only)	<p>Use the Windows applications section (Tools > Options > GUI Testing tab > Active Screen node > Custom Level).</p> <p>See the section on the Custom Active Screen Capture Settings dialog box in the <i>HP Unified Functional Testing User Guide</i>.</p>
Application Area Additional Settings pane (business components only)	<p>Use the Applications pane.</p> <p>In the application area, select Additional Settings > Applications in the sidebar.</p> <p>See the section on defining Application Settings for your application area in the <i>HP Unified Functional Testing User Guide</i>.</p>

Using the Windows Runtime Add-in in UFT - Overview

The Windows Runtime Add-in enables you to test applications running in a Windows Runtime environment. A **Windows Runtime environment** describes the area in Windows 8.x or higher or Windows Server 2012 that runs applications from the Windows Store.

Note: On Windows 8.x or higher or Windows Server 2012, UFT uses the Standard Windows Add-in to test applications that run on the desktop.

You can test different types of Windows Runtime applications using UFT:

- WPF or XAML-based Windows applications

For details on WPF or XAML-based applications, see <http://msdn.microsoft.com/en-us/magazine/jj651571.aspx>.

- HTML or JavaScript-based Windows applications

For details on HTML-based Windows Runtime application, see <http://msdn.microsoft.com/en-us/library/windows/apps/hh770842.aspx>. For details on Javascript-based Windows Runtime applications, see [http://msdn.microsoft.com/en-us/library/hh710230\(v=vs.94\).aspx](http://msdn.microsoft.com/en-us/library/hh710230(v=vs.94).aspx).

Note: If you are testing an HTML or JavaScript-based Windows application, you must also load the Web add-in when starting UFT.

For basic configuration details for the Windows Runtime Add-in, see "[Windows Runtime Add-in - Quick Reference](#)" on page 505.

How to Use UFT in a Windows Runtime Environment

This task describes how to perform common UFT tasks in a Windows Runtime environment.

This task includes the following steps:

- "[Prerequisites](#) " below
- "[Display UFT and the Windows Runtime application together](#)" on the next page
- "[Use UFT tools with a Windows Runtime application](#) " on the next page

Prerequisites

You must meet the following requirements before using the Windows Runtime add-in:

- UFT must be installed on a computer running Windows 8.x or higher or Windows Server 2012
- The UFT4WinRT service must be running. For details on the service, see the [description of the service](#).

Display UFT and the Windows Runtime application together

UFT runs on the desktop in Windows 8.x or higher or Windows Server 2012. Therefore, direct interaction between UFT dialog boxes, panes, and windows is difficult. To enable UFT to work with Windows Runtime applications available on the **Start** screen, you can resize the open window:

- Dock the desktop window containing UFT on an edge of the screen:
 - a. On the Desktop window, move the mouse to the upper edge of the window. The mouse cursor changes to a hand.
 - b. Drag the desktop window toward the bottom of the screen.
 - c. Move the dragged window to either edge of the screen.
 - d. Resize the window as needed.

Note: The window can be resized to use 1/3 or 2/3 of the screen.

- Use UFT in full screen view.

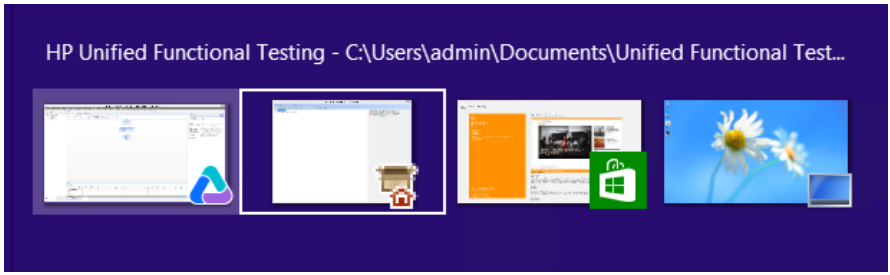
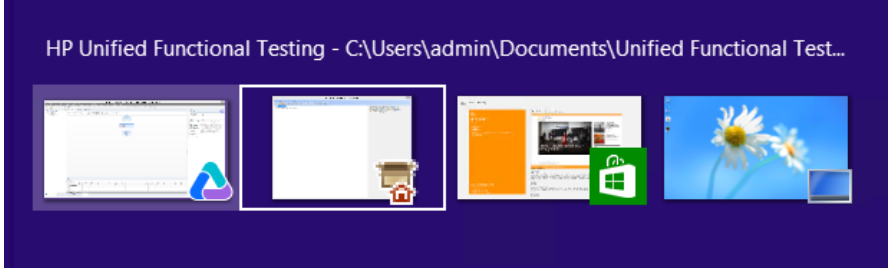
By default, the desktop window containing your UFT session is displayed at full screen view. To navigate to your Windows Runtime application, use the keyboard shortcuts described in the following steps.

Use UFT tools with a Windows Runtime application

Because UFT runs on the Desktop screen in Windows 8.x or higher, direct navigation between the Object Spy dialog box, Navigate and Learn toolbar, and the Record Toolbar is difficult. Use the following steps to enable UFT to access your Windows Runtime application:

1. Open the relevant UFT tool.
2. Do one of the following:

Object Spy	Press the Windows + SHIFT key. Windows cycles through all open Windows Runtime applications. After selecting your application, the Object Spy dialog box is displayed on top of the open application. <p>Note: If you use this option to switch to your application, the Keep on Top option in the Object Spy dialog box does not work.</p>
Navigate and Learn toolbar	a. Press ALT + TAB . Windows displays a window displaying the list of open applications:

	 <p>b. Using the arrow keys, select your application. When the Windows Runtime application is displayed in the main window, the Navigate and Learn toolbar is displayed on top of the open application.</p>
<p>Record Toolbar</p>	<p>a. Press ALT + TAB. Windows displays a list of open applications.</p>  <p>b. Using the arrow keys, select your application. When the Windows Runtime application is displayed in the main window, the Record toolbar is displayed on top of the open application.</p>

Note: If you close your application using **ALT + F4** while recording or spying on an object, the application automatically reopens when continuing to record or spy.

- Continue spying on objects, adding objects to the object repository, or recording actions as necessary.

Troubleshooting and Limitations - Windows Runtime

- You cannot use the **.Object** property to invoke native methods for a Windows Runtime object.
- Analog recording and Low level recording are not supported for the Windows Runtime add-in.
- If you modify the mandatory and assistive identification properties of a Windows Runtime object in the Object Identification dialog, UFT does not recognize the modified properties when spying on or adding the same object in the same UFT session.

Workaround: Restart UFT and spy on or add the object again.

- If you close a Windows Runtime application using **ALT + F4** while recording, UFT does not record the application close.

Workaround: Manually add a **WindowsApp.Close** statement step after recording.

- UFT cannot record or spy on a Windows Runtime application that is in a suspended state.
Workaround: Restart the Windows Runtime application.
- UFT cannot record over applications that have a higher integrity level, such as setup programs or programs that are run as administrator.
Workaround: Run UFT as administrator if you need to record over an application with a higher integrity level.

Part 20: Appendix

This section includes:

["GUI Checkpoints and Output Values Per Add-in" on page 512](#)

Appendix A: GUI Checkpoints and Output Values Per Add-in

The tables in this chapter show the categories of checkpoints and output values that are supported by UFT for each add-in.

For details about using checkpoints and output values in a specific add-in, see the relevant add-in section.

This chapter includes:

- ["Supported Checkpoints " on the next page](#)
- ["Supported Output Values " on page 515](#)

Supported Checkpoints

The following table shows the categories of checkpoints that are supported by UFT for each add-in.

Table Legend

- S: Supported
- NS: Not Supported
- NA: Not Applicable

Note: Only standard and bitmap checkpoints are supported for keyword components.

For additional information, see ["Footnotes" on the next page.](#)

	Accessibility	Bitmap	Database	File Content	Image	Page	Standard	Table	Text	Text Area	XML (Application)	XML (Resource)
.NET Web Forms³	S	S	NA	NA	NA	NA	S	S	S ⁶	S ⁶	S	S
.NET Windows Forms	NA	S	NA	NA	NA	NA	S	S	S ⁶	S ⁶	NA	NA
ActiveX	NS	S	NA	NA	NS	NA	S	S	S	S	NA	NA
Delphi	NS	S	NA	NA	NS	NA	S	S	S	S	NA	NA
Flex	NA	S	NA	NA	NA	NA	S	S	S	S	NA	NA
Java	NA	S	NA	NA	NA	NA	S	S	S	S ⁴	NA	NA
Mobile	NA	S	NA	NA	NA	NA	S	NA	S	NS	NA	NA
Oracle	NA	S	NA	NA	NA	NA	S	S	NS	NS	NA	NA
PeopleSoft	S	S	NA	NA	S	S	S	S	S ¹	NS	S	S
PowerBuilder²	NS	S	NA	NA	NS	NA	S	S	S	S	NA	NA
Qt	NS	S	NA	NA	NS	NA	S	S	S	S	NA	NA
SAP Web-based	S	S	NA	NA	S	S	S	S	S	NS	S	S
SAP Windows-based	S ⁵	S	NA	NA	S ⁵	S ⁵	S	S	S ⁵	NS	S ⁵	NA

	Accessibility	Bitmap	Database	File Content	Image	Page	Standard	Table	Text	Text Area	XML (Application)	XML (Resource)
Siebel	S	S	NA	NA	S	S	S	S	S	NS	S	S
Silverlight	NA	S	NA	NA	NA	NA	S	S	S	S	NA	NA
Standard Windows	NS	S	NA	NA	NS	NA	S	S	S	S	NA	NA
Stingray	NA	S	NA	NA	NA	NA	S	S	S	S	NA	NA
Terminal Emulator	NA	S	NA	NA	NA	NA	S	NA	NA	NA	NA	NA
VisualAge for Smalltalk	NA	S	NA	NA	NA	NA	S	S	S	S	NA	NA
Visual Basic	NS	S	NA	NA	NS	NA	S	S	S	S	NA	NA
Web	S	S	NA	NA	S	S	S	S	S ¹	S	S ⁷	NA
Windows Runtime	NA	S	NA	NA	NA	NA	S	S	S	S	NA	NA
WPF	NA	S	NA	NA	NA	NA	S	S	S	S	NA	NA

Footnotes

1 Text checkpoints are supported only for Page, Frame, and ViewLink objects.

2 When you insert a checkpoint on a PowerBuilder DataWindow control, UFT treats it as a table and opens the Table Checkpoint Properties dialog box.

3 For NET Web Forms, text checkpoints for WbfTreeView, WbfToolBar, and WbfTabStrip objects are not supported.

4 The text area checkpoint mechanism for Java Applet objects is disabled by default. You can enable it in the Advanced Java Options dialog box.

5 This is supported only when UFT 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).

6 This is supported only when UFT is configured to use the OCR (optical character recognition) mechanism.

7 XML checkpoints are not supported on Internet Explorer 9 or later running in standard mode, on Google Chrome, on Mozilla Firefox, or on Apple Safari because the WebXML test object is not supported for these browsers.

8 - Checkpoints on Mobile object are supported only when recording your test.

Supported Output Values

The following table shows the categories of output values that are supported by UFT for each add-in.

Table Legend

- **S:** Supported
- **NS:** Not Supported
- **NA:** Not Applicable

Note: Only standard and bitmap output values are supported for keyword components.

For additional information, see ["Footnotes" on the next page.](#)

	Accessibility	Bitmap	Database	File Content	Image	Page	Standard	Table	Text	Text Area	XML (Application)	XML (Resource)
.NET Web Forms	NA	NA	NA	NA	NA	S	S	S	S ⁵	S ⁵	NA	NA
.NET Windows Forms	NA	NA	NA	NA	NA	NA	S	S	S ⁵	S ⁵	NA	NA
ActiveX	NS	NA	NA	NA	NA	NA	S	S	S	S	NA	NA
Delphi	NS	NA	NA	NA	NA	NA	S	S	S	S	NA	NA
Java	NA	NA	NA	NA	NA	NA	S	NA	S	S ³	NA	NA
Mobile	NA	NA	NA	NA	NA	NA	S	NA	S	NS	NA	NA
Oracle	NA	NA	NA	NA	NA	NA	S	S	NA	NA	NA	NA
PeopleSoft	NA	NA	NA	NA	NA	S	S	S	S ¹	NS	S	S
PowerBuilder²	NA	NA	NA	NA	NA	NA	S	NA	S	S	NA	NA
Qt	NA	NA	NA	NA	NA	NA	S	S	S	S	NA	NA
SAP Web-based	NA	NA	NA	NA	NA	S	S	S	S	NS	S	S
SAP Windows-based	NA	NA	NA	NA	NA	S ⁴	S	S	S ⁴	NS	S ⁴	S

	Accessibility	Bitmap	Database	File Content	Image	Page	Standard	Table	Text	Text Area	XML (Application)	XML (Resource)
Siebel	NA	NA	NA	NA	NA	S	S	S	S	NS	S	S
Silverlight	NA	NA	NA	NA	NA	NA	S	S	S	S	NA	NA
Standard Windows	NA	NA	NA	NA	NA	NA	S	S	S	S	NA	NA
Stingray	NA	NA	NA	NA	NA	NA	S	S	S	S	NA	NA
Terminal Emulator	NA	NA	NA	NA	NA	NA	S ⁸	NA	S ⁷	NA	NA	NA
VisualAge for Smalltalk	NA	NA	NA	NA	NA	NA	NA	S	S	S	NA	NA
Visual Basic	NA	NA	NA	NA	NA	NA	S	NA	S	S	NA	NA
Web	NA	NA	NA	NA	NA	S	S	S	S ¹	NS	S ⁶	NA
Windows Runtime	NA	NA	NA	NA	NA	NA	S	S	S	S	NA	NA
WPF	NA	NA	NA	NA	NA	NA	S	S	S	S	NA	NA

Footnotes

- 1 Text output values are supported only for Page, Frame, and ViewLink objects.
- 2 When you insert an output value step on a PowerBuilder DataWindow control, UFT treats it as a table and opens the Table Output Value Properties dialog box.
- 3 The text area output mechanism for Java Applet objects is disabled by default. You can enable it in the Advanced Java Options dialog box.
- 4 This is supported only when UFT 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).
- 5 This is supported only when UFT is configured to use the OCR (optical character recognition) mechanism.
- 6 XML output values are not supported on Internet Explorer 9 or later running in standard mode, on Google Chrome, or on Mozilla Firefox, because the WebXML test object is not supported for these browsers.
- 7 You can create text output values (tests only) only for TeScreen and TeTextScreen objects.
- 8 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. UFT recognizes these as standard Windows objects.

Accessing UFT in Windows 8.X or Higher Operating Systems

By default, you can access UFT directly from the **Start** or **Apps** Screen in Windows 8.x or higher.

In addition, you can add UFT tools and files that were accessible from the **Start** menu in previous versions of Windows to the **Start** screen, including:

- **Applications (.exe files).** For example:
 - The Run Results Viewer
 - All UFT tools, such as the Password Encoder and the License Validation Utility
 - The API testing sample Flight applications
- **Non-program files.** You can access documentation and the link to the Mercury Tours Website from the **Apps** screen.

Note: By default, the Start and Apps screens on Windows 8.x or higher are set to open Internet Explorer in Metro Mode. However, if User Account Control is turned off on your computer, Windows 8 will not open Internet Explorer in Metro mode. Therefore, if you try to open an HTML shortcut from the Start or Apps screen, such as the UFT Help or Readme file, an error will be displayed.

To solve this, you can change the default behavior of Internet Explorer so that it never opens in Metro mode. In the **Internet Properties** dialog box > **Programs** tab, select **Always in Internet Explorer on the desktop** for the **Choose how you open links** option. For more details, see <http://support.microsoft.com/kb/2736601> and <http://blogs.msdn.com/b/ie/archive/2012/03/26/launch-options-for-internet-explorer-10-on-windows-8.aspx>.

Send Us Feedback



Can we make this Add-ins Guide better?

Tell us how: sw-doc@hp.com

